Clavier DUCK : Utilisation d'un système de déduction de mots pour faciliter la saisie de texte sur écran tactile pour les non-voyants

Philippe Roussille, Mathieu Raynal, Christophe Jouffrais

Clavier DUCK : Utilisation d'un système de déduction de mots pour faciliter la saisie de texte sur écran tactile pour les non-voyants.

Philippe Roussille
Université de Toulouse & CNRS ; IRIT ;
F31 062 Toulouse, France
philippe.roussille@irit.fr

Mathieu Raynal
Université de Toulouse & CNRS ; IRIT ;
F31 062 Toulouse, France
mathieu.raynal@irit.fr

Christophe Jouffrais
CNRS & Université de Toulouse ; IRIT ;
F31 062 Toulouse, France
christophe.jouffrais@irit.fr

RÉSUMÉ
L'utilisation des écrans tactiles et en particulier les claviers logiciels est extrêmement compliquée pour les non-voyants qui manquent de repères physiques sur ce type d'appareil. Nous proposons dans cet article une solution clavier logicielle qui propose une liste de mots pouvant correspondre au mot recherché à partir de frappes approximatives des utilisateurs non-voyants. Cette technique évite ainsi à l'utilisateur d'explorer le clavier en permanence pour trouver précisément les caractères à saisir. Une première évaluation nous permet de montrer que notre système est efficace pour les mots de plus de quatre caractères. Il permet aussi d'éviter certains types d'erreur de frappe.

Mots Clés
Saisie de texte ; déficience visuelle ; écran tactile ; dispositifs mobiles ; clavier logiciel ; système déductif.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): User Interfaces — Input devices and strategies.

INTRODUCTION

Bien que les premières utilisations des mobiles étaient limitées à l'appel et l'envoi de petits messages textuels (SMS), les appareils mobiles ont évolué, et encouragent maintenant l'utilisation de différents services, comme les applications web et les réseaux sociaux. L'émergence des tablettes a également permis, au détriment des ordinateurs de bureau, l'utilisation de ces dispositifs mobiles pour la plupart des tâches bureautiques habituelles (email, prise de notes, support de présentation, etc.). D'autre part, les claviers physiques ont presque disparu de ce type d'appareil pour laisser la place à un espace d'affichage de plus en plus grand. Les dispositifs mobiles intègrent maintenant de grands écrans tactiles où l'interaction multitouch est utilisée comme principale technique d'interaction en entrée. Évidemment, l'absence de clavier physique a profondément modifié les paradigmes de saisie de texte en fournissant des claviers logiciels qui s'affichent sur ces surfaces tactiles. Ceci a plusieurs avantages, dont notamment une disposition des touches plus facilement modifiable ; et une meilleure réactivité aux actions de l'utilisateur (système de prédiction, de déduction, complétion, etc.)

Cependant, ce type de dispositifs mobiles (i.e. avec écran tactile et sans clavier physique) cause des problèmes d'accessibilité. Sur un ordinateur de bureau classique, les utilisateurs en situation de déficience visuelle ont pour habitude d'utiliser presque exclusivement le clavier physique pour interagir avec leur environnement. Ils utilisent aussi des raccourcis clavier pour accéder aux diverses commandes d'une application. Sur les appareils mobiles avec écrans tactiles, le manque de repères tangibles (tels que la présence de touches physiques) rend la tâche très difficile, voire impossible. En effet, sur ces dispositifs, les utilisateurs accèdent aux différents éléments affichés à l'écran en interagissant directement sur l'écran. Les déficients visuels sont donc contraints d'explorer l'intégralité de l'environnement afin de trouver les éléments avec lesquels ils veulent interagir.

Dans le cas de la saisie de texte réalisée au moyen de solutions logicielles, ce problème est amplifié : ce processus d'exploration est requis pour chaque caractère à saisir. En effet, la méthode de saisie de texte la plus connue pour les déficients visuels repose sur l'exploration de l'écran avec le doigt. Au fur et à mesure du déplacement du doigt à l'écran, le système produit grâce à une synthèse vocale une vocalisation simultanée de la touche située sous le doigt (les exemples les plus courants sont VoiceOver d'Apple, ou TalkBack de Google). De ce fait, les utilisateurs malvoyants doivent constamment explorer l'écran pour localiser la touche souhaitée. Bien que le retour vocal soit améliorable avec des indices vibratoires, ce processus, appelé « exploration douloureuse » [3], est lent et nécessite beaucoup d'efforts de concentration de la part de l'utilisateur.

Pour pallier ce problème d'exploration, nous avons conçu et évalué un clavier logiciel « déductif » appelé DUCK (deDUCtive Keyboard) visant à diminuer cette exploration douloureuse. DUCK est basé sur le fait que la grande majorité des utilisateurs malvoyants connaissent la disposition spatiale des caractères de leur clavier physique, du fait de leur utilisation quotidienne. Avec DUCK, l'utilisateur doit explorer la disposition du clavier logiciel une seule fois par mot, au début, pour choisir la première

http://dx.doi.org/10.1145/2820619.2820638

lettre. Puis, il suffit de taper, pour chaque autre caractère du mot souhaité, à l'endroit où l'utilisateur estime que le caractère se trouve sur le clavier. À la fin de chaque mot, un système de déduction propose une liste de mots probables selon les frappes que vient d'effectuer l'utilisateur. Ainsi, ce clavier évite à l'utilisateur plusieurs phases d'exploration par mot en favorisant les frappes directes sur le clavier.

Dans cet article, nous faisons un survol des principales méthodes d'entrée existantes sur appareils mobiles, et notamment celles qui sont spécifiques aux utilisateurs ayant une déficience visuelle. Ensuite, nous décrivons les principes de DUCK ainsi que l'algorithme qui permet au clavier d'effectuer la déduction des mots. Nous présentons également une comparaison expérimentale de DUCK face à la méthode de saisie traditionnellement utilisée par les non-voyants. Enfin, nous discutons les résultats et les améliorations que nous imaginons pouvoir apporter, afin d'améliorer l'efficacité de DUCK et la satisfaction des utilisateurs.

ETAT DE L'ART

Saisie de texte sur dispositif mobile

Les claviers logiciels constituent les systèmes de saisie de texte les plus communs sur les appareils mobiles récents. Ils reproduisent généralement la disposition de caractères des claviers physiques la plus utilisée parmi celles de la langue de l'utilisateur (i.e. QWERTY ou AZERTY selon la langue). L'utilisateur doit appuyer sur l'écran à l'endroit où se situe la touche, et ce dans le but de sélectionner le caractère associé à cette touche. Toutefois, ces dispositions furent initialement conçues pour les machines à écrire afin de répondre à une entrée de texte à dix doigts, tout en isolant les caractères les plus fréquents. Par conséquent, ces dispositions ne sont pas adaptées pour une interaction avec un ou deux doigts comme c'est généralement le cas sur les appareils mobiles.

De plus, la présence d'un écran multitouch permet d'autres types d'interaction avec le clavier logiciel. Par exemple, Zhai et al. ont montré qu'il peut être plus rapide pour saisir un mot de tracer un chemin reliant tous les caractères de ce mot [19]. Pour les mots les plus fréquemment utilisés, le chemin peut même être remplacé par un geste qui est reconnu partout sur l'écran [6].

D'autre part, certaines dispositions de claviers ont été optimisées pour une frappe à un seul doigt. Ces dispositions ont été soit conçues à la main [8] ou ont impliqué des heuristiques d'optimisation [5], [15]. Ces heuristiques sont basés sur la fréquence des bigrammes dans une langue et ont pour but de rapprocher les caractères composant les bigrammes les plus fréquemment utilisés. Avec ce type de disposition, l'utilisateur a alors moins de mouvements à effectuer pour saisir du texte [18].

Dans les travaux mentionnés précédemment, les claviers virtuels contiennent autant de touches que de caractères. La taille des écrans de dispositifs mobiles étant encore assez petite, afficher un clavier complet sur ce type d'écran revient à avoir des touches réduites, ce qui oblige alors l'utilisateur à se concentrer sur la précision de ses frappes lors de la saisie de texte. Ainsi, afin de répondre au problème de la taille des touches, il est par exemple possible d'étendre une touche ou une partie du clavier. Cette modification dynamique peut s'appliquer sur une zone de l'écran en fonction du pointage de l'utilisateur [12], [13], [14] ou peut être faite après la saisie d'un caractère par l'utilisateur. Dans ce cas, la disposition des touches ou la taille de celles-ci sont modifiées sur la base de ce qui a été précédemment saisi [1], [10].

Une autre solution consiste à réduire le nombre de touches affichées sur l'écran et ainsi augmenter la taille de chaque touche. Plusieurs caractères sont alors regroupés sur une même touche. Pour chaque touche, l'utilisateur peut sélectionner un caractère avec une interaction spécifique. Le plus connu de ces claviers dits « ambigus » est celui utilisant la technique multifrappe. Cette technique consiste à taper une, deux ou trois fois de suite sur la même touche pour sélectionner le premier, le deuxième ou le troisième caractère affecté à cette touche [17]. Les claviers dits à « accords » sont une autre solution pour réduire le nombre de touches sur l'écran. Un caractère est entré lorsque toutes les touches requises sont pressées simultanément [4] ou successivement [7].

L'ensemble de ces différentes techniques permet d'accélérer la saisie de texte. Cependant, elles s'appuient sur une sélection visuelle des touches qui sont affichées sur l'écran. En l'absence d'indices physiques sur l'écran, il est nécessaire d'effectuer des adaptations afin qu'elles soient utilisables par des personnes en situation de déficience visuelle.

Solutions spécifiques pour déficients visuels

Le plus grand défi pour un utilisateur malvoyant lors de saisie de texte sur un appareil mobile est la localisation des différents caractères du clavier logiciel. La solution la plus courante consiste à vocaliser la touche située sous le doigt de l'utilisateur grâce à une synthèse vocale (voir par exemple VoiceOver d'Apple ou Google TalkBack). Les caractères sont entrés lorsque l'utilisateur soulève son doigt de l'écran. La nécessité pour l'utilisateur de parcourir l'ensemble du clavier avec son doigt afin de trouver le caractère attendu en constitue l'inconvénient majeur : même avec une bonne connaissance de la disposition spatiale des caractères, l'utilisateur doit vérifier la position du caractère pour chaque nouvelle saisie. Bien qu'elle soit fonctionnelle, cette solution est extrêmement consommatrice de temps. Vertanen et al. [16] ont proposé un système permettant de taper sur un clavier logiciel sans le regarder. Ce système de saisie par phrase permet de saisir du texte assez rapidement ($29,4\ wpm$), mais a l'inconvénient d'avoir un taux d'erreur très élevé ($18,5\ \%$).

Différents claviers virtuels ont été spécifiquement conçus pour les utilisateurs malvoyants. Ces claviers sont souvent basés sur l'alphabet Braille où les lettres sont codées dans une matrice à points composée de trois lignes et deux colonnes, chaque caractère correspondant à un code codé par un arrangement particulier de ces six points. BrailleType [11], par exemple, repose sur la division de l'écran en six cellules de même taille (trois lignes et deux colonnes), chacune correspondant à un point Braille. L'uti-

lisateur doit appuyer sur les cellules souhaitées pour activer les points nécessaires à la composition d'un caractère. Le système valide l'entrée après un certain délai. Le système TypeInBraille [9] est également basé sur le Braille. Dans ce cas, la sélection de caractères ne se base pas sur la division de l'écran, mais sur une interaction multifrappe. L'utilisateur doit effectuer trois multifrappes successives sur l'écran pour sélectionner un caractère. Les trois multifrappes correspondent aux trois lignes de la matrice Braille. Pour chaque ligne, l'utilisateur doit effectuer l'un des quatre gestes possibles : activer le point de gauche, activer le point de droite, activer les deux points ou ne pas activer de point pour la ligne.

Bien qu'une saisie de texte non-visuelle basée sur le Braille soit fonctionnelle, il est important de noter que « *moins de 10% des 1,3 millions de personnes considérées comme légalement aveugles aux États-Unis sont des lecteurs Braille*[1] ». Par conséquent, ces systèmes sont limités à une minorité des utilisateurs malvoyants. De plus, le temps nécessaire pour apprendre le Braille ou une nouvelle disposition du clavier pour être efficace avec ce nouveau système de saisie est extrêmement long [8]. La plupart des utilisateurs ne feront pas cet effort pour utiliser un dispositif spécifique.

A partir de ce constat, nous avons cherché à concevoir un clavier qui améliore la vitesse de saisie de texte pour les utilisateurs malvoyants, mais qui ne nécessite pas d'apprentissage particulier pour en avoir une utilisation optimale. Par conséquent, nous nous sommes appuyés sur l'agencement classique du clavier, connu par tous les utilisateurs malvoyants utilisant un clavier physique. Le défi était d'accélérer la vitesse de saisie de texte sans produire plus d'erreurs.

CONCEPTION DE DUCK

Principes de conception

Nous avons conçu le clavier DUCK sur la base de différentes observations : d'une part, l'utilisateur dispose d'une bonne connaissance de l'agencement des caractères de son clavier physique ; et d'autre part, il est plus efficace de fournir un retour vocal à l'utilisateur une fois que la saisie du mot est terminée plutôt que de l'informer après chaque caractère saisi.

DUCK est un clavier logiciel affiché sur l'ensemble de la surface de l'écran. Afin de ne pas perdre la connaissance préalable de l'agencement du clavier, la disposition des caractères alphabétiques est identique à celle du clavier physique de l'utilisateur.

Lorsqu'il souhaite saisir un mot, l'utilisateur fait d'abord glisser un doigt sur le clavier pour chercher la première lettre. Au cours de cette exploration, l'utilisateur a un retour vocal correspondant au caractère survolé. Pour valider la première lettre, il doit soulever son doigt de l'écran au moment où il se trouve sur le caractère souhaité. Pour terminer la saisie du mot dans sa totalité, l'utilisateur doit ensuite taper sur l'écran tactile autant de fois que ce mot comporte de lettres restantes. La seule instruction est de taper à l'endroit où il estime que le caractère souhaité se situe, et l'utilisateur n'a pas à se soucier de la précision de sa frappe. Ces lettres restantes n'ont pas de retour vocal pour ne pas perturber l'utilisateur lors de la saisie, à la place un simple retour audio (bip) est fourni pour chaque frappe. Une fois que l'utilisateur a fini de taper l'ensemble des lettres constituant son mot, il doit valider sa saisie en appuyant deux doigts sur l'écran tactile. Pour déterminer le mot que l'utilisateur a choisi, DUCK calcule les écarts entre les frappes théoriques nécessaires pour entrer chaque mot ayant la même initiale et le même nombre de lettres ; et renvoie les mots minimisants ces écarts. Les quatre mots les plus probables sont fournis dans une liste. Le premier mot est lu et épelé. L'utilisateur peut sélectionner le mot en cours par un tap à deux doigts, ou faire défiler la liste avec un geste à un doigt. En cas d'erreur, l'utilisateur peut annuler la saisie du mot en effectuant un geste avec deux doigts vers la gauche.

Avec cette technique nous estimons pouvoir réduire considérablement le temps d'exploration. En effet, l'utilisateur explore le clavier une seule fois par mot, lors de la saisie du premier caractère, puis effectue simplement des frappes pour les autres caractères du mot sans se soucier de la précision de ses frappes.

Algorithme déductif

Pour décrire l'algorithme, nous choisissons un cas où l'utilisateur veut taper un mot de N caractères commençant par la lettre L. Soit E le sous-ensemble de mots disponibles dans le dictionnaire, commençant par L et contenant N lettres. Notons $h = [h_i]$ la séquence de $N-1$ frappes de l'utilisateur, où $h_i = (x_i, y_i)$ sont les coordonnées de la i-ème frappe. Comme chaque mot m de E peut être décrit comme une séquence de caractères c_i, la touche correspondante au caractère et sa position sur l'écran sont connues. Soit $position(c_i) = (x'_i, y'_i)$ les coordonnées du centre de la touche correspondant au caractère marqué c_i. Pour calculer le meilleur mot possible pour la séquence h, l'algorithme calcule la distance entre la séquence de frappes h et chaque candidat de E :

$$D(h, m) = \sum_{i=1}^{N-1} \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2}$$

Enfin, DUCK renvoie les mots par ordre de probabilité, qui sont les mots ayant la même première lettre et qui minimisent la distance entre la séquence de hits h et le mot m. Le fichier du dictionnaire peut être modifié. Il s'agit d'un fichier décrivant les mots possibles, et la position des touches sur la disposition du clavier. Nous fournissons un outil pour créer des dictionnaires à partir d'une liste de mots et d'un agencement de caractères spécifique.

Interactions gestuelles

Lorsque l'utilisateur est dans la phase de validation du mot, une liste de mot lui est proposée. Pour faire défiler les mots de cette liste, il peut alors employer n'importe quel geste à un doigt. Nous avons fait ce choix pour éviter les erreurs de reconnaissance de gestes. La distinction entre défilement et validation se fait sur le nombre de doigts

[1] `https://nfb.org/Images/nfb/documents/word/The_Braille_Literacy_Crisis_In_America.doc`

utilisés. La validation du mot se fait au moyen de deux doigts.

En plus de saisir des mots, l'utilisateur peut aussi faire glisser un doigt vers le haut ou vers le bas pour sélectionner l'affichage des symboles ou des signes de ponctuation. Comme il n'y a pas de prédiction ou de déduction pour les symboles et les signes de ponctuation, l'interaction de DUCK bascule alors dans le mode de « recherche et validation » : l'utilisateur doit déplacer le doigt sur l'écran jusqu'à ce que le caractère soit trouvé. Relâcher le doigt valide l'entrée du caractère.

MATÉRIEL ET MÉTHODES

Dispositif

Nous avons réalisé une étude comparative entre DUCK et un clavier vocalisé nommé VODKA. VODKA reprend le principe du clavier logiciel le plus couramment utilisé sur dispositif mobile (VoiceOver d'Apple, ou TalkBack de Google) : chaque caractère est oralisé lorsque l'utilisateur passe son doigt dessus. La dernière touche choisie est validée lorsque l'utilisateur lève son doigt. Les deux claviers sont basés sur un agencement identique : cet agencement est un sous-ensemble d'AZERTY comprenant uniquement les caractères alphabétiques, disposés sur trois rangées de dix touches. Les quatre dernières touches de la troisième rangée ont été laissées vierges (là où les signes de ponctuation sont généralement situés). Les utilisateurs peuvent effacer la dernière lettre tapée en effectuant un geste avec deux doigts vers la gauche. Avec les deux claviers, il est possible d'entrer un espace en faisant glisser deux doigts vers la droite. Les espaces sont toutefois automatiquement ajoutés avec DUCK dès que le mot est validé.

Le corpus de mots utilisé dans cette expérience contient 336 531 mots extraits du dictionnaire Gutenberg français[2]. Ces mots sont utilisés par l'algorithme de déduction pour générer la liste de mots possibles. Nous avons choisi de ne proposer que les 4 mots les plus probables. Ce choix s'est fait suite à une pré-étude avec un utilisateur non-voyant à qui nous avons fait utiliser le système avec différentes tailles de liste de mots. Les listes supérieures à 4 mots n'apportaient pas de réel gain sur le taux d'apparition des mots dans la liste et l'utilisateur passait plus de temps à rechercher dans la liste. C'est pourquoi nous avons choisi de ne proposer que 4 mots dans la liste de déduction.

L'expérience a été menée sur un Samsung Galaxy S III, en utilisant une version stock d'Android 4.1, un quadruple cœur de 1,4 GHz Cortex-A9 pour CPU et 1 Go de RAM. L'écran tactile a une taille de 59,8 $\times$ 106,2 mm avec une résolution de 720 $\times$ 1280 pixels. Les deux claviers sont codés en Java pour une utilisation directe sur le smartphone. La synthèse vocale est proposée par le moteur IVONA, que nous avons sélectionné pour sa clarté et sa prosodie proche d'une voix naturelle.

Participants

Nous avons recruté douze participants, six hommes et six femmes (âge moyen : 32,4 ans, écart-type : 16,2 années). Dix d'entre eux étaient considérés comme aveugles sans aucune perception. Deux utilisateurs avaient une perception lumineuse. Aucun d'entre eux n'avait de capacités visuelles suffisantes pour percevoir la forme générale du clavier virtuel. Tous les participants ont déclaré avoir une bonne connaissance de la disposition de caractères AZERTY : huit utilisateurs en avaient une utilisation quotidienne personnelle ; tandis que quatre donnaient des cours de dactylographie, bénéficiant de ce fait d'une connaissance dite experte.

L'expérience a été menée en conformité avec les principes de la déclaration d'Helsinki, et de l'acte français de protection des données. Chaque utilisateur a signé un formulaire de consentement écrit.

Procédure

La tâche mesurée consistait à taper une phrase courte après dictée par la synthèse vocale, en utilisant soit DUCK, soit VODKA. Les sujets devaient taper cette phrase aussi vite que possible tout en minimisant le nombre d'erreurs. La moitié des utilisateurs a commencé avec une session utilisant DUCK, tandis que les autres ont commencé avec VODKA. Au sein de chaque session, l'utilisateur devait saisir huit phrases courtes sous la dictée. Chaque participant avait seize phrases identiques, mais attribuées dans un ordre pseudo-aléatoire différent. Les seize phrases étaient simples afin de faciliter leur mémorisation, mais également pour minimiser le nombre de fautes de frappe. Les phrases comprenaient quatre à sept mots parmi les mots les plus fréquents de la langue française[3] ; tout en s'assurant que les lettres de ces phrases correspondaient à la répartition de la fréquence des lettres de la langue française. Chaque séance débutait par une brève phase de familiarisation pour le clavier étudié. Les sujets devaient apprendre à saisir des lettres, des mots courts, des mots longs et enfin des phrases. A tout moment, l'utilisateur pouvait réécouter la phrase à saisir en faisant glisser deux doigts vers le haut. Il pouvait écouter sa saisie réelle en faisant glisser deux doigts vers le bas, et effacer sa dernière entrée (mot pour DUCK, lettre pour VODKA) en faisant glisser deux doigts vers la gauche. À la fin de chaque session, l'utilisateur devait compléter un questionnaire SUS sur le clavier. En outre, une fois l'expérience terminée, ce dernier devait mentionner sa préférence générale et ses opinions sur les deux claviers.

Toutes les interactions, y compris les événements utilisateur (mouvements des doigts, appuis, relâchements), la saisie de texte (caractères et mots tapés) et les listes de déductions ont été enregistrées dans un fichier XML. Tous les questionnaires ont été passés oralement.

RÉSULTATS

Pour chaque sujet, l'expérience entière a duré entre 50 à 165 minutes, comprenant une pause de 15 minutes entre les deux sessions. Nous avons enregistré les saisies textuelles de douze utilisateurs différents tapant seize phrases

[2] `http://www.pallier.org/ressources/dicofr/dicofr.html`

[3] `http://netia59a.ac-lille.fr/va.anzin/IMG/pdf/mots_les_plus_frequents.pdf`

de quatre à sept mots (pour un total de 192 phrases et 1032 mots). Dans les sections suivantes, le niveau de signification (α) a été fixé à 0,05.

Les sections suivantes présentent les résultats quantitatifs et qualitatifs, ainsi que des statistiques déductives. Les corrections de Bonferroni ont été systématiquement appliqués pour les comparaisons multiples.

Vitesse de frappe pour les mots et les phrases

Afin de comparer l'efficacité des utilisateurs lors de l'utilisation des deux claviers, nous avons d'abord sélectionné les phrases et les mots qui ont été tapés correctement. L'analyse sur les erreurs de frappe est détaillée dans la section suivante. La vitesse d'entrée a été calculée de deux façons. Tout d'abord, nous avons calculé la vitesse de saisie de texte par phrase. Ici, la longueur de chaque phrase (y compris les espaces entre les mots et à la fin d'une phrase) a été divisé par le temps (en secondes) nécessaire pour entrer dans cette phrase. Cette vitesse est donnée en caractères par seconde (CPS).

Les vitesses moyennes d'entrée des phrases étaient $0,37$ cps ($\sigma = 0,12$) pour VODKA et $0,38$ cps ($\sigma = 0,15$) pour DUCK, la médiane étant de $0,33$ cps pour les deux. Comme les distributions de vitesse ne sont pas normales (Shapiro-Wilk, $W = 0,95$, $p < 0,01$), nous avons utilisé un test de Wilcoxon pour les comparer. Ce test a montré que les vitesses ne sont pas statistiquement différentes ($W = 2,115$, $p = 0,819$) entre les deux claviers.

Nous avons également comparé la vitesse de saisie d'un mot entre les deux claviers. La vitesse de saisie d'un mot correspond à la durée entre le moment l'entrée du premier caractère d'un mot et celui de la sélection finale de l'ensemble du mot (en appuyant sur la barre d'espace pour VODKA ou en appuyant deux doigts sur l'écran pour DUCK), divisée par la longueur de ce mot. Il est important de noter ici que le temps de saisie pour DUCK comprend la phase de validation, qui consiste à écouter les différents mots de la liste et à sélectionner celui attendu. Les vitesses moyennes d'entrée de mots étaient $0,383$ cps ($médiane = 0,28$, $\sigma = 0,20$) pour VODKA et $0,398$ cps ($médiane = 0,32$, $\sigma = 0,18$) pour DUCK. Un test de Wilcoxon a montré que ces vitesses sont significativement différentes ($W = 774$, $p = 0,048$).

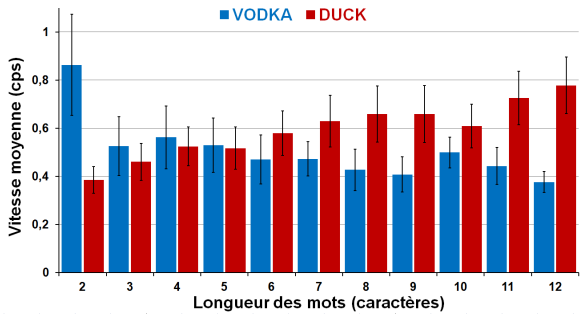

Figure 1. Vitesse de frappe moyenne (en cps) en fonction de la longueur de mot - la vitesse de frappe pour VODKA diminue au fil du temps tandis qu'elle augmente pour DUCK.

La figure1 décrit la vitesse de saisie moyenne d'un mot en fonction de la longueur de ce mot. Il semble que VODKA est très efficace pour saisir un mot de deux lettres. Par contre, le temps nécessaire pour saisir un caractère augmente avec la longueur des mots. La vitesse pour saisir un mot de 12 lettres est seulement de $0,4$ cps. Cependant, DUCK est de plus en plus rapide. Les deux droites de régression s'interceptent autour de quatre caractères. Nous avons comparé les vitesses pour chaque longueur de mot en utilisant une série de tests de Wilcoxon avec une correction de Bonferroni. Il semble que DUCK soit systématiquement plus efficace que VODKA pour les mots de plus de six caractères. Plus précisément, le graphe de la figure 1 montre que la vitesse pour taper des mots de douze lettres avec DUCK est proche de $0,8$ cps, soit deux fois la vitesse de VODKA.

Saisie de mots sans validation

Comme mentionné précédemment, le temps nécessaire pour taper un mot avec DUCK comprend deux phases. La première phase, que nous avons appelée « frappe », comprend le temps nécessaire pour trouver et sélectionner la première lettre d'un mot de N caractères, et le temps nécessaire pour faire les $N - 1$ frappes suivantes. La deuxième phase, que nous avons appelé « validation », comprend le temps nécessaire pour choisir le mot correct dans la liste. La position du mot désiré dans la liste dépend de la distance entre les frappes de l'utilisateur et la distance avec le mot désiré. De ce fait, dans les sections suivantes, nous avons évalué le temps de saisie avec VODKA (T_V) ; et le temps de saisie avec DUCK, séparé en deux temps, le temps de frappe (T_{TAP}) et le temps de validation ($T_V AL$).

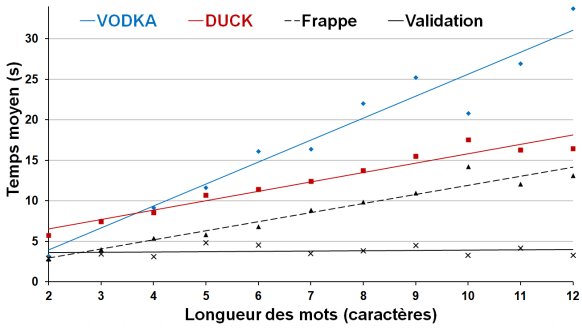

Figure 2. Temps moyens (en s) en fonction des longueurs de mots. Les temps de frappe (T_{TAP}) et de validation (T_{VAL}) sont également indiqués ($T_D = T_{TAP} + T_{VAL}$).

La figure 2 indique le temps nécessaire pour saisir un mot avec VODKA (T_V) et DUCK (T_D). Elle montre aussi le temps nécessaire pour la frappe (T_{TAP}) et la validation (T_{VAL}) pour DUCK. Ces différentes durées ont été calculées en fonction de la longueur de mot (2 à 12 caractères). La régression linéaire des moindres carrés montre que le temps de saisie avec VODKA (ligne bleue) augmente avec un coefficient de 2,7 ($R^2 = 0,95$). Le temps de saisie de DUCK (ligne rouge) est un peu moins affecté par la longueur des mots ($y = 1,2$, $R^2 = 0,95$). Le temps de frappe (ligne pointillée) est inférieur, mais fortement corrélé à l'ensemble du temps de saisie avec un coefficient très similaire ($y = 1,1$, $R^2 = 0,94$). Comme prévu, le temps de validation (ligne noire) ne dépend pas de la longueur de mot ($y = 0,03$, $R^2 = 0,03$). Il est presque équivalent à

l'ordonnée à l'origine ($a = 3,5$) et constant, indépendamment de la longueur des mots, ce qui montre que le temps de saisie avec DUCK ne dépend que du temps de frappe.

Il est intéressant de noter que les droites de régression pour les temps de saisie de DUCK et VODKA se croisent à 3,7 lettres, ce qui signifie que les mots de plus de quatre lettres sont plus rapides à saisir avec DUCK. Il semble qu'il y ait une différence significative (les tests de Wilcoxon avec une correction de Bonferroni donnent $W = 1402$, $p = 0,02 < \alpha$) entre le temps de frappe de DUCK et le temps de saisie de VODKA. De même, la différence de temps nécessaire pour taper de longs mots avec les deux claviers augmente. Par exemple, il est deux fois et demie plus long de taper des mots de dix lettres avec VODKA qu'avec DUCK ($13,1$ s contre $33,7$ s, $p = 0,009$). En utilisant seulement deux catégories de mots qui sont soit plus courts ou plus longs que quatre lettres, il y a une différence de temps très significative (Wilcoxon, $p = 0,0002$). Évidemment, le temps total de saisie (comprenant la phase de validation obligatoire) avec DUCK minimise cette observation. Cependant, diviser les données à partir des mots de six lettres donne un résultat significatif ($p < 0,001$), montrant que l'étape de validation nécessaire avec DUCK est entièrement compensée pour les mots de plus de cinq caractères.

Phase de validation

Comme mentionné précédemment, la phase de validation est égale à $3,5$ s, et n'a pas été affectée par la longueur de mot. Toutefois, cela dépend de la position du mot attendu dans la liste (test de Wilcoxon, $W = 954$, $p = 0,024$). Il faut en moyenne $3,0$ s pour sélectionner un mot en première position, et ce temps augmente lorsque le mot est situé plus loin dans la liste (respectivement $6,6$ s, $8,2$ s et $9,6$ s pour la deuxième, troisième et quatrième positions).

Pendant toute l'expérience, pour tous les sujets, le mot attendu était en première position dans 83 % des cas. Les mots en deuxième, troisième et quatrième sont respectivement choisis dans 8 %, 5 % et 3 % des cas. Indépendamment de la longueur du mot, le mot sélectionné est habituellement dans les deux premières positions.

Ceci peut s'expliquer en partie par la bonne connaissance que les utilisateurs ont de leur clavier. En effet, pour les mots sélectionnés et se trouvant en première position de la liste, les frappes de l'utilisateur sur l'écran sont en moyenne à 95 px du centre de la touche associée au caractère souhaité (159 px pour les mots en deuxième position, 165 px pour les mots en troisième position et 152 px pour les mots en dernière position). Chaque touche mesurant 113 px de long et 214 px de haut, cela veut dire que les frappes de l'utilisateur sont soit sur la bonne touche, soit dans un périmètre très proche de la touche voulue.

Niveau de correction de la saisie de texte

Nous avons considéré qu'une phrase est correcte lorsque l'utilisateur n'a pas fait la moindre erreur dans la phrase. Les utilisateurs n'avaient pas de contraintes en termes de position du téléphone ou d'utilisation de doigts spécifiques. Ils pouvaient ou non corriger leurs erreurs avant de valider leur entrée. En regardant les données dans leur globalité, en tenant compte des douze utilisateurs (soit 192 phrases), 156 additions et 62 omissions de lettres avaient été commises avec VODKA ; contre 218 additions et 118 omissions pour DUCK. Cependant, nous avons constaté que parmi les 192 phrases saisies par les utilisateurs, 69 avaient été correctement tapées avec DUCK, tandis que ce nombre chutait à 31 pour VODKA. Nous avons regardé en détail ces phrases tapées correctement.

La métrique habituelle, la distance minimale interchaîne (Minimum String Distance ou MSD), est définie par le nombre minimum de modifications uniques de caractères (comprenant les ajouts, les omissions et les substitutions) pour transformer une chaîne en une autre. En utilisant DUCK et VODKA, il y a peu de différences au niveau MSD (96 % pour VODKA et 94 % pour DUCK). Les deux claviers offrent une tolérance similaire pour des erreurs au niveau de la lettre. Comme la MSD classique ne serait pas applicable sur ces résultats, nous avons conçu une MSD légèrement modifiée. Pour chaque phrase, nous avons compté le nombre total de mots qui ont été saisis, et les mots qui ont été tapés correctement. Nous avons ensuite défini la distance minimale intermots (Minimum Word Distance ou MWD) qui représente la distance entre deux phrases en mots :

$$MWD = \frac{W_{correct}}{W_{total}}$$

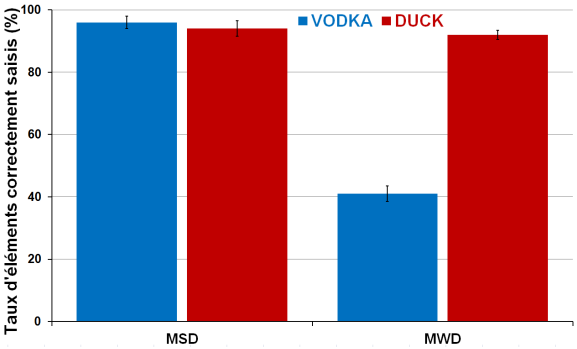

Figure 3. MSD et MWD pour les deux claviers. Bien que les deux claviers offrent une MSD similaire, DUCK est plus efficace que VODKA pour la saisie de mots (MWD).

La comparaison des MWD (Wilcoxon ; $W = 486$, $p < 0,001$) a montré que DUCK est nettement supérieur à VODKA ($MWD = 41$ % pour VODKA contre 92 % pour DUCK) en termes de correction des mots (voir figure 3). Ce résultat montre que DUCK permet une réduction impressionnante des erreurs de frappe au niveau des mots.

Préférences des utilisateurs

Parmi les retours utilisateurs, sept utilisateurs préféraient VODKA tandis que cinq préféraient DUCK. Les utilisateurs ont expliqué leur préférence à VODKA en indiquant que certains des mots utilisés dans l'expérience étaient plus facile à taper avec ce type de clavier. Pour certains, il est plus facile de s'appuyer sur la disposition du clavier et de l'écran physique pour taper des mots (« *Quand je veux saisir le 'p', par exemple, je dois seulement glisser*

mon doigt sur le coin en haut à droite jusqu'à ce que je le trouve. »). Pour d'autres, ce clavier est plus adapté à un usage quotidien, notamment à l'utilisation de mots abrégés des SMS (« *Je ne tape pas des mots comme ça quand j'envoie des SMS, ils sont écrits en abrégé.* »). La dernière raison, et probablement la plus compréhensible, porte sur le sentiment de confiance et de contrôle (« *Je trouve ça effrayant de ne pas savoir ce qui a été tapé, même si le mot est correctement deviné. Tout ce que j'entends, ce sont les bips lorsque je tape.* »). Avec VODKA, chaque lettre est vocalisée et il n'y a pas besoin d'une étape de validation finale. En outre, certains utilisateurs ont l'habitude de ce type de clavier, car il est similaire à VoiceOver sur iPhone (« *Je suis habitué à une saisie de cette façon : glisser et relâcher pour sélectionner une lettre. Il y a aussi un mode où on appuie deux fois sur les lettres, mais c'est plus lent.* »). D'autre part, les utilisateurs préférant DUCK ont vraiment apprécié la vitesse de frappe. Très vite, ils ont remarqué qu'il était plus rapide que VODKA : si l'utilisateur possède une connaissance suffisante de la disposition du clavier à l'esprit, taper avec DUCK revient à taper sur un clavier normal (« *Je peux taper sur ce clavier aussi vite que je le fais sur mon PC.* »).

Enfin, nous avons examiné les questionnaires SUS [2] pour les deux claviers. VODKA a obtenu un score de 58,1 ; tandis DUCK a reçu un bien meilleur score de 70,5.

DISCUSSION

A l'analyse des résultats, il apparaît que les claviers DUCK et VODKA se valent sur la saisie de l'ensemble des phrases. Cependant, si nous décomposons le temps de saisie pour le clavier DUCK en temps de frappe puis temps de validation, il apparaît que le temps nécessaire pour frapper sur les différents caractères avec DUCK est bien plus faible que le temps nécessaire pour saisir un mot avec VODKA. En revanche, la saisie d'un mot avec DUCK nécessite en plus un temps de validation difficilement compressible (en moyenne $3,5\ s$). Pour que la saisie avec le clavier DUCK soit intéressante, il faut donc que le gain lors de la phase de frappe soit plus conséquent que ce temps de validation. Le gain lors de la phase de frappe est proportionnel au nombre de caractères dans le mot. Il compense le temps de validation à partir des mots constitués d'au moins 4 caractères. Par conséquent, le clavier DUCK n'est donc intéressant que pour les mots d'au moins 4 caractères. Pour les mots de 1, 2 ou 3 caractères, l'interaction proposée ne convient donc pas car l'utilisateur perd trop de temps, lors de la phase de validation, à choisir le mot qu'il souhaite dans la liste des mots proposés. Ceci n'est pas dû à un problème de performance du système de déduction car même pour les mots courts, le mot souhaité par l'utilisateur est généralement positionné à la première place dans la liste. Dans ce cas, l'utilisateur n'a pas à chercher le mot dans la liste, il doit simplement le valider.

Dans tous les cas, la position du mot dans la liste a son importance. En effet, plus le mot se trouvera à une position éloignée dans la liste et plus l'utilisateur mettra du temps à valider le mot. Bien que l'algorithme de déduction de mots utilisé se montre performant, certains problèmes de déduction subsistent. Le premier d'entre eux est la proximité géographique de deux mots : en effet, si deux mots de même longueur ont des caractères qui sont à la même position dans le mot et qu'ils sont situés à proximité l'un de l'autre sur le clavier, alors le système de déduction aura du mal à trancher entre les deux mots. Par exemple, les mots « nuit » et « nuis » ne diffèrent que de leur quatrième lettre. Ces deux lettres étant assez proches sur le clavier, le système de déduction ne saura pas faire la différence entre les deux mots si l'utilisateur venait à effectuer sa quatrième frappe à équidistance entre la touche contenant le caractère « s » et celle contenant le caractère « t ».

Enfin, la faible tolérance aux erreurs constitue le dernier problème de notre système de déduction. En effet, l'algorithme étant basé sur le nombre de frappe de l'utilisateur, si ce dernier a ajouté ou oublié un caractère lors de ses frappes, alors le système ne lui proposera pas le mot souhaité car le nombre de frappes effectuées par l'utilisateur ne correspond pas au nombre de caractères du mot souhaité. Par conséquent, comme le système ne permet pas la correction des frappes au cours de la saisie d'un mot, l'utilisateur devra alors ressaisir complètement le mot. Ce problème (ressaisir complètement le mot) est valable à chaque fois que le mot souhaité ne se trouve pas dans la liste des mots proposés. Ceci a alors une forte incidence sur la vitesse de saisie de texte si ce phénomène se reproduit plusieurs fois.

TRAVAUX FUTURS

Cette première expérimentation a mis en évidence un certain nombre de faiblesses que nous nous attacherons à combler dans nos futurs travaux.

Les mots courts

Il apparaît que les mots courts ne peuvent être saisis aussi rapidement avec DUCK qu'avec VODKA du fait de la phase de validation. Pour résoudre ce problème, nous envisageons de mettre en place une interaction à base de gestes pour sélectionner les mots les plus courants de moins de 4 caractères. Le challenge sera de proposer une interaction qui permettra de distinguer la recherche d'un caractère sur le clavier et la réalisation d'un geste effectué pour saisir un mot court. Seuls les mots courts les plus fréquemment utilisés seront représentés par un geste pour éviter une trop forte charge d'apprentissage à l'utilisateur. Ceci devrait permettre de gagner du temps sur les mots que l'utilisateur utilise le plus souvent.

Le système de déduction

L'objectif de cet article est de montrer la pertinence des interactions proposées pour saisir du texte par déduction de mots. Pour ce faire, nous avons utilisé l'algorithme de déduction le plus simple qui existe. Pour améliorer les performances de déduction de notre système, il serait maintenant intéressant d'utiliser des algorithmes de déduction plus complexes, tel que celui utilisé dans Shapewriter [19], qui prend en compte la forme du tracé pour déduire le mot souhaité. Dans notre cas, l'utilisateur n'effectue pas de tracé sur l'écran, mais il serait possible de construire celui-ci à partir des différents taps effectués par l'utilisateur.

D'autre part, de manière à optimiser la déduction tout au long de la saisie de l'utilisateur, nous utiliserons un système de renforcement : dans ce type de système, chaque mot à une probabilité d'apparition dans le dictionnaire utilisé par le système de déduction. Chaque mot saisi par l'utilisateur vient augmenté la probabilité de ce mot dans le dictionnaire. Ainsi, plus un mot est utilisé par un utilisateur, plus il a de chance de ressortir dans les premières places de la liste de déduction, et ce, même si l'utilisateur le saisit avec un peu moins de précision. Ce système de renforcement permet aussi d'ajouter des mots au dictionnaire. Ainsi l'utilisateur pourra ajouter les mots hors dictionnaire qu'il a l'habitude d'utiliser régulièrement (comme par exemple des noms propres).

Les mots hors dictionnaire

Notre système de déduction se base sur un corpus de mots pour proposer un ensemble de mots à l'utilisateur. Donc si le mot que souhaite saisir l'utilisateur n'appartient pas au corpus de mots, le mot ne sera alors jamais proposé à l'utilisateur. Pour faire face à ce problème, il est donc nécessaire de proposer une interaction permettant à l'utilisateur de pouvoir faire de la saisie caractère par caractère. L'interaction envisagée consiste à rechercher le caractère de la même manière que l'utilisateur cherche actuellement le premier caractère du mot (c'est-à-dire par exploration du clavier avec un doigt posé sur l'écran), puis pour le valider sans passer par la phase de déduction, l'utilisateur effectuera un double tap sur le caractère souhaité pour le valider directement. Ainsi, en cas de problème répété avec le système de déduction, l'utilisateur aura toujours la possibilité de saisir le mot voulu avec la saisie caractère par caractère.

REMERCIEMENTS

Nous tenons à remercier les utilisateurs qui ont participé à cette étude pour leur temps et leur intérêt. Nous remercions également Claude GRIET, l'IJA et LACII pour leur collaboration et leur disponibilité.

BIBLIOGRAPHIE

1. Aulagner G., François R., Martin B., Michel D. & Raynal M. Floodkey: increasing software keyboard keys by reducing needless ones without occultation. In *Proceedings of the 10th WSEAS international conference on Applied computer science*, World Scientific and Engineering Academy and Society (WSEAS) (2010), 412–417.
2. Bangor A., Kortum P. T. & Miller J. T. An empirical evaluation of the system usability scale. *International Journal of Human-Computer Interaction 24*, 6 (2008), 574–594.
3. Bonner M. N., Brudvik J. T., Abowd G. D. & Edwards W. K. No-look notes: accessible eyes-free multi-touch text entry. In *Pervasive Computing*. Springer, 2010, 409–426.
4. Cho H. & Kim C. Bubstack: A self-revealing chorded keyboard on touch screens to type for remote wall displays. In Proc. *AH '14*, ACM (2014), 23:1–23:2.
5. Dunlop M. & Levine J. Multidimensional pareto optimization of touchscreen keyboards for speed, familiarity and improved spell checking. In Proc. *CHI '12*, ACM (2012), 2669–2678.
6. Kristensson P.-O. & Zhai S. Shark2: A large vocabulary shorthand writing system for pen-based computers. In Proc. *UIST '04*, ACM (2004), 43–52.
7. MacKenzie I. S., Soukoreff R. W. & Helga J. 1 thumb, 4 buttons, 20 words per minute: Design and evaluation of h4-writer. In Proc. *UIST '11*, ACM (2011), 471–480.
8. MacKenzie I. S. & Zhang S. X. The design and evaluation of a high-performance soft keyboard. In Proc. *CHI '99*, ACM (1999), 25–31.
9. Mascetti S., Bernareggi C. & Belotti M. Typeinbraille: Quick eyes-free typing on smartphones. In Proc. *ICCHP'12*, Springer-Verlag (2012), 615–622.
10. Merlin B. & Raynal M. Evaluation of spreadkey system with motor impaired users. In *Computers Helping People with Special Needs*. Springer, 2010, 112–119.
11. Oliveira J. a., Guerreiro T., Nicolau H., Jorge J. & Gonçalves D. Brailletype: Unleashing braille over touch screen mobile phones. In Proc. *INTERACT'11*, Springer-Verlag (2011), 100–107.
12. Oney S., Harrison C., Ogan A. & Wiese J. Zoomboard: A diminutive qwerty soft keyboard using iterative zooming for ultra-small devices. In Proc. *CHI '13*, ACM (2013), 2799–2802.
13. Pollmann F., Wenig D. & Malaka R. Hoverzoom: Making on-screen keyboards more accessible. In Proc. *CHI EA '14*, ACM (2014), 1261–1266.
14. Raynal M. & Truillet P. Fisheye keyboard: whole keyboard displayed on pda. In *Human-Computer Interaction. Interaction Platforms and Techniques*. Springer, 2007, 452–459.
15. Raynal M. & Vigouroux N. Genetic algorithm to generate optimized soft keyboard. In Proc. *CHI EA '05*, ACM (2005), 1729–1732.
16. Vertanen K., Memmi H. & Kristensson P. O. The feasibility of eyes-free touchscreen keyboard typing. In Proc. *ASSETS '13*, ACM (2013), 69:1–69:2.
17. Vigouroux N., Vella F., Truillet P. & Raynal M. Evaluation of aac for text input by two groups of subjects: able-bodied subjects and disabled motor subjects. In *8th ERCIM Workshop "User Interfaces for All", Vienne, Autriche* (2004).
18. Zhai S., Hunter M. & Smith B. A. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In Proc. *UIST '00*, ACM (2000), 119–128.
19. Zhai S., Kristensson P. O., Gong P., Greiner M., Peng S. A., Liu L. M. & Dunnigan A. Shapewriter on the iphone: From the laboratory to the real world. In Proc. *CHI EA '09*, ACM (2009), 2667–2670.