THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Discipline ou spécialité :

INFORMATIQUE - Interaction Homme-Machine

Présentée et soutenue par :
Georges BADR

le : lundi 20 juin 2011

Titre :

Modèle théorique et outil de simulation pour une meilleure évaluation des claviers logiciels augmentés d'un système de prédiction de mots

---

Ecole doctorale :
Mathématiques Informatique Télécommunications (MITT)

Unité de recherche :
IRIT - UMR 5505

Directeur(s) de Thèse :

Pr. Philippe PALANQUE, M. Mathieu RAYNAL

Rapporteurs :

Pr. Franck POIRIER, M. Benoît MARTIN

Autre(s) membre(s) du jury

Pr. Patrick GIRARD, Pr. Xavier DE BOISSEZON

Remerciements

Je tiens à exprimer ma profonde gratitude envers M. Mathieu RAYNAL pour son soutien et ses encouragements inappréciables. Qu'il soit remercié pour ses enseignements rigoureux, ses conseils lucides et pertinents qui m'ont stimulé à poursuivre avec persévérance le chemin initiatique de la thèse. Pour son aide et ses directives qui furent déterminantes dans la rédaction et la réalisation de ce projet. Je le remercie également pour sa gentillesse et sa disponibilité, sans lui, ce travail n'aurait jamais vu le jour.

J'exprime ma reconnaissance à M. Frank POIRIER, M. Benoît MARTIN, M. Patrick Gérard, M. DE BOISSEZON, M. Philippe PALANAQUE membres du jury qui ont bien voulu prendre connaissance de ces recherches et me faire part de leurs remarques.

Je remercie aussi les membres de l'équipe, sans oublier personne pour le temps qu'ils ont porté sur les expérimentations parfois longues et fatigantes.

Mes remerciements vont également à Caritas et Sésobel qui m'ont accueilli chaleureusement et m'ont permis de faire quelques expérimentations avec les personnes en situation d'handicap chez eux.

Je suis très sensible à l'inestimable soutien que m'ont prodigué mes parents durant de longues années. Qu'ils soient chaleureusement remerciés pour leur patience, leur amour et leur soutien financier !

Enfin, je souhaite témoigner ma gratitude la plus vive et la plus affectueuse envers, tous celles et ceux qui me sont proches et chers et qui m'ont aidé à arriver au sommet !

Georges BADR

Table des matières

Liste des figures

Liste des tableaux

Introduction

De nos jours, la saisie de données numériques est devenue une activité extrêmement importante au quotidien. Au-delà de la rédaction de document dans le cadre des activités de bureautique, la saisie numérique s'est aussi considérablement accrue avec l'émergence de la communication Homme-Homme médiée. Dans ce contexte, la communication orale, auparavant privilégiée, laisse de plus en plus place à la communication écrite avec l'envoi de messages courts par téléphone (Short Message Service, SMS ou Multimedia Messaging Service, MMS), les mails ou encore les communications synchrones via les messageries instantanées.

Cette modificication des usages de la saisie de texte est aussi accompagnée d'une évolution importante des dispositifs informatiques. L'ordinateur de bureau est de plus en plus souvent délaissé pour l'utilisation de dispositifs plus mobiles (téléphone portable, tablette tactile, etc.). Cependant, pour des raisons de portabilité et de mobilité, ces dispositifs informatiques sont de plus en plus réduits et ne sont généralement plus équipés de claviers. D'où le besoin de disposer de méthodes alternatives de saisie de texte pour se substituer au clavier standard.

Le clavier logiciel s'est présenté comme une solution intuitive et naturelle pour remplacer le clavier physique. L'utilisateur interagit avec le système à l'aide d'un dispositif de pointage comme une souris, un stylet ou même le doigt. Mais les expériences ont montré que ces claviers logiciels étaient moins performants que leurs homologues physiques (45 mots par minute pour le clavier standard de type QWERTY [MacKenzie 1999b], 20 mots par minute pour le clavier logiciel avec la même disposition des caractères [MacKenzie 1999a] et 5 mots par minute pour une personne en situation d'handicap saisissant du texte avec un système d'aide à la communication [Le Pévédic 1997]).

Cette faible performance peut être expliquée par la disposition même des touches. En effet, Sholes [Land 1981] avait conçu cette disposition de caractères QWERTY en 1878 pour

éviter que les marteaux de la machine à écrire ne s'entrecroisent entre eux quand les utilisateurs écrivaient trop rapidement. Les touches ont alors été réparties de façon à ce que les caractères qui ont le plus de chance de se succéder dans la langue anglaise[1] soient éloignés. Avec l'utilisation des ordinateurs, cette contrainte mécanique a disparu. Cependant, cette disposition des touches est restée inchangée. Si elle ne pose pas de problèmes sur les claviers physiques du fait de la possible utilisation des dix doigts pour saisir, cette disposition pose problème sur les claviers logiciels. En effet, sur ces derniers, la saisie se fait au moyen d'un dispositif de pointage et donc génralement d'un pointeur unique (il peut y en avoir plusieurs si l'on considère la saisie sur écran tactile multitouch [SPB]). Le temps consacré au déplacement s'en retrouve décuplé et la saisie d'une séquence de caractères extrêmement lente comparée à l'équivalent sur un clavier standard (par exemple la distance entre les deux caractères du *'m'* au *'a'* pour le pronom possessif 'ma'). Outre la lenteur des déplacements, ce problème peut également engendrer un surcroît de fatigue aux personnes à faible motricité des membres supérieurs et qui ne disposent que de ce moyen de saisie [Bérard 2004b] [Vella 2005].

De ce constat est née la nécessité de concevoir des systèmes d'aide à la communication écrite faisant intervenir le Traitement Automatique des Langues (TAL) et l'Interaction Homme-Machine (IHM). Ces systèmes dits de communication assistée ont pour but d'augmenter la performance des claviers logiciels : c'est-à-dire accélérer la vitesse de saisie tout en limitant les erreurs de saisie. Plusieurs voies de recherches ont été explorées pour coupler ces claviers à des systèmes de prédiction, que ce soit une prédiction de mots ou de lettres. Cependant, même si ces systèmes dynamiques permettent de réduire le nombre d'actions à réaliser pour saisir du texte, mais les performances de saisie de l'utilisateur ne sont pas toujours améliorées et peuvent même se trouver diminuées.

Un des exemples les plus connus et répandus de système d'aide à la communication est la liste de prédiction. Cette liste vient en complément du système de saisie utilisé, et

[1] A noter que le clavier AZERTY est une adaptation du clavier QWERTY à la langue française.

propose une liste de mots qui pourraient compléter la saisie initiée par l'utilisateur. L'utilisation de cette liste de prédiction permet souvent de diminuer de manière significative le nombre d'actions à réaliser pour saisir un mot. Un utilisateur, selon le système de prédiction utilisé, peut espérer réduire de 20 et 50 % le nombre d'actions à réaliser. En revanche, sa vitesse de saisie n'est pas forcément autant améliorée.

L'objet de cette thèse était donc de comprendre où l'utilisateur venait à perdre du temps lorsqu'il saisit du texte au moyen d'un système d'aide à la communication. Nous nous sommes focalisés sur l'étude du fonctionnement de la liste de prédiction, et ce dans un contexte d'utilisation où la personne a une déficience motrice des membres supérieurs qui l'empeche d'utiliser directement ses doigts pour interagir un dispositif d'entrée (clavier physique, écran tactile multipoint, etc.). Nos utilisateurs étaient donc contraints à l'utilisation d'un dispositif de pointage mono-pointeur.

A partir des constats que nous avons pu tirer de notre étude, nous proposons alors des améliorations de ce type de liste pour la rendre plus efficace.

Les travaux effectués lors de cette thèse sont présentés dans ce mémoire sous une structuration en cinq chapitres :

- Le chapitre 1 présentera les différents systèmes de prédiction existants et les types d'interface proposés pour en présenter les résultats à l'utilisateur ;
- Dans le chapitre 2, nous étudierons comment l'utilisateur interagit avec un clavier logiciel couplé à une liste de prédiction. Pour cela, nous nous baserons sur les interactions entre l'utilisateur et le système, mais aussi sur une analyse du suivi du regard de l'utilisateur durant sa saisie au moyen du clavier logiciel.
- Le chapitre 3 pose les bases du modèle théorique que nous proposons pour prédire la vitesse de saisie moyenne avec un clavier logiciel couplé à une liste de prédiction. Nous présenterons aussi un outil de simulation utilisant notre modèle, ce qui permettra ainsi à l'utilisateur de facilement modéliser son système et pouvoir en évaluer les performances avant de l'expérimenter avec des utilisateurs.

- Enfin, les chapitres 4 et 5 présenteront deux propositions d'amélioration pour les listes de prédiction. Le chapitre 4 présentera une nouvelle interaction pour ce type de liste de manière à pouvoir les utiliser plus régulièrement. Le chapitre 5 étudiera l'opportunité de replacer la liste au centre du système de saisie et ainsi focaliser davantage l'attention de l'utilisateur sur les résultats qui y sont présentés.

Chapitre 1 - Utilisation des systèmes de prédiction pour la saisie de données

Nous présentons et discutons dans cette partie les différentes méthodes de prédiction et d'interaction couplées aux claviers logiciels. Nous commencerons par un survol des systèmes de prédiction existants. Nous proposons ensuite un état de l'art des claviers associés à un système de prédiction. Notre étude distingue deux axes de classification. Elle propose d'une part de classer l'existant par rapport au type de prédiction utilisée par le système de saisie, et d'autre part par rapport à la manière de présenter et d'interagir avec les propositions suggérées par le système de prédiction.

Section I - Les systèmes de prédiction utilisés

Nous entendons par prédiction de texte, de mots ou de caractères, tout mécanisme qui ne fait pas intervenir directement l'utilisateur, mais qui utilise une base statistique ou linguistique (et potentiellement des informations de contexte déjà fournies par l'utilisateur) pour proposer des lettres et des mots. L'utilisation de tels outils a pour objet d'accélérer la vitesse de saisie. Un certain nombre de claviers, que ce soit physique ou logiciel, disposent d'un système de prédiction. Dans un contexte bureautique, ils sont surtout utilisés par des personnes en situation d'handicap moteur afin de leur faciliter l'accès à la saisie de textes. Ils sont en revanche plus largement utilisés en contexte d'interaction dégradée (par exemple, en situation de mobilité sur dispositifs de petite taille tels que les téléphones ou les assistants personnels, PDA). Les systèmes de prédiction sont regroupés en trois grands types :

- Les systèmes reposant sur des études statistiques (fréquences, uni-gramme, bigramme, Ngramme, etc.) ;
- Les systèmes utilisant des études linguistiques (arbre lexicographique, Tree RetrIEvial, TRIE) ;

- Les systèmes utilisant en parallèle les deux mécanismes ci-dessus.

Notons que quelque soient le type et la nature du système de prédiction, celui-ci peut être utilisé pour la prédiction de mots comme pour la prédiction de caractères.

I.1. Prédiction Statistique

Les systèmes basés sur des études statistiques constituent l'une des implémentations les plus classiques des mécanismes de prédiction. Ces systèmes calculent la probabilité pour chaque caractère de succéder à la dernière séquence de caractères reçus, puis les classent par ordre croissant relativement à leur probabilité d'apparition. Cette approche est utilisée dans un grand nombre de systèmes car elle s'est montrée pertinente et facile à mettre en œuvre. Parmi ces approches statistiques, nous distinguons les systèmes de prédiction basés sur les fréquences, les modèles n-gramme ou encore le modèle de Markov d'ordre n.

I.1.a. Prédiction basée sur la fréquence

La méthode la plus simple est de prendre les mots et leurs fréquences en considération [Garay 1994], [Heckathorne 1983], [Hunnicutt 1987], [Swiffin 1987], [Venkatagiri 1993]. Lorsqu'un utilisateur commence à écrire le début d'un mot, le système cherche tous les mots commençant par le préfixe déjà saisi et les classe par ordre de leur fréquence. L'utilisateur a le choix de choisir parmi les options proposées par le système ou de continuer manuellement sa saisie. Dans les deux cas, le système adapte son lexique en sauvegardant le mot de l'utilisateur s'il n'y était pas, ou bien incrémente la fréquence du mot utilisé. Si un nouveau mot est ajouté au système, une fréquence initiale est attribuée.
Afin d'améliorer les résultats de cette approche, il est possible d'enregistrer le dernier mot utilisé en l'affectant d'un coefficient (mot récent). Dans ce cas, le système offre les mots les plus récemment utilisés, choisis parmi les plus probables mots qui commencent par le (s) caractère (s) écrit (s) par l'utilisateur. Bien que les résultats obtenus avec cette méthode soient meilleurs que celles basées sur la fréquence seule [Swiffin 1987], la méthode nécessite le stockage de plus d'informations et accroît la complexité de calcul.

I.1.b. Prédiction basée sur les k-grammes

Un k-gramme fonde la prédiction sur une sous-séquence de k-1 éléments construite à partir d'une séquence donnée, k-1 est alors le nombre d'observations utilisées. L'idée directrice est d'obtenir une fonction fournissant la probabilité d'apparition d'une lettre à partir de ces k-1 observations. À partir d'un corpus d'apprentissage, il est possible de construire une distribution de probabilité pour la prochaine lettre avec un historique de taille k-1.

Souvent cités, les bigrammes et trigrammes sont des modèles de Markov respectivement d'ordre 2 et 3. Le bigramme ne prend en compte pour la prédiction que le caractère qui vient d'être saisi tandis que le trigramme calcule les probabilités d'occurrence sur les deux derniers.

La modélisation n-gramme, qui correspond à un modèle de Markov d'ordre n-1 [Rabiner 1986], calcule la probabilité d'apparition d'une lettre en fonction de toute la séquence déjà saisie (c.-à-d. les n-1 dernières observations, quel que soit n-1). Le module de prédiction classe les lettres en fonction des probabilités de l'ordre n non nulles, puis pour les lettres restantes, il fait appel aux probabilités sur l'ordre n-2 et ainsi de suite jusqu'à un classement complet.

Les méthodes basées sur des modèles de Markov sont combinées à des claviers logiciels afin de prédire ou proposer des mots et des lettres durant la saisie de texte. La robustesse, généralement bonne, est relative à la pertinence du corpus utilisé pour déterminer les occurrences et cooccurrences des lettres. L'entraînement d'un k-gram est donc spécifique pour une langue donnée et parfois spécifique pour un jargon professionnel (Médecine, informatique, etc.). Dans la mesure où ils permettent de fournir une statistique tenant compte de tous caractères préalablement saisis, les n-gram sont souvent plus sensibles à la qualité du corpus et à la présence du vocabulaire utilisé par l'utilisateur dans celui-ci.

Certains claviers comme dans [Trnka 2006] utilisent le modèle uni-gramme pour prédire la première lettre du mot ou le premier terme de la phrase. D'autres comme [Lewis 1999b] exploitent le bigramme pour étudier les cooccurrences des lettres et établir une représentation graphique de celles-ci. Dans Trackball EdgeWrite, [Wobbrock 2006], les

auteurs utilisent le modèle trigramme pour la prédiction de mots, et un modèle bigramme adaptatif tenant compte, par apprentissage, des mots saisis par l'utilisateur.

Ces mêmes modèles sont utilisés dans [Potipiti 2001] pour prédire des mots dans les deux langues: anglaise et Thaï. Les auteurs ont par ailleurs comparé les résultats fournis par une prédiction basée sur des bigrammes et des trigrammes et ont obtenu, dans leur contexte d'usage, des performances supérieures avec le premier modèle. De même, la première version du système HandiAS [Le Pévédic 1998] utilisait un bigramme de lettres comme modèle de prédiction. [Jones 1998] propose un clavier virtuel augmenté par la prédiction de mots et de lettres. Ce clavier se montrera en revanche plus performant avec une prédiction basée sur le modèle trigramme.

A la différence de nombreux claviers ambigüs pour téléphones portables souvent désambiguïsés par la confrontation de la saisie à un dictionnaire, LetterWise [MacKenzie 2001a] confronte la saisie à un tri-gramme ce qui le rend plus léger et plus tolérant à l'absence d'un mot dans le corpus de référence.

Le LURD-Writer du framework HaMCoS [Felzer 2006] complète le préfixe déjà saisi en se basant sur le modèle n-gramme. Le clavier VITIPI, [Boissière 2000] est lui un exemple de système d'aide à l'écriture basé sur un principe d'auto-apprentissage. Le clavier est adapté à tous les déficients moteurs. Il repose lui aussi sur le modèle n-gramme.

La prédiction dans Sibylle, Sibylettre, SibyMot, [Schadle 2001][Schadle 2004] est fondée sur ce modèle statistique avec n=5 (pentagram). Ceci permet d'estimer la probabilité d'occurrence d'un événement (ici des lettres) en fonction des n-1 derniers événements (en fonction des 4 dernières lettres du mot en cours dans Sibyllettre). Dasher [Ward 2000], The Reactive Keyboard [Darragh 1999], Fasty [Beck 2004] , la saisie de mots abrégés [Shieber 2003], ou encore les travaux présentés dans [Copestake 1997], [Lesher 1999], [MacCaul 2004a] sont d'autres exemples de systèmes de prédiction fondés sur ce modèle.

I.2. Prédiction Syntaxique

À chaque frappe de touche, le système prédit la chaîne de caractères la plus probable à partir des connaissances linguistiques (lexique, modèles de langage) et permet à l'utilisateur de simplement vérifier et confirmer (ou non) la prédiction proposée plutôt que de saisir l'ensemble des caractères. Ces systèmes nécessitent une bonne connaissance des propriétés linguistiques et une description syntaxique très précise afin de pouvoir prédire la fin du mot en fonction des mots déjà saisis. Par exemple, l'algorithme T9 [T9], basé sur les travaux de [Witten 1982] et [MacCaul 2004b], prédit, à chaque frappe de touche, la future chaîne de caractères la plus probable à partir de connaissances linguistiques (en l'occurrence un dictionnaire).

I.2.a. Prédiction reposant sur l'arbre lexical

L'arbre lexicographique est un autre exemple de traitement linguistique. C'est une représentation arborescente d'un lexique (cf. Figure 1). Il permet de déterminer, à chaque caractère saisi, l'ensemble des prochains caractères possibles selon un lexique de mots. En effet, chaque mot est représenté par un chemin de la racine à une feuille de l'arbre, où chaque caractère est un nœud. Deux mots ayant le même préfixe utilisent le même chemin initial pour éviter de créer de nouveaux nœuds.

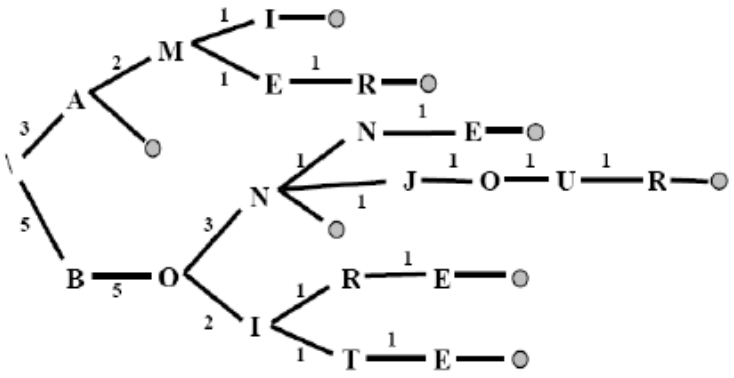


Figure 1: Arbre lexicographique construit à partir des mots a, ami, amer, bon, bonne, bonjour, boire, boîte extraite de [Raynal 2005a].

Le système dans [Raynal 2004] utilise ce principe pour déterminer à chaque saisie, les lettres susceptibles d'être choisies.

I.2.b. Prédiction utilisant le traitement de la langue naturelle.

Le traitement de la langue naturelle est une technique qui consiste à analyser la phrase à travers un ensemble représentatif de règles appelé grammaire. Cette analyse se focalise sur la syntaxe ou le lexique de la phrase, ainsi que sa sémantique. La syntaxe donne l'organisation des mots en structures et la sémantique informe sur le sens des mots / le sens de la phrase. La prédiction de mots développée dans [Copestake 1997] est basée sur une méthode syntaxique : «chart parsing » [Kieras 1993][Allen 1987]. Elle repose sur une grammaire et des règles linguistiques illustrée dans Figure 2.

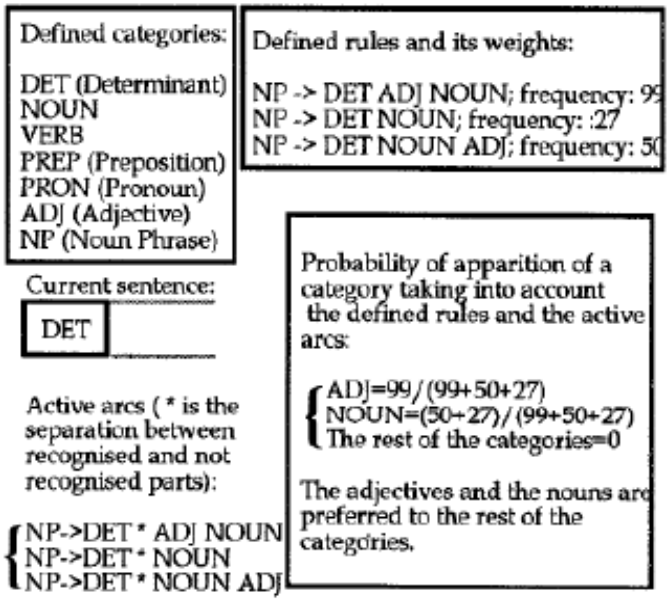


Figure 2: « chart parsing » extrait de [Garay-Victoria 1997]

[Dominowska 2002] exploite des informations contextuelles pour proposer des termes de vocabulaire. [Li 2005] exploite un modèle de relation sémantique pour améliorer les résultats fournis par un n-gram. [Gong 2008] utilise à la fois un modèle de relation sémantique et un modèle de langage pour réorganiser et prioriser les items (fournis par un n-gram).

I.3. Prédiction mixte

A mi-chemin entre les deux approches, cette méthode vise à fusionner les approches syntaxique et statistique déjà citées ci-dessus. Les attributs et propriétés linguistiques sont élaborés par lemmatisation. Ainsi, les phrases peuvent être analysées comme étant des chaînes de caractères. Selon les principaux attributs des mots précédents, le système calcule la probabilité de chaque propriété syntaxique. Bien entendu, ces propriétés ne sont pas spécifiées directement à la conception mais sont automatiquement extraites d'un corpus d'apprentissage. Cette approche mixte a été utilisée par divers systèmes et s'est montrée assez performante du point de vue de la qualité des lettres et des mots prédits. Ces types de claviers utilisent simultanément deux techniques de prédiction ou plus.

Dans le système Unipad [MacKenzie 2006], la prédiction repose sur des études syntaxiques pour suggérer la fin du mot et du suffixe. Cette approche est associée à la technique statistique bigramme et peut être adaptée jusqu'à 4-gramme.

D'autre part, [Ménier 2001] combine la méthode statistique, le modèle de Markov, avec un arbre lexical TRIE (Tree RetrIEvial). Le premier modèle agit sur la prédiction de mots, tandis que le second est utilisé pour la suggestion de lettres. Dans le système KeyGlass [Raynal 2004], deux systèmes de prédiction différents sont utilisés : un, sous forme d'arbre lexicographique et l'autre sous forme de bigramme.

Les systèmes de prédiction peuvent alors être schématisés comme suit (cf. Figure 3):

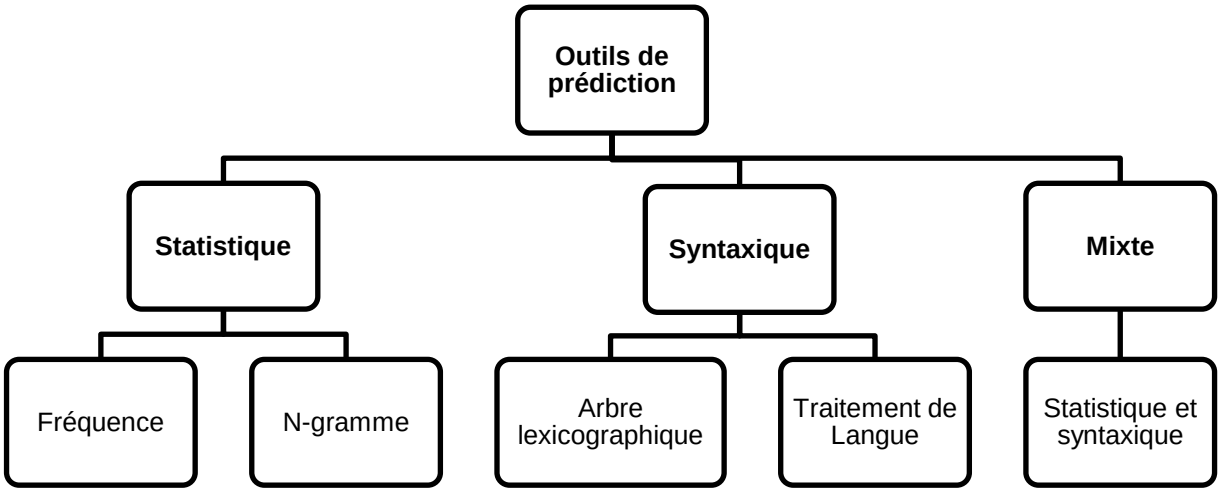


Figure 3: Classification des systèmes de prédiction

Quelque soit le dispositif d'entrée utilisé ou le système de prédiction, la limitation des déplacements du curseur et la réduction du nombre d'opérations restent une priorité. Ces optimisations forment le sujet des études et des recherches en interaction homme-machine.

Section II - Interaction avec les systèmes de prédiction

Différentes stratégies permettent d'exploiter le résultat fourni par un système de prédiction. Nous avons choisi de les diviser en deux axes suivant la nature des résultats fournis par les algorithmes de prédiction : prédiction de caractères ou de mots. Ensuite, nous avons choisi de diviser l'interaction avec la prédiction de caractères en deux sous axes : d'une part les optimisations dites statiques, c'est-à-dire lorsque l'utilisation du système de prédiction a été faite lors de la conception du clavier ; et d'autre part, les optimisations dites dynamiques, c'est-à-dire lorsque le clavier est modifié en cours de saisie pour proposer les meilleures solutions de saisie à l'utilisateur. L'interaction avec la prédiction de mots se divise à son tour en deux parties : d'une part, l'insertion du résultat dans le texte à la position du curseur ; d'autre part la présentation d'une liste contenant les mots candidats qui continuent le préfixe de l'utilisateur.

II.1. Interaction avec la prédiction de caractères

La machine à écrire fut inventée au XVIII^{ème} siècle (l'américain Henry Mill déposa le premier brevet en 1714) et subit plusieurs modifications au niveau de l'agencement des touches. La disposition QWERTY et AZERTY a été conçue de manière à ralentir la vitesse de saisie de texte. Ceci ne pose pas de problème sur un clavier physique où l'interaction se fait avec 10 doigts. Cependant cette disposition est plus problématique avec les claviers logiciels, sur lesquels l'utilisateur n'utilise qu'un doigt ou un pointeur. Ainsi de nombreuses recherches ont été menées pour optimiser les dispositions de la position des caractères de façon à augmenter les performances du clavier et à accélérer la vitesse de saisie. Ces recherches incluent l'alliance des systèmes de prédiction avec les claviers logiciels. Deux techniques d'optimisation peuvent être distinguées : les optimisations dites statiques faites lors de la conception du clavier et des optimisations dynamiques qui arrangent les lettres au fur et à mesure de la saisie.

II.1.a. Optimisations statiques

Quelque soit le dispositif d'entrée utilisé, la limitation des déplacements du curseur est bien une priorité. Afin de minimiser au plus la distance à parcourir d'une touche à l'autre, des études ont proposé de nouvelles configurations spatiales plus adaptées à une interaction à un seul pointeur. Ces approches sont fondées sur des études statistiques des occurrences et cooccurrences des lettres et ont pour objectif de rapprocher les caractères qui ont le plus de chance de se succéder.

Optimisations « manuelles »

Dans un premier temps, la lecture des tables de fréquences d'apparition des caractères a amené les concepteurs à proposer des réagencements de la disposition des caractères sur le clavier. Nous appelons ces optimisations « statiques » car elles ont lieu au moment de la conception du clavier et ne sont pas modifiées durant la saisie de l'utilisateur. De plus, on dit qu'elles sont manuelles car elles sont proposées par le concepteur et non pas par un algorithme.

La disposition la plus intuitive ou naturelle est celle des claviers alphabétiques. Ces systèmes sont constitués des lettres disposées les unes après les autres dans l'ordre alphabétique. Bien que cet agencement ne diminue pas les distances et les déplacements du curseur, il a l'avantage d'être connu par tout le monde. Norman et Fisher [Norman 1982] ont distribué les lettres sur plusieurs lignes (Figure 4 A). D'après leurs expérimentations, les claviers alphabétiques ne sont pas plus performants que les claviers QWERTY. [MacKenzie 1999b] propose de réduire le clavier sur deux lignes contenant chacune 13 lettres (cf. Figure 4 B). Mais la vitesse de saisie reste inférieure à celle avec un clavier ordinaire à cause de la grande distance à parcourir entre les caractères éloignés.

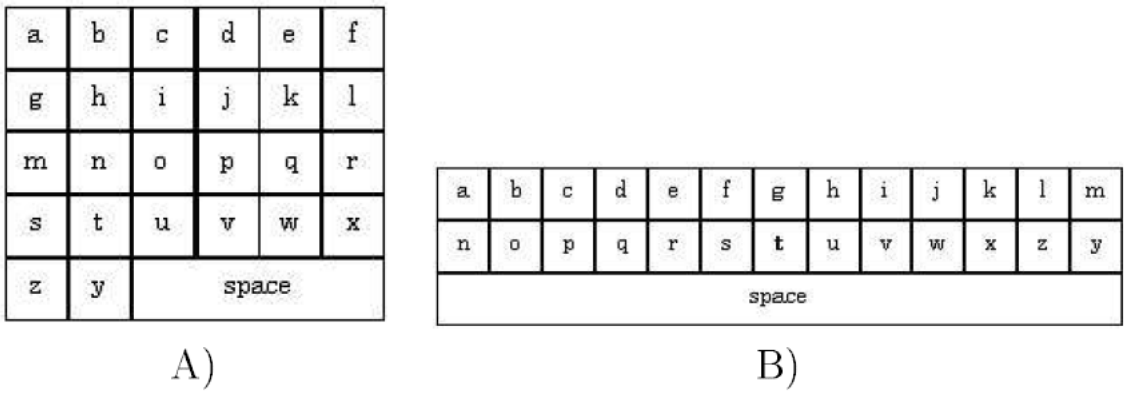


Figure 4: Claviers alphabétiques: A) configuration 5 x 6 proposée par [Lewis 1999b] ; B) configuration 3 x 13 proposée par [MacKenzie 1999b]

L'autre solution de concevoir des claviers est la méthode intuitive. Le clavier DVORAK dans sa forme logicielle a été réalisé en rapprochant les caractères qui ont le plus de chance de se succéder dans la langue anglaise. Ceci a pour but de diminuer les mouvements du stylet sur le clavier. Cependant, ce clavier a été conçu au début pour un usage avec les deux mains, ce qui explique les performances modestes de ce système par rapport au clavier QWERTY [MacKenzie 1999b].

Le clavier logiciel Chubon [Chubon 1988] a été conçu pour optimiser l'utilisation d'un pointeur. D'après les évaluations « *a priori* » de [MacKenzie 2002d], ce clavier a réussi à faire gagner plus de 20% en vitesse de saisie comparé au QWERTY. Dans ce domaine, les

claviers logiciels les plus connus sont FITALY[2] et OPTI [MacKenzie 1999a] créés dans le but de minimiser les distances parcourues par le pointeur lors de la saisie. Dans ces claviers, les caractères les plus fréquents ont été placés au centre du clavier, à côté des caractères qui ont plus de chance à se succéder. Comme l'espace est le caractère le plus utilisé, plusieurs touches espaces ont été utilisées dans les claviers (cf. Figure 5). Ces systèmes présentent des avantages pour les experts mais les performances des novices sont assez basses.

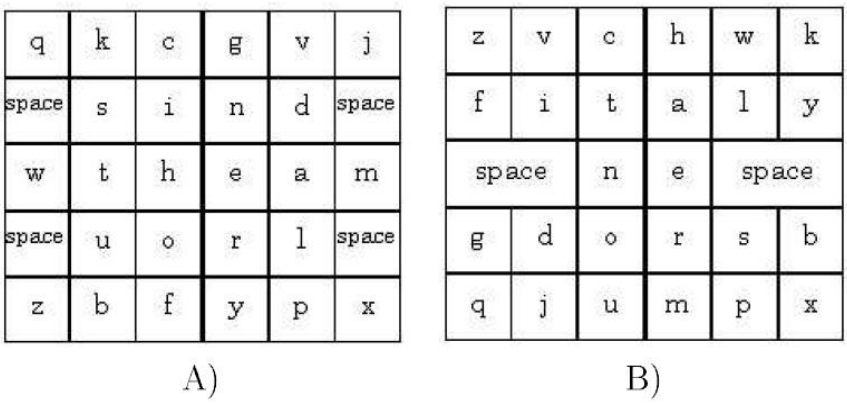


Figure 5: Claviers intuitifs: A) Clavier OPTI; B) Clavier FITALY

Un autre clavier conçu par la méthode intuitive est le clavier CATKey [Go 2007] mais en utilisant le digramme de Voronoï[3]. Ce système permet à l'utilisateur de concevoir son propre clavier de type AZERTY ou QWERTY avec la possiblité de changer la taille des touches de façon à être confortable dans sa saisie. Cette méthode admet à l'utilisateur d'augmenter la taille des touches qu'elles lui paraîssent intéressantes et de diminuer les autres. A savoir que les grandes touches ne masquent pas celles de taille réduite.

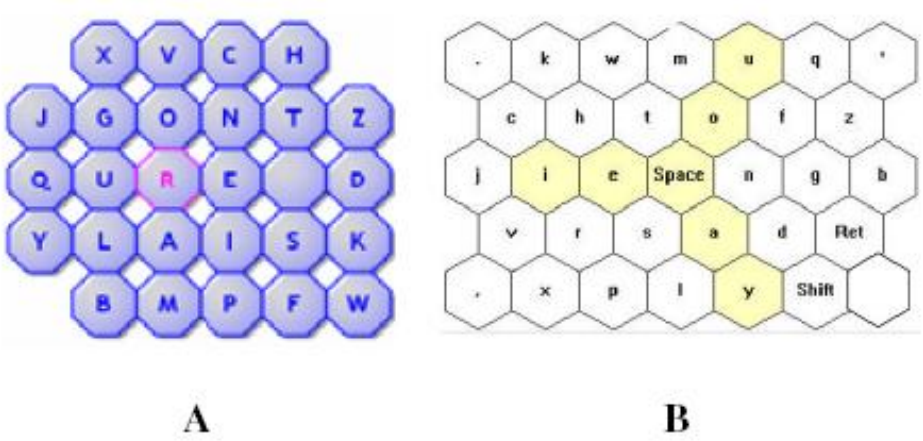
[2] Textware Solutions Inc. http://www.textwaresolutions.com/

[3] En mathématiques, un diagramme de Voronoï (aussi appelé décomposition de Voronoï) représente une décomposition particulière d'un espace métrique déterminée par les distances à un ensemble discret d'objets de l'espace, en général un ensemble discret de points [Buisson 2002].

Optimisations par algorithme

Loin des solutions naturelles et intuitives, on peut trouver les méthodes qui appliquent des algorithmes d'optimisation. Ces systèmes calculent un ensemble de possibilités et ne gardent que les meilleures. Le premier système conçu à partir de telle méthode est le clavier Getschow [Getshow 1986] qui utilise le « greedy algorithm » (ou algorithme par construction) et les fréquences d'apparition des lettres dans la langue anglaise pour positionner les touches de façon à minimiser les distances moyennes entre elles. Lewis et ses collègues [Lewis 1999a], [Lewis 1999b] proposent de remplir une matrice symétrique avec la fréquence d'apparition de chaque cooccurrence. Ensuite, ils cherchent le chemin qui passe par tous les caractères et qui minimise les distances entre les fortes cooccurrences. L'inconvénient de cette méthode est qu'elle ne prend pas toutes les connexions possibles sur un clavier.

Des claviers tels que ceux générés par GAG [Raynal 2005b] (cf. Figure 6 A), sont le résultat d'un algorithme génétique fondé sur les bigrammes d'occurrences de la langue française. Le but étant toujours de réduire les distances entre les touches pour minimiser les déplacements du curseur en tâche de saisie. Raynal a constaté un apport en performance de 52,36% dans sa première version (GAG I) et 55,23% dans la seconde (GAG II). L'algorithme génétique fut aussi utilisé pour arranger les caractères sur un clavier ambigu [Sad 2008]. Dans ce cas, l'intérêt est d'augmenter les performances de saisie tout en limitant le nombre de frappes à réaliser sur une touche pour accéder au bon caractère. Cette technique montre tout son intérêt quand le nombre de touches est faible (3).



A B

Figure 6: A) Le clavier GAG ; B) Le clavier Métropolis

D'autres comme Metropolis et Hooke [Zhai 2000] (cf. Figure 6 B) appliquent les lois de la physique et de la thermodynamique aux tables de bi-grammes pour réduire les distances entre les lettres les plus fréquemment consécutives. Le clavier Hooke prend son nom de la loi physique de Hooke. Dans ce clavier, chaque liaison entre deux caractères est vue comme un ressort dont l'élasticité est proportionnelle à la cooccurrence. Le clavier est conçu d'une façon à avoir une tension minimale pour chaque ressort. L'algorithme Metropolis [Metropolis 1953] est utilisé pour créer le clavier portant son nom, de manière à diminuer l'énergie matérialisée par la loi prédictive de Soukoreff.

L'algorithme des fourmis a été la base pour trouver un arrangement optimisé pour les touches d'un clavier utilisé par les personnes en situation d'handicap [Colad 2008].

Le Clavier KNITS proposé par Lesher [Lesher 2000] utilise l'algorithme n-opt initialement proposé par [Lin 1965], [Lin 1973] pour résoudre le problème du voyageur de commerce. Cet algorithme est utilisé pour rechercher le meilleur agencement possible pour ces « n » caractères. Cela consiste à échanger la position de deux caractères sur le clavier et à tester les performances jusqu'à l'obtention de la meilleure solution. Cette méthode est dite de 'voisinage'.

Le clavier Atomik (Alphabetically Tuned and Optimized Mobile Interface Keyboard) [Buisson 2002] (cf. Figure 7) est une implémentation adaptée à une utilisation sur petite surface tactile. C'est une disposition qui satisfait à la fois des critères statistiques, mais qui prend par

ailleurs en compte une certaine logique de connectivité des caractères : les caractères des mots les plus fréquemment utilisés en anglais ont été positionnés les uns à proximité des autres (par exemple les caractères ‘T’, ‘H’ et ‘E’ qui forment le mot ‘the’).

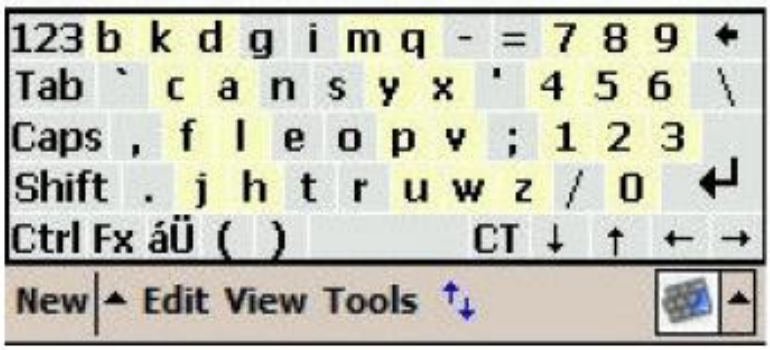


Figure 7: Le clavier Atomik

Un autre clavier logiciel a été conçu en modifiant automatiquement la disposition des touches de façon à avoir des performances supérieures à celles des claviers AZERTY ou QWERTY. C’est le clavier multi-layer [Merlin 2011]. L’idée est « d’accompagner l’utilisateur dans la transition de la distribution de touches originelle (AZERTY, QWERTY, etc.) vers une distribution de touches optimisée ». Les touches adjacentes par le côté sont permutées jusqu’à 39 fois afin d’obtenir la version finale du clavier (cf. Figure 8), la géométrie du clavier étant fixe. Les performances théoriques évoluent de 28.7 à 38.5 wpm.

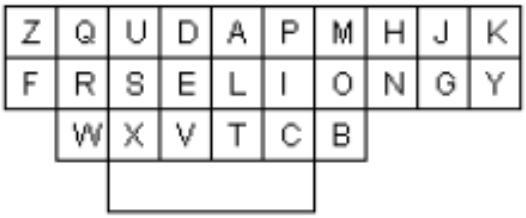


Figure 8: Le clavier multi-layer

Le Tableau 1 présente un récapitulatif des performances théoriques[4] (en nombre de mots saisis par minute (MPM)) sur les claviers logiciels conçus lors de la programmation. On peut ainsi y voir que toutes les optimisations proposées ont généralement des performances théoriques bien plus importantes que celles du clavier QWERTY qui est actuellement la référence car le plus utilisé.

Clavier	**Performance en MPM**	**Clavier**	**Performance en MPM**
QWERTY	29.91	Hooke	41.15
GAG I	45.57	Chubon	37.02
GAG II	46.43	Lewis	34.65
Metropolis	42.94	ABC	32.5
OPTI	42.37	FITALY	41.96
		Multi-layer	38.5

Tableau 1 : Tableau donnant les performances en vitesse des claviers présents dans le chapitre, extrait de [MacKenzie 2002d] et [Raynal 2005a].

Si ces dispositions de caractères ne sont pas plus utilisées, c'est à cause du coût que représente l'apprentissage de la nouvelle disposition des caractères. Mackenzie [MacKenzie 1999b] a ainsi montré qu'il fallait environ 20 sessions pour juste atteindre le même niveau de performances que celles que nous avons avec QWERTY. Ce temps d'apprentissage est donc trop long pour pousser un utilisateur à changer de disposition de caractères.

Optimisation de la disposition des caractères des claviers à défilement

[4] Ces valeurs théoriques ont été obtenues au moyen de la loi de Fitts en prenant comme Indice de performance (IP) 4,9 bits/s.

Les dispositions que nous venons de proposer sont optimisées pour une interaction avec un pointeur. Pour une sélection basée sur le balayage du clavier logiciel, [Cantegrit 2001] ont trouvé une configuration optimale suivant le nombre de touches du clavier. Pour un balayage ligne/colonne, les auteurs ont montré que la disposition optimale des touches est sur une matrice triangle rectangle au point en haut à gauche de l'écran (cf. Figure 9). La disposition des lettres sur la matrice dépend de leur fréquence d'apparition dans la langue.

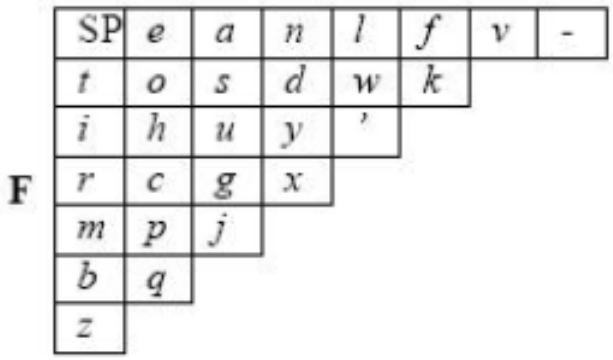


Figure 9: Clavier optimisé (pour l'anglais) pour un défilement ligne/colonne

Dans le clavier UKO-II (Unbekanntes Kommunikatios Objekt – Objet de communication non identifié) [Harbusch 2003], les agencements de lettres sont optimisés en fonction de la langue utilisée. Le clavier, allié à un système de prédiction, présente en revanche seulement 4 boutons sur lesquels sont réparties les lettres de l'alphabet. Dans LURD-Writer [Felzer 2006], les lettres sont également réparties entre 4 touches positionnées sur les quatre côtés par rapport à une zone centrale. L'utilisateur accède aux touches par des mouvements directionnels (haut-bas-droite-gauche) d'une souris virtuelle contrôlée par les impulsions musculaires de deux muscles.

Dans des claviers logiciels comme WiVik[5] , la disposition des touches de certains claviers peut être dépendante du type de sélection (par défilement ou par pointage). Les caractères sont alors présentés selon leur fréquence d'apparition dans la langue pour la sélection par

[5] http://www.wivik.com/index.html

défilement. Pour les utilisateurs munis d'un dispositif de pointage, un positionnement alphabétique est appliqué aux touches du clavier.

Claviers ambigus ou « multi pages »

D'autres approches proposent des claviers logiciels réduits avec des interfaces ne contenant pas toutes les lettres à la fois. Les claviers de ce type ne sont pas nombreux. On peut trouver le clavier DotNote[6] (Figure 10) qui répartit les touches sur deux interfaces réduisant ainsi le nombre de touches affichées à la fois sur l'écran. La première interface contient les caractères les plus fréquents et l'autre les moins fréquents. Cependant, d'après [MacKenzie 2002d], les performances maximales que pourrait avoir « *a priori* » un expert avec DotNote sont inférieures à celles qu'il pourrait avoir avec un clavier QWERTY.

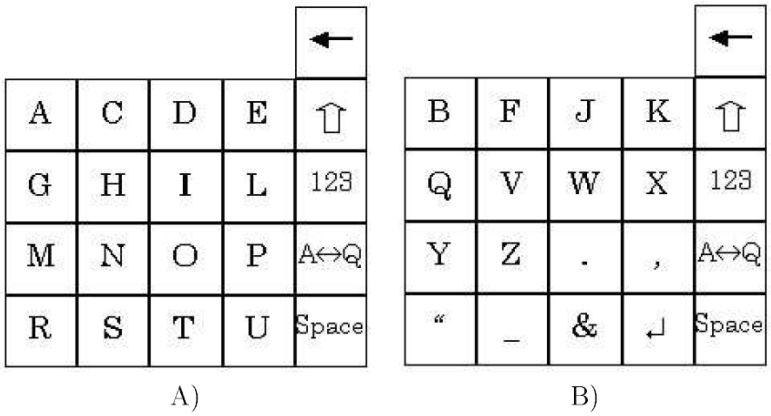


Figure 10: Le clavier DotNote: A) les lettres les plus fréquentes; B) les caractères les moins utilisés.

La disposition téléphonique se présente aussi comme une autre solution de clavier réduit. Deux principales méthodes de distribution de lettres sur les touches peuvent être distinguées. L'ordonnancement alphabétique et JustType [King 1995] (cf. Figure 11). D'après [MacKenzie 1999b], il est possible d'avoir des performances équivalentes avec ces types de clavier qu'avec QWERTY. Un utilisateur expert peut saisir 43.5, 44.2 et 43.2 mots par minutes respectivement avec le clavier Téléphone, JustType et QWERTY.

[6] DotNote est un produit de Utilware : http://utilware.com

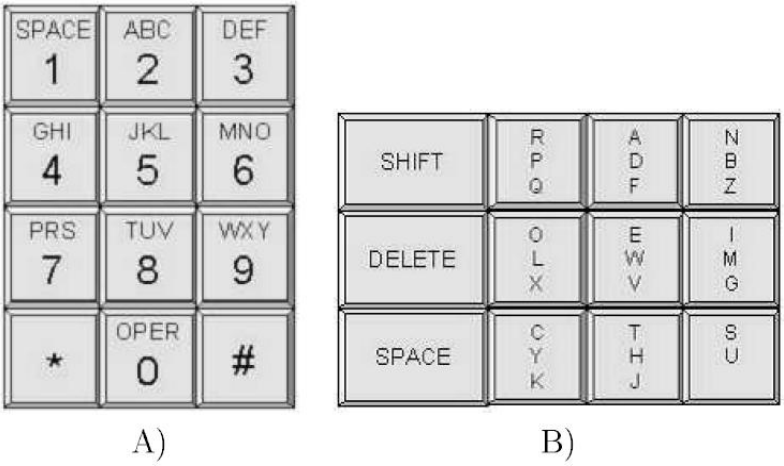


Figure 11: Claviers téléphoniques, A) clavier téléphonique classique; B) clavier JustType

Plusieurs travaux observent que l'ordre alphabétique initialement proposé sur les claviers ambigus n'est pas le mieux adapté à la désambigüisation par un système de prédiction et proposent d'autres agencements de lettres. [Arnott 1992] compare, avec le même protocole et le même système de prédiction, l'efficacité de la prédiction en fonction des différents agencements proposés par Levine [Levine 1987], TOC [Foulds 1987] et Frequency [Arnott 1992]. L'indice d'efficacité de la prédiction est établi à l'inverse du nombre de frappe par caractères et exprimé en pourcentage. Par exemple, [Foulds 1987] nécessite 1.12 frappe par caractère (l'efficacité de la prédiction est donc évaluée à 89%, 1/1.12 * 100) contre 1.14 (88%) pour [Arnott 1992]. Sur cette base de comparaison, Lesher [Lesher 1998] propose une méthode pour élaborer un agencement optimal en fonction d'une langue (et plus largement d'un corpus). L'agencement proposé pour la langue anglaise augmente l'efficacité de la prédiction à 92%.

D'autre part, [Sad 2008] a montré que le nombre de touche était aussi très important sur l'efficacité de la prédiction. Ainsi avec seulement 3 touches, le taux de bonne prédiction est de 80%, alors qu'il augmente respectivement à 98% et 99% avec 5 et 8 touches.

II.1.b. Interactions dynamiques

Par interactions dynamiques, nous entendons modification du clavier au fur et à mesure des saisies de l'utilisateur. Parmi ces interactions, nous distinguons les réarrangements des touches, les agencements des lettres, et les ajouts de nouveaux composants. Les systèmes de prédiction sont alors utilisés pour réduire le nombre de frappes ou augmenter la vitesse de saisie en réduisant dynamiquement les distances entre les touches. Les animations/interactions proposées sont variées.

L'agencement dynamique des lettres est ainsi utilisé dans le clavier FOCL (Fluctuating Optimal Character Layout) [Bellman 1998] qui refond les caractères de façon à ce que les plus probables se trouvent le plus près possible du pointeur. De même, le système Sibylettre [Schadle 2001], est destiné à des utilisateurs atteint d'une déficience physique majeure. Sibylettre est basé sur un défilement automatique linéaire des touches. L'utilisateur appuie sur un bouton poussoir lorsque le système donne le focus à la bonne touche. Le système réarrange les touches associées aux caractères en fonction de ce qui vient d'être saisi (cf. Figure 12) et ordonne les caractères en fonction de leur probabilité d'apparition. Les caractères les plus probables sont ainsi projetés en tête de la liste de défilement.

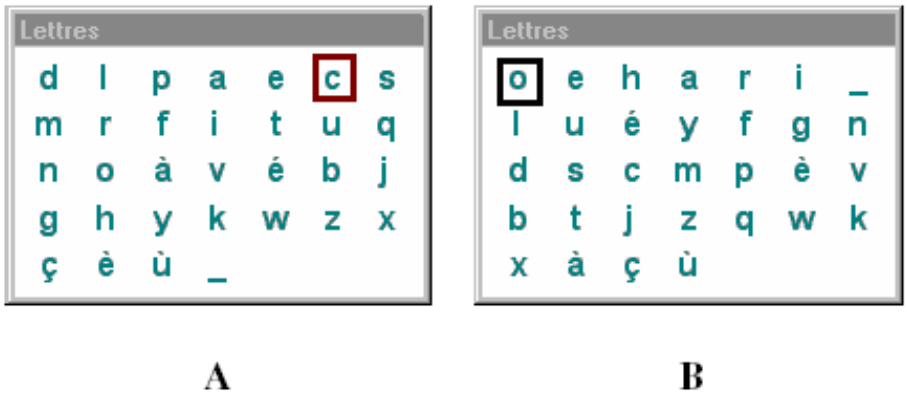


Figure 12: Sibylettre : A) Saisie du caractère 'c' ; B) Réagencement des caractères après la saisie du 'c'

Cette même technique d'interaction est trouvée dans ChewingWord [Grange 2010] (cf. Figure 13). Théoriquement, ces claviers simplifient le temps de recherche d'un caractère et réduisent le temps et les déplacements. Cependant, ce genre de clavier peut s'avérer fatigant pour l'utilisateur car il nécessite une charge supplémentaire : l'utilisateur est en permanence en train de rechercher visuellement le caractère qu'il souhaite saisir. Dès lors, il lui est impossible d'apprendre l'agencement des caractères sur le clavier et donc d'anticiper ses mouvements.

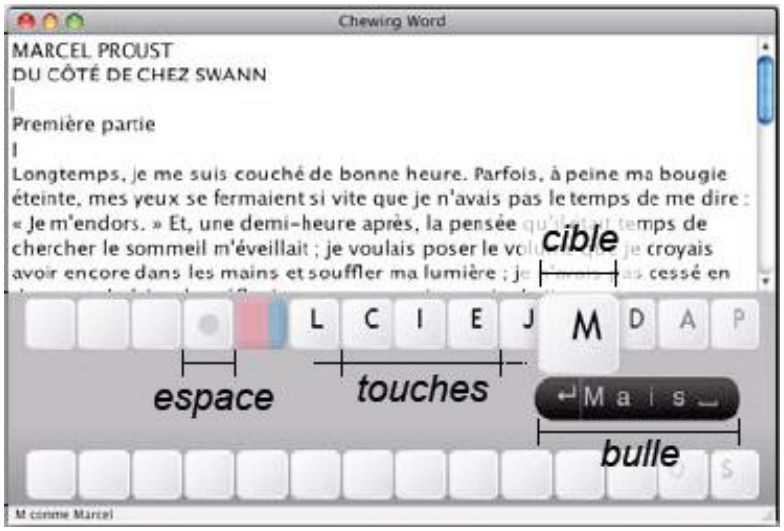


Figure 13: L'interface Chewing Word [Grange 2010]

L'agencement dynamique peut également être exploité pour des claviers type téléphone [Raynal 2005a]. Sur chaque touche, le caractère ayant la plus forte probabilité d'être saisi est placé en première position. Sachant que sur ce type de clavier, les caractères sont regroupés par trois sur chaque touche et que les caractères positionnés en deuxième et troisième position nécessitent respectivement deux et trois clics pour être atteints, le réagencement des caractères revêt alors une importance particulière.

Dasher [Ward 2000] propose un mode de saisie de texte très singulier (cf. Figure 14 A). Il ne s'agit pas d'une émulation de clavier présenté à l'écran, mais d'une interface nouvelle qui allie sélection continue et prédiction de mots. Ce système de saisie est basé sur une prédiction déterminant la taille des lettres présentées à l'écran et l'agencement de caractères

qui ont le plus de chance d'être sélectionnés. Les caractères les plus probables sont affichés dans un rectangle de taille supérieure de manière à mettre la « touche » en évidence et de manière à rendre ces caractères plus accessibles. Le système fait défiler les lettres. Le défilement est recentré avec un effet « fisheye » [Furnas 1986] relativement au déplacement vertical du pointeur. Lorsque le curseur rencontre une nouvelle lettre, celle-ci est empilée aux autres lettres préalablement saisies. La vitesse de défilement est, elle, contrôlée par le déplacement horizontal du curseur : plus le curseur est déplacé vers la droite, plus le défilement est rapide ; si le curseur est déplacé vers la gauche, le déplacement est effectué en sens inverse permettant ainsi de dépiler les lettres préalablement saisies.

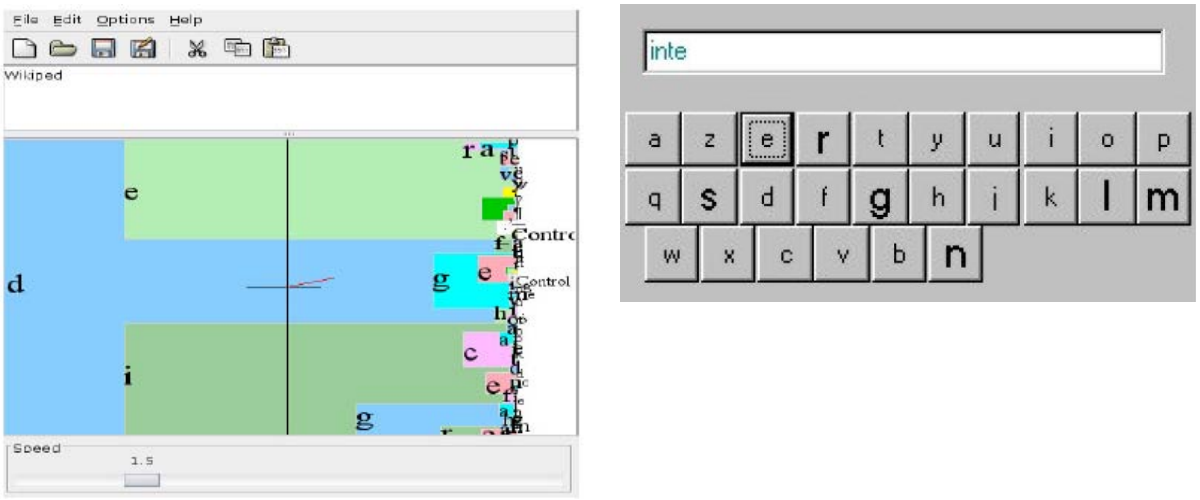


Figure 14: Interactions dynamiques avec la prédiction de caractères : A) l'interface Dasher; les caractères probables sont en forme de carreaux colorés ; B) Mise en contraste des caractères prédits.

Dans les travaux de [Merlin 2009] les auteurs ont choisi de « recycler » les caractères qui n'ont pas de chance de suivre le caractère déjà saisi par l'utilisateur. Ce clavier nommé SpreadKey, est basé sur le clavier standard AZERTY ou QWERTY substitue temporairement les caractères de très faible probabilité par des caractères plus probables tout en laissant accessibles les caractères de faible chance aveec une pression longue sur la touche. Le caractère originel de la touche reste visible dans l'angle supérieur gauche de la touche (cf. Figure 15). Un tel système permet de réduire la distance à parcourir par le curseur et

augmente la taille de la touche à saisir en recyclant les touches voisines se traduisant ainsi par une augmentation de vitesse de saisie.

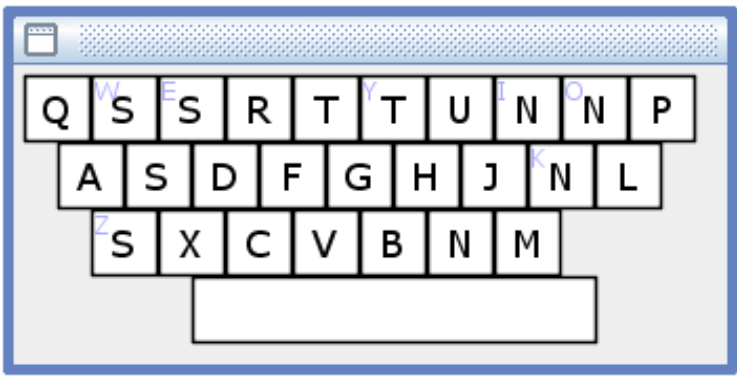


Figure 15: Le système SpreadKey après la saisie du caractère « e » : Les caractères W, E, Z, Y, I, O, K sont improbables et sont remplacer par S, T et N.

Nous distinguons aussi des systèmes de saisie qui mettent en relief les caractères prédits pour attirer l'attention de l'utilisateur lors de la saisie. Dans [Magnien 2003], les touches susceptibles d'être sélectionnées sont prédites et mises en contraste afin d'aider l'utilisateur au cours de la saisie. Ce type de clavier apporte, d'après la loi de Hick-Hyman, un gain théorique de 50% pour un usager novice (cf. Figure 14 B). De même BigKey [AL Faraj 2009] augmente la taille des 4 touches les plus probables en utlisant le digramme de fréquence pour calculer la probabilité. Plus la probabilité du caractère est grande, plus la taille de la touche est grande (cf. Figure 16). Cela peut faciliter la tâche de pointage pour les utilisateurs. Mais l'augmentation de la taille d'une touche est limitée de façon à ne pas masquer les autres touches.

Figure 16: L'interface BigKey après la saisie de « t »

D’autre part, pour remédier à ce type de contrainte FloodKey [Aulagner 2010], offre aux utilisateurs un clavier redimensionnable qui change automatiquement la taille des touches suceptibles de suivre le caractère saisi. La taille de certaines touches augmente et la taille d’autres diminue en fonction de la probabilité des caractères (cf. Figure 17). Les probabilités sont calculées par la méthode des 5-grammes. FloodKey se base sur le clavier AZERTY/QWERTY - facilement repéré - et utilise le diagramme de Voronoï3 pour changer dynamiquement la taille des touches après chaque caractère entré.

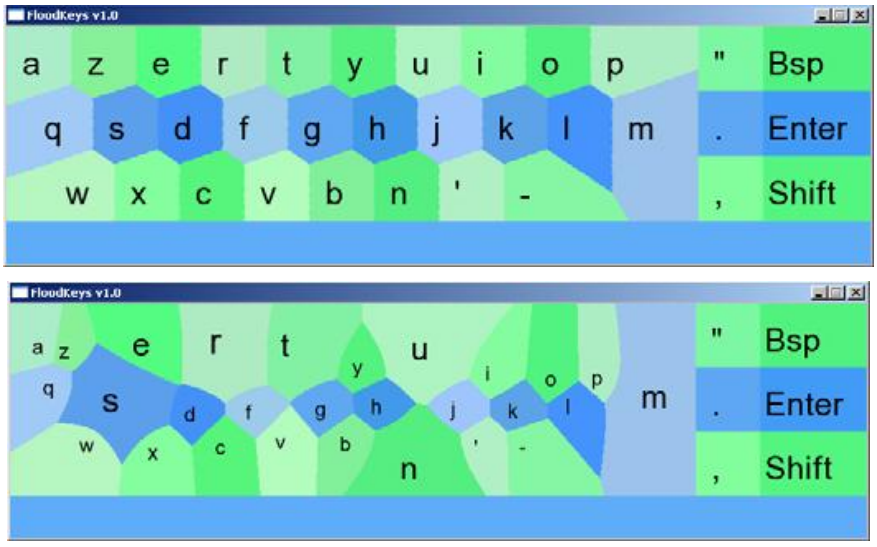


Figure 17: FloodKey avant la saisie de lettres (en haut) et après la saisie du mot « joue » (en bas)

Une autre forme d’animation est l’ajout dynamique de nouveaux composants sur l’interface du clavier comme de listes ou de touches au cours de la saisie. Le système KeyGlass (cf. Figure 18) [Raynal 2004] repose sur le principe d’ajout de touches supplémentaires semi-transparentes affichant les caractères les plus probables autour du caractère qui vient d’être saisi. Ainsi le clavier de base reste entièrement visible, et chaque touche du clavier reste accessible même avec les touches supplémentaires.

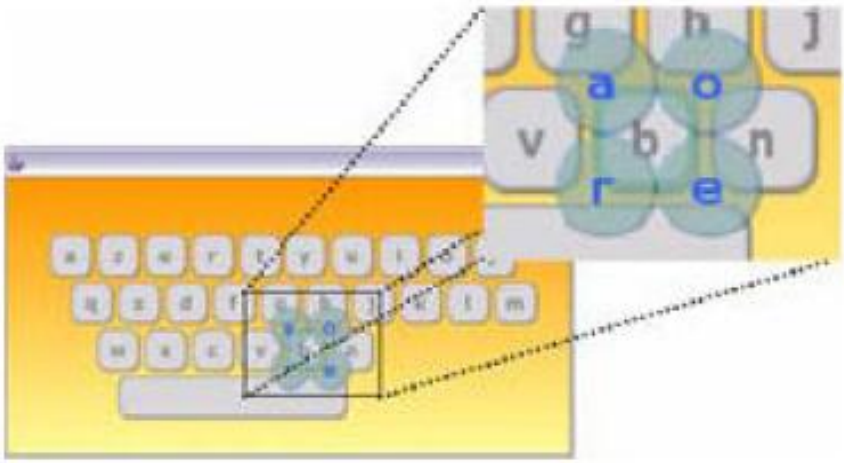


Figure 18: Le système KeyGlass. 4 touches semi transparentes positionnées autour de la touche déjà saisie.

Le clavier SlideKey (Figure 19) [Godard 2010] est un clavier polyvalent qui enferme toutes les techniques déjà citées : c'est un clavier réduit, dont les caractères sont disposés sur une ligne suivant leur fréquence dans la langue. Après chaque saisie de caractère, les lettres du clavier sont réorganisées en fonction de leur probabilité de succéder à ce qui vient d'être saisi. Pour aider l'utilisateur à anticiper les caractères suivant, SlideKey propose aussi deux autres lignes contenant les lettres qui ont le plus de chance de succéder au caractère sélectionné sur la première ligne (rectangle rouge sur la figure ci-après). Ces deux lignes complémentaires permettent ainsi à l'utilisateur de visualiser rapidement les caractères qui pourraient suivre. Ceci représente un avantage en fin de mot où les caractères les plus probables sont souvent ceux saisis par l'utilisateur. Dans ce cas de figure, l'utilisateur anticipe alors la saisie et n'attend pas le réagencement des touche pour valider le caractère suivant. Il gagne ainsi en rapidité de saisie.

Figure 19: L'interface de SlideKey

Hormis les optimisations proposées pour les claviers à défilement automatique, les techniques basées sur de la prédiction de caractères et proposées pour les claviers à base de pointage sont généralement moins efficaces que le clavier logiciel ordinaire. Ceci est du au fait que le changement apporté sur l'interface nécessite un temps de prise en compte par l'utilisateur qui le ralentit dans sa saisie. Le gain apporté par la technique est alors compensé par ce temps de prise en compte. Par conséquent, l'optimisation caractère par caractère n'est pas très avantageuse. C'est pourquoi, plusieurs études visent à améliorer la saisie en proposant une prédiction ou une prédiction au niveau du mot. Proposer un mot ou plusieurs caractères d'un coup parait alors difficilement compensable juste par le temps de prise en compte.

II.2. Interaction avec la prédiction de mots

Un autre type d'aide à la communication concerne la prédiction des mots. Celle-ci repose dans la plupart des cas sur un moteur de prédiction linguistique. Comme tout système de prédiction, son rôle est d'accélérer la saisie ainsi que de diminuer le nombre d'opérations qu'un sujet doit faire pour entrer le même message sans prédiction. Cette méthode diminue aussi le nombre d'erreurs orthographiques.

Selon [Greenberg 1995], l'aide à la saisie de texte proposée par les systèmes de prédiction de mots se fait de deux manières principales :

- La première méthode consiste à afficher une liste de prédiction contenant les mots les plus probables permettant à l'utilisateur de compléter le préfixe déjà saisi. Le sujet a alors le choix de sélectionner un mot dans la liste ou de continuer sa saisie sur le clavier dans le cas où le mot désiré n'est pas encore affiché.
- Dans la deuxième technique, le système insère directement le mot prédit dans le texte. Si le mot introduit est celui visé par l'utilisateur, celui-ci continue son message. Sinon, l'utilisateur le rejette et continue manuellement le mot qu'il est en train de saisir.

II.2.a. Insertion dans le texte

Comme dans tout clavier, l'approche consiste à saisir un mot, caractère par caractère et dès qu'il n'y a plus d'ambigüité, le système affiche, soit une partie du mot, soit la fin du mot. L'utilisateur a le choix soit de refuser la proposition et de complèter ainsi manuellement le mot qu'il doit entrer, soit de le valider et continuer son texte.

Cette méthode a été mise en œuvre dans le système VITIPI [Boissière 2000] qui réalise cette insertion automatique. De plus, il prend en considération les fautes de frappe dues aux difficultés motrices et certaines fautes d'orthographe, et propose de les corriger. Par exemple, lors de la saisie, si le sujet a tapé la suite de lettres « auk ». Comme il n'existe aucun mot dans la langue française qui commence par ce préfixe, VITIPI fait l'hypothèse qu'il y a une erreur de frappe entre k et j (position de la lettre sur le clavier : k à côté de j). Il substitue donc le k par le j et propose « aujourd'hui » qui est le seul mot dans son corpus commençant par « auj ».

L'algorithme de Automatic Whiteout++ [Clawson 2008] utilise à la fois des tri-grammes, la disposition des touches sur le clavier et le rythme de la frappe (temps de frappe entre chaque touche) pour détecter et corriger 37% des erreurs de frappe sur des claviers mini-QWERTY.

II.2.b. Affichage de liste de mots

Lorsque le système de prédiction présente plusieurs propositions au cours de la saisie, la liste de mots correspondants est affichée dans une zone spécifique du clavier. Les mots sont rangés dans une liste selon un ordre alphabétique ou suivant un taux décroissant de prédiction dans laquelle l'utilisateur peut sélectionner les mots « *candidats* ». Dans ce cas, un appui supplémentaire est nécessaire pour choisir le mot de la liste. De nombreux systèmes intègrent ce mode : on peut mentionner le système HandiAS [Maurel 2001], le système PROPHET [Carlberger 1997], [Hunnicutt 2001], et le système PolyPredix [Bérard 2004b].

Dans la plupart des cas, les listes sont positionnées à gauche du clavier comme c'est le cas dans Keystrokes[7] (cf. Figure 20 à gauche) et HandiAS [Maurel 2001] (Figure 20 à droite). Dans d'autres claviers tels Pen-Operation Based on eXample - POBox [Masui 1999], la liste est affichée sous forme de touches directement au voisinage de la touche déjà saisie (cf. Figure 21 à gauche). Le problème soulevé par ce type d'affichage est que la liste de mots proposés cache en partie le clavier. Ceci peut alors gêner l'utilisateur dans sa saisie si celui-ci ne valide aucune proposition de la prédiction et cherche à saisir un caractère se trouvant sous la liste de mots.

Le système KOMBE [Guenthner 1992] utilise un dictionnaire pour guider l'utilisateur dans la composition de ses phrases. L'utilisateur compose pas à pas son message en sélectionnant des mots parmi une longue liste contenant tous les mots possibles (cf. Figure 21 à droite).

[7] http://www.assistiveware.com/keystrokes.php

Figure 20: Affichage de la liste de prédiction: A gauche le système KeyStrokes, A droite HandisAs.

FASTY [Beck 2004], Dicom [Marcou], le Premier Predictor Pro - P3[8] ainsi que d'autres systèmes commerciaux, affichent une liste là où se trouve le curseur. A noter que ces trois derniers systèmes ne sont pas des claviers logiciels, mais sont alliés à des systèmes de prédiction de mots et présentent une liste des mots prédits.

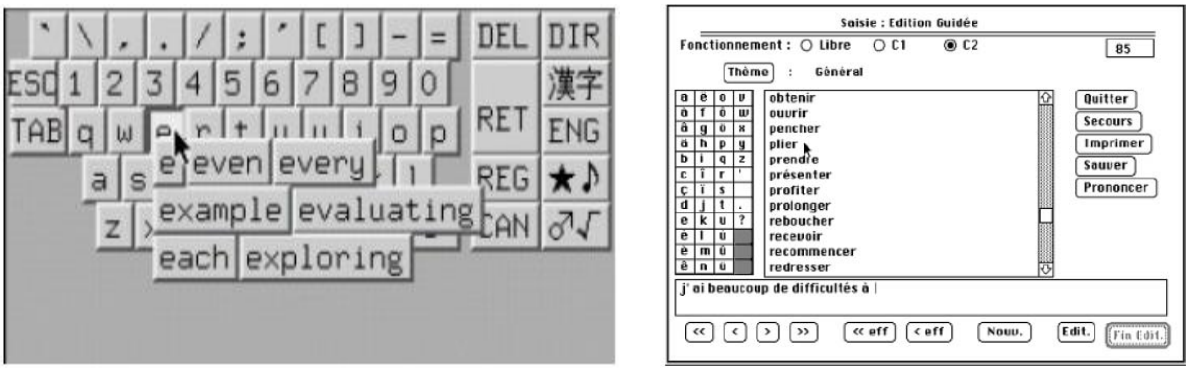


Figure 21: A gauche le système POBox, A droite le système KOMBE

[8] Premier Literacy, http://www.readingmadeeasy.com

Pour résumer, l'utilisation de systèmes de prédiction avec les claviers logiciels du point de vue interaction et présentation des résultats de la prédiction pour les utilisateurs peut être classée dans le schéma suivant de la Figure 22.

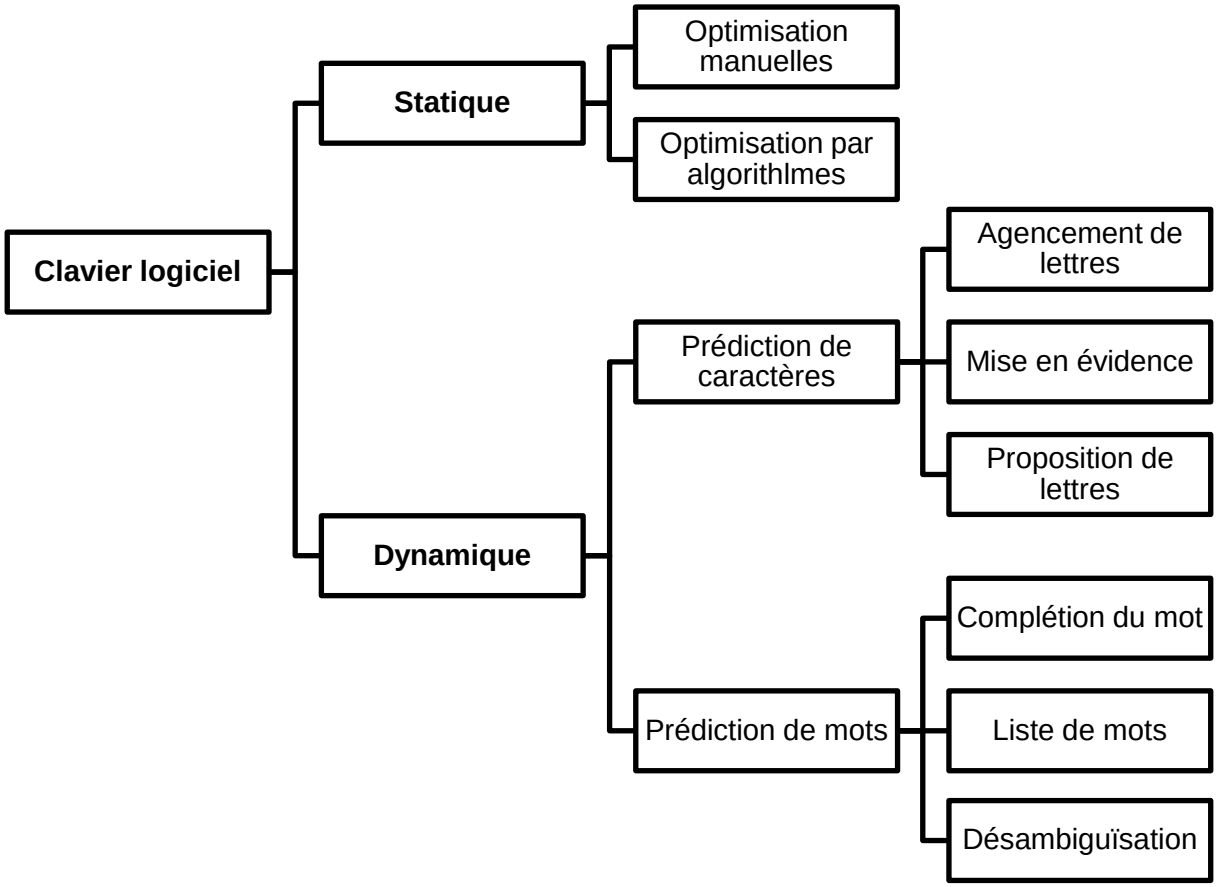


Figure 22: Classification des claviers logiciels par rapport à la presentation des résultats des systèmes de prédiction

Le Tableau 2 représente le bilan de quelques claviers cités. Ces claviers sont classés en fonction de leur système de prédiction, de la manière dont ils en exploitent les résultats et la méthode de conception.

<table>
<tr><th colspan="3" rowspan="2"></th><th colspan="3">Systèmes de prédiction</th></tr>
<tr><th colspan="3">Type d'interaction / Conception</th><th>Statistique</th><th>Syntaxique</th><th>Mixte</th></tr>
<tr><td rowspan="4">Agencement</td><td rowspan="3">Statique</td><td>Intuitive</td><td rowspan="3">• GAG
• Metropolis
• Atomik
• Claviature
• [Cantegrit 2001]
• UKO-II
• WiVik
• DoteNote
• JustType
• TOC
• Frequency</td><td rowspan="3"></td><td rowspan="3"></td></tr>
<tr><td>• Alphabétique
• Chubon
• FITALY
• OPTI
• CATKey</td></tr>
<tr><td>Autre algorithme
• Muti-layer</td></tr>
<tr><td colspan="2">Dynamique</td><td>• Sibylettre
• Sibylle
• Sibyword
• ChewingWord
• FOCL
• SpreadKey</td><td>• Dasher</td><td>Slidekey</td></tr>
<tr><td colspan="3">Mise en évidence</td><td>• BigKey
• FloodKey</td><td>• [Magnien 2003]
• Dasher</td><td></td></tr>
<tr><td colspan="3">Désambiguïsation</td><td>• Claviature
• [Raynal 2005a]</td><td>• T9</td><td></td></tr>
<tr><td colspan="3">Complétion de mot</td><td>• VITIPI
• [Darragh 1999]</td><td>• [Garay-Victoria 1997]</td><td>• Unipad</td></tr>
</table>

Liste de mots et /ou caractères	• LURD-Writer • Keystrokes • PO-Box • PROPHET • UKO-II • Clavicom • The Grid • PolyPredix • CVK • Keyvit • clickNtype • Sibylle • Sibylettre • Sibyword • Fasty	• WordTree • KOMBE	• KeyGlass • SlideKey • [Ménier 2001]

Tableau 2: Tableau représentant la classification des claviers les plus connus

Section III - Synthèse

Initialement, les systèmes de communication assistée ont été développés pour fournir aux personnes à faible motricité des moyens alternatifs leur permettant de saisir du texte. De tels systèmes ont été conçus non seulement pour remplacer les claviers physiques, mais aussi pour accélérer la saisie de ces personnes et améliorer leur performance. Plusieurs tentatives ont été mises en place pour pallier, le plus efficacement possible, l'incapacité motrice de ces utilisateurs. Bien que ces nouvelles conceptions puissent réduire les déplacements, elles peuvent également ajouter des exigences cognitives et perceptives à l'utilisateur. Ces systèmes diminuent le nombre de clics et d'opérations mais présentent des charges additionnelles pouvant avoir des effets indésirables sur la performance globale [Koester 1993]. Les chercheurs ont depuis longtemps reconnu que la réduction de frappes ou des difficultés motrices ne se traduit pas d'une manière égale par une amélioration du taux de texte produit [Gibler 1982], [Dabbagh 1985], [Damper 1984], [Goodenough-Trepagnier 1988], [Vanderheiden 1987]. En effet, le taux global de génération de texte peut diminuer spécialement dans le cas d'une liste de prédiction, à cause du temps perdu dans la sélection et la recherche du mot dans la liste, des processus cognitifs et perceptifs supplémentaires [Gibler 1982], [Horstmann 1990], [Newell 1991], [Horstmann 1991], [Horstmann 1992], [Koester 1994a]. La Figure 23 présente l'amélioration du taux de génération de texte par

rapport à la diminution du nombre d'opérations [Scull 1988], [Gibler 1982]. Un exemple dans [Newell 1991] montre la problématique suivante : pour un utilisateur, le taux de génération augmente de 100% pour une diminution de 58% de frappes. Tandis que chez un autre utilisateur le taux diminue jusqu'à 5% avec 53% d'opérations en moins. Ceci explique que la performance des utilisateurs ne dépend pas seulement de la performance du système mais aussi des charges cognitives additionnelles.

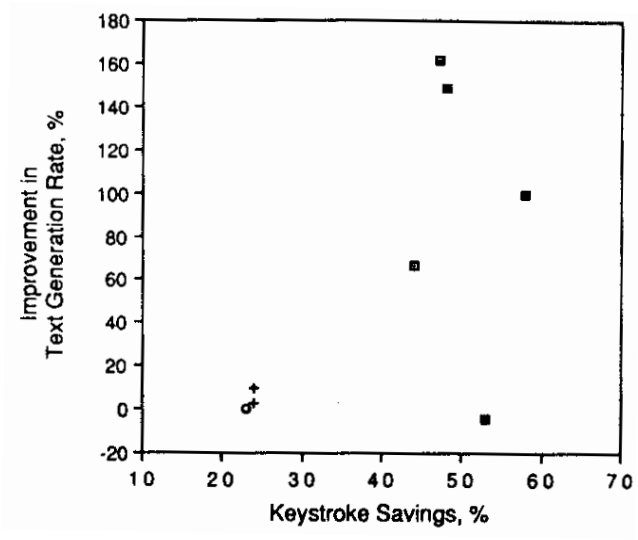


Figure 23: Amélioration du taux de génération de texte par rapport à l'économisation du nombre de frappes

Dans la suite de ce mémoire, nous allons nous focaliser sur l'étude de la liste de prédiction et essayer de comprendre pourquoi l'utilisateur peut y perdre plus de temps qu'il n'en gagne.

Chapitre 2 - Etude du comportement de l'utilisateur face à un système de prédiction

Les travaux de [Horstmann 1992], [Koester 1994a] et bien d'autres chercheurs affirment que malgré la réduction du nombre d'opérations durant la saisie, le taux global de génération de texte n'augmente pas, mais au contraire diminue à cause des charges cognitives additionnelles et la perte de temps dans la recherche du mot dans la liste. Pour confirmer cette hypothèse, nous avons essayé d'étudier le comportement de l'utilisateur saisissant du texte ou des mots avec un clavier logiciel comportant une liste de prédiction.

Avant de présenter notre étude, nous proposons 1) un état de l'art des techniques existantes pour évaluer un système de saisie de texte quel qu'il soit ; 2) une étude des différentes variables existantes qui permettent d'obtenir des informations sur la pertinence d'un système de saisie et de la liste de prédiction qui l'accompagne ; 3) un tour d'horizon des travaux qui ont déjà été effectués sur les listes de prédiction et qui déterminent les facteurs qui peuvent avoir un impact significatif sur la sélection dans une liste.

Section I - Les méthodologies d'évaluation

A la différence des claviers physiques où la disposition des caractères est fixe, ces systèmes présentent des fonctionnalités supplémentaires comme la réorganisation des lettres [Schadle 2001], la proposition de fin des mots [Darragh 1999], ou bien l'affichage d'une liste de prédiction [Masui 1999]. Si des lois prédictives existent pour prédire « *a priori* » le temps nécessaire pour utiliser ces systèmes, elles ne prennent pas (ou peu) en considération la réaction des utilisateurs vis-à-vis du système de saisie, la nature de l'handicap, ou l'effort supplémentaire (physique ou cognitif) nécessaire à l'utilisateur pour saisir du texte. C'est pourquoi, les évaluations expérimentales sont prépondérantes pour l'évaluation des systèmes interactifs dynamiques. Ces expérimentations peuvent porter sur les claviers logiciels et les systèmes de prédiction pour pouvoir évaluer leurs performances. Ces effets

dynamiques peuvent perturber l'utilisateur et par conséquent diminuer les performances du système, ou au contraire réduire la fatigue pour une personne en situation d'handicap [Bérard 2004a]. D'où la nécessité d'une évaluation expérimentale.

I.1. Les variables mesurées

Lors d'une évaluation avec des sujets on cherche à calculer les critères suivants : la vitesse de saisie, le taux d'erreur, le nombre de clics nécessaires pour la sélection d'un caractère (KSPC), le pourcentage de réduction du nombre de saisies (KSR), le taux de prédictions correctes (HR) et le nombre moyen d'appuis avant la prédiction (KUC).

I.1.a. Vitesse de saisie

Pour calculer les performances « *a priori* » d'un utilisateur, Soukoreff et MacKenzie [Soukoreff 1995] proposent deux valeurs représentatives de la variable de vitesse de saisie :

- Le nombre de caractères saisis par seconde (en anglais Characters Per Second (CPS)) : il s'agit du nombre de caractères saisis lors de la tâche d'entrée de texte. Cette vitesse est obtenue en divisant le nombre de caractères (N) par le temps. Le temps de saisie est généralement pris entre le premier caractère à saisir et le dernier caractère. Dans ce cas, le temps est alors divisé par N-1.

- Le nombre de mots qu'il est possible de saisir par minute (en anglais, words per minute (WPM)) en estimant qu'un mot est en moyenne composé de 5 caractères [Gentner 1983]. Ce nombre est obtenu en multipliant par 60 le nombre de caractères par seconde et en le divisant donc par 5.

$$WPM = \frac{CPS \times 60}{5}$$

I.1.b. Taux d'erreur

Durant les expérimentations faites pour évaluer le clavier et le système de prédiction, l'utilisateur peut être sujet à un nombre d'erreurs plus ou moins important. Avant de calculer le taux d'erreur, il faut tout d'abord trouver la distance d'édition.

Distance d'édition

La distance d'édition est la distance qui sépare deux chaînes de caractères A et B. C'est le coût minimal des transformations élémentaires nécessaires pour passer de la chaîne A à une chaîne B. Il existe trois types de transformations :

- **L'insertion** : La chaîne A est obtenue à partir de la chaîne B en ajoutant un caractère supplémentaire. Par exemple : EST → ESTT ;
- **La suppression** : C'est la suppression d'un caractère de la chaîne B par rapport à la chaîne A. Par exemple : EST → ET,
- **Le remplacement** : Cela consiste à remplacer un caractère par un autre dans la chaîne B. Par exemple : EST → ESR;

Cette distance est calculée lors de l'évaluation d'un système de saisie de texte lorsque les utilisateurs ne se soucient pas de corriger les erreurs, et par suite il sera possible de trouver la différence entre la chaîne source et la chaîne recopiée.

Exemple 1 : Voici un exemple de calcul de la distance d'édition repris de [Soukoreff 2001].

Prenons comme chaîne de caractères initiale (**I**) '**abcd**' et imaginons que la chaîne produite (**P**) par le sujet soit '**acbd**'. Il existe 3 manières de transformer de manière minimale la chaîne de caractères **I** en **P** :

- Supprimer le caractère 'c' et insérer un caractère 'c' après le caractère 'b';
- Insérer un caractère 'b' après le caractère 'a' puis supprimer le second caractère 'b' ;

- Remplacer le caractère 'c' par un caractère 'b' puis remplacer le second caractère 'b' par un caractère 'c'.

Chacune des manières présentées nécessite deux opérations pour passer de **I** à **P**. Par conséquent la distance d'édition est de 2.

Cette distance est calculée lors de l'évaluation du système de saisie, et est utilisée pour calculer le taux d'erreur[9] que l'utilisateur fait pendant son entrée de texte.

Taux d'erreur

La distance d'édition entre le texte produit et le texte de référence peut être obtenue par les algorithmes de Levenstein [Levenshtein 1966]. Le calcul du taux d'erreur est fondé sur le calcul de cette distance comme a été proposé par Soukoreff et MacKenzie [Soukoreff 2001]. Ainsi le taux d'erreur sera calculé comme suit :

$$Tx = \frac{MSD(I,P)}{\max(|I|,|P|)} \times 100$$

Avec :

- MSD(I,P), ou « Minimum String Distance », c'est la distance d'édition entre la chaîne initiale **I** et la chaîne produite **P** ;
- max(|I|,|P|), représente la plus grande des deux chaînes.

Exemple 2 : repris de [Raynal 2005a]
Texte source : bonjour tout le monde
Texte produit : bon**kjour** to**u** le mon**f**e

[9] Il est nécessaire de calculer le taux d'erreur pour discuter les erreurs faites lors de la saisie, car le nombre d'erreurs n'a une importance que par rapport au nombre de caractères saisis.

L'utilisateur n'a fait réellement que 3 erreurs en recopiant le texte : il a inséré le caractère 'k' avant le 'j' puis il a oublié le caractère 't' et enfin a remplacé le caractère 'd' par un caractère 'f'. La distance d'édition est de 3. Le nombre de caractères sur chaque chaîne est de 21. Le taux d'erreur de saisie est donc de 14,29%.

D'autre part, MacKenzie [MacKenzie 2002a] calcule le taux d'erreur en se basant sur ce qu'il appelle la taille moyenne des alignements. Il définit un alignement comme étant l'ensemble des transformations nécessaires pour passer de la chaîne A à la chaîne B. La taille d'un alignement est la somme du nombre de caractères qu'il faut normalement saisir et le nombre d'insertions (sans les opérations de suppression et de remplacement) nécessaires pour obtenir la nouvelle chaîne. Le nouveau taux d'erreur sera alors calculé comme suit :

$$Tx = \frac{MSD(I,P)}{\overline{S_A}} \times 100$$

Avec :

- MSD(I,P), la distance d'édition entre la chaîne initiale **I** et la chaîne produite **P** ;
- $\overline{S_A}$, représente la taille moyenne des alignements, calculée en faisant la moyenne des tailles d'alignements qui donnent une distance d'édition minimum.

KSPC

Parfois, dans les évaluations, le sujet essaie de corriger ses erreurs. Il utilise alors d'autres touches pour supprimer les fautes comme la touche de « retour arrière » ou de « suppression », etc. D'autre part, Il existe des exercices d'évaluations où des systèmes de saisie n'acceptent que les caractères correctement saisis. Le sujet doit retaper la lettre jusqu'à ce que le système reçoive le bon caractère. Dans ces cas, on ne peut pas calculer le taux d'erreur en utilisant les formules précédemment décrites. On a recours alors à une autre métrique : le KSPC ou le nombre de frappes par caractère (en anglais, « KeyStrokes Per Character ») [MacKenzie 2002b].

Ce calcule consiste à compter le nombre de touches que l'utilisateur a frappées durant la saisie. Les erreurs de frappe sont ainsi prises en compte dans les touches frappées ainsi que les touches qui ont été utilisées pour supprimer ces erreurs (« retour arrière », « suppr » etc.). Le KSPC est obtenu alors par :

$$KSPC = \frac{Nb_F}{|T|}$$

Avec :

- Nb_F : le nombre de touches réellement frappées.
- | T | : le nombre de caractères du texte à saisir.

Exemple 3 :

Texte source : voiture

Texte saisi : vo**u**←**i**ture

Texte produit : voiture

Dans cet exemple, l'utilisateur a commis une faute qu'il a corrigée et par suite deux touches supplémentaires (la touche de retour et la lettre correcte) ont été saisies pour que le texte produit soit homologue au texte source.

Si les utilisateurs ne corrigent pas leurs erreurs alors KSPC est égale à 1 et le taux d'erreur sera positif. Inversement, dans le cas de la correction, le taux d'erreur aura une valeur nulle et la valeur de KSPC sera supérieure à 1. Dans ce cas, le texte ou les caractères saisis peuvent être décomposés en quatre types [Soukoreff 2003] :

- Correct (C) : qui représente l'ensemble des caractères correctement saisis ;
- Incorrect and Not Fixed (INF): qui regroupe l'ensemble des caractères incorrects et non corrigés saisis par le sujet ;
- Incorrect but Fixed (IF) : qui constitue les caractères erronés et corrigés ;
- Fixed (F) : les touches de caractères utilisées pour corriger une erreur.

En se basant sur cette décomposition, deux nouvelles mesures ont été définies :

$$\mathrm{MSD} = \frac{INF}{C + INF} \times 100$$

$$KSPC = \frac{C + INF + IF + F}{C + INF}$$

CPC

CPC, « Cost Per Correction » ou le coût de la correction, est une nouvelle métrique pour évaluer les erreurs lors de la saisie de texte. Cette métrique a été proposée par Gong et Tarasewich [Gong 2006] pour calculer le nombre de corrections que l'utilisateur commet durant une entrée de texte. CPC est défini suivant la formule :

$$CPC = \frac{WLS + WCS + FS}{NoC}$$

Où :

- WLS, « Wasted Letter KeyStrokes » ou les frappes fausses et corrigées, similaire à IF (Incorrect but Fixed).
- FS, « Fixing KeyStrokes » ou les touches utilisées pour la correction, identique à F (Fixed).
- WCS, « Wasted Control KeyStrokes », ou les sélections et les touches utilisées par la personne mais que n'appartient pas à l'ensemble du texte, comme par exemple la touche « next » dans le cas d'une saisie avec prédiction.
- NoC, « Number of Correction », c'est le nombre total de corrections commises durant la tâche de saisie.

Cette métrique prend en considération les frappes gaspillées et les touches de correction relatives aux erreurs corrigées et par suite ne dépend pas du nombre de caractères correctement saisis.

I.1.c. Variables pour évaluer la pertinence de la prédiction

KSR

Le taux d'économie de saisie (KSR pour *Keystroke Saving Rate*) exprime le pourcentage moyen d'appuis qui sont évités grâce à la prédiction. Il est calculé pour une taille de liste de prédictions donnée. Le KSR évalue alors le taux d'utilisation de touches normales pour saisir le texte même avec l'existence du système de prédiction. Cette métrique concerne les appuis et non pas les lettres non saisies ce qui peut être un défaut dans l'évaluation de la prédiction puisque la stratégie de prédiction dans certains systèmes peut nécessiter des appuis supplémentaires, par exemple pour atteindre la liste de prédictions.

KSR est défini par la formule suivante :

$$KSR = \frac{keys_{normal} - keys_{with\ prediction}}{keys_{normal}} * 100$$

Où :

- $Keys_{normal}$ désigne le nombre de touches que l'utilisateur clique sur un clavier normal pour saisir un certain mot
- $Keys_{with\ prediction}$ est le nombre de touches cliquées en considérant les lettres prédites.

HR

HR ou le taux de prédictions correctes (HR pour « Hit Rate ») [Fazly 2003] évalue le pourcentage de fois où le mot recherché est proposé dans la liste de prédictions. Ce taux dépend aussi de la taille de la liste de prédictions. Dans le cas idéal, chaque fois que le mot cible apparaît dans la liste, l'utilisateur le sélectionne. La formule du HR peut être écrite comme suit :

$$HR = \frac{Nombre\ de\ mots\ qui\ apparaissent\ dans\ la\ liste}{Nombre\ de\ mots} \times 100$$

KUC

Une autre métrique peut être utilisée dans l'évaluation des logiciels de saisie et des claviers associés à des systèmes de prédiction : la KUC (ou Keystrokes Until Completion en anglais). Cette métrique calcule le nombre moyen d'appuis avant la prédiction. Dans le cas d'une liste de mots prédits, cette technique évalue le nombre moyen de touches ou de lettres que l'utilisateur doit saisir manuellement avant que le système ne prédise la prédiction du mot et la propose à l'utilisateur.

I.2. Les tâches

La prédiction de mots associée à un clavier logiciel vise à accélérer un peu plus la vitesse de saisie de texte et éviter le maximum de saisies. Les métriques déjà citées aux paragraphes précédents sont utilisées pour évaluer la qualité du système. Dans notre cas, le système est un clavier logiciel allié à une liste de prédiction. Cette liste se met à jour après chaque touche d'où nous pouvons la définir comme 'dynamique'. Or, les événements dynamiques qui se produisent sur le système peuvent perturber l'utilisateur et lui fait perdre sa concentration, ou au contraire réduire l'effet de la fatigue pour une personne ayant un handicap moteur au niveau des membres supérieurs [Bérard 2004b]. Il est alors indispensable d'évaluer les

systèmes avec les utilisateurs. Mais quelle procédure doit-on suivre dans les expérimentations pour que ce soit une bonne estimation ?

Lors d'une évaluation avec des sujets, l'activité de saisie de texte peut consister en :

- une recopie d'un texte ou d'un ensemble de mots,
- une saisie libre d'un texte de leur propre imagination.

I.2.a. La tâche de recopie

Le focus d'attention

MacKenzie dans [MacKenzie 2002c] définit le focus d'attention par le nombre de points nécessitant une attention visuelle de l'utilisateur pour accomplir la tâche. Un focus d'attention est, par exemple, le cas d'un utilisateur expert sur un clavier physique et qui ne regardera que le texte source à recopier : on parle alors de simple focus d'attention. D'autre part, l'utilisateur est obligé de regarder à la fois le clavier logiciel et le texte à recopier dans le cas de l'utilisation d'un clavier logiciel avec un stylet ; on parle alors de double focus d'attention.

Ainsi, durant une expérimentation, plus le nombre de focus est faible, plus les sujets pourront se concentrer sur leur saisie. Par conséquent, la manière de présenter le texte source a une grande importance pour diminuer le focus d'attention et ainsi augmenter les performances de saisie.

Que faire recopier ?

Dans cette tâche, nous demandons à l'utilisateur de recopier un texte, des phrases courtes ou un ensemble de mots à l'aide du système de saisie mis en test :

- **Des mots isolés :** dans cette tâche les utilisateurs sont demandés de copier un ensemble de mots isolés. Dans ce cas, les utilisateurs peuvent mémoriser le mot avant de le sasir, ce qui peut diminuer le focus d'attention en ne regardant le mot qu'une seule fois. Cependant, l'utilisateur ne peut pas connaître le mot suivant ce qui

peut intercepter la saisie. Cette méthode a été utilisée dans [Badr 2009], [Magnien 2004], [Zhai 2003].

- **Des phrases courtes :** Une autre alternative consiste à faire recopier des phrases courtes. On trouvera dans [MacKenzie 2003] une série de phrases courtes en anglais pour être saisies lors de l'évaluation d'un clavier logiciel. Ces phrases ont l'avantage de respecter la fréquence d'apparition des caractères et des bigrammes, permettant à l'utilisateur de saisir la plupart des caractères dans une même phrase. Cette alternative a été le sujet dans les évaluations de [Raynal 2005b], [Badr 2011].

- **Un texte entier :** D'autres auteurs préfèrent de faire copier les utilisateurs des textes plus longs comme par exemple dans [Ward 2000] ou [Matias 1996]. Cette méthode peut être plus lourde introduisant un biais dans l'estimation de temps de saisie. Cependant, elle permet de voir l'évolution des performances et de caractériser la fatigue de l'utilisateur.

La présentation du texte à recopier

Le texte source (ou le texte à recopier) est généralement affiché à l'écran. Dans ce cas le problème qui se pose est celui du *feedback*[10] *:* où afficher le *feedback* et comment ?

Dans le cas des mots isolés et des phrases courtes, [Magnien 2004] et [Pavlovych 2005] affichent le *feedback* juste en dessous du mot saisi par l'utilisateur ce qui réduit le focus d'attention à un puisque l'utilisateur peut en même temps voir le texte et le *feedback.*

[10] Par définition, le feedback est une rétroaction. C'est une conséquence de la réponse fournie par le sujet. Un feedback donne de l'information au sujet de l'action qu'il a faite. Dans le contexte d'expérience de saisie de texte, le feedback informe l'utilisateur en cas d'une entrée erronée. Le feedback informe aussi l'utilisateur en cas de saisie exacte en affichant la lettre saisie à l'écran

Le problème se pose dans l'affichage du *feedback* dans le cas de recopie d'un texte où l'utilisateur ne peut pas voir dans un même regard ce qui a déjà écrit et ce qu'il doit saisir. Cela aboutit à une augmentation du focus d'attention à deux. Une solution vise à intercaler le texte de l'utilisateur entre chaque ligne du texte source comme proposé dans [Matias 1993] et [Matias 1996], (cf. Figure 24).

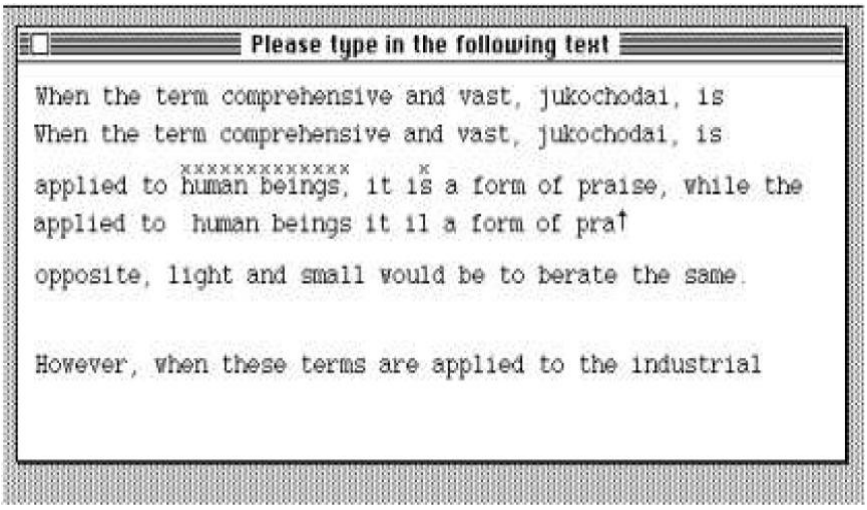


Figure 24: Interface de présentation du texte à recopier (repris de [Matias 1996]**)**

Une autre solution est proposée par [Vigouroux 2004] qui vise à limiter la recherche visuelle de l'utilisateur en n'affichant que les 4 ou 5 prochains caractères et à supprimer au fur et à mesure le caractère qui vient d'être saisi. Le problème du focus d'attention est aussi réduit par le fait de dicter au sujet ce qu'il doit saisir [Ward 2000]. Cette solution peut être stressante pour l'utilisateur et il est difficile de savoir si les erreurs commises sont dues à un problème d'interaction avec le système de saisie ou s'il s'agit d'une mauvaise compréhension de la dictée.

Le *feedback* tactile proposé par Dunlop et Taylor [Dunlop 2009] alerte l'utilisateur en train de saisir les caractères et attire son attention en faisant vibrer son dispositif ou téléphone mobile. Deux types de vibration incitent l'utilisateur à regarder l'écran. Une courte vibration (75 ms) indique qu'un mot ou liste de mots sont prédits et sont disponibles pour compléter son préfixe ce qui pourrait réduire le temps de saisie. Une longue vibration (150 ms) alerte

l'utilisateur que le mot qu'il est en train de taper n'existe pas dans le dictionnaire et qu'une erreur de saisie risque d'avoir lieu.

I.2.b. La tâche de création

Dans cette tâche, l'utilisateur n'a pas de texte à recopier mais utilise son imagination pour écrire le texte. La tâche de création présente comme principal avantage d'être plus proche de l'utilisation normale du système. Cette technique est surtout utilisée pour connaître les performances du système de prédiction en premier lieu et non celles du système de saisie [Wester 2003]. Cependant, peu d'études utilisent ce type de tâche pour les désavantages que présente cette technique [MacKenzie 2002c] :

- Dans cette tâche, on ne peut pas contrôler le comportement de l'utilisateur et ses hésitations. Par conséquent, il devient difficile de savoir si la vitesse de saisie est optimale ou s'il y a eu des hésitations. Ce qui a pour effet de ralentir la saisie.

- Il est difficile de contrôler les erreurs : est-ce des erreurs d'interaction ou des fautes d'orthographe et de grammaire commises par l'utilisateur ?

- Il est difficile de contrôler le texte saisi par l'utilisateur et par conséquent sa représentativité des fréquences d'apparition des caractères dans la langue. Ce qui rend presque impossible la généralisation des résultats.

Section II - Précédentes études sur les listes de prédiction

Plusieurs facteurs jouent un rôle déterminant dans l'acceptabilité d'une liste de prédiction ainsi que dans la performance des utilisateurs [Horstmann 1991], [Newell 1992], [Soede 1986]:

- Les caractéristiques des utilisateurs ;
- Les performances et l'efficacité de l'ordinateur ;

- L’économie du nombre d’opérations et des charges visuelle et cognitive.

D’après [Soede 1986], avec l’avancement des technologies on peut négliger le facteur concernant les performances de l’ordinateur. D’autre part, les caractéristiques des utilisateurs sont considérées comme relativement constantes, alors l’efficacité globale d’un système de prédiction de mots dépend du dernier facteur [Soede 1986], [Vanderheiden 1987]. Les efforts cognitifs relatifs à l’utilisation des systèmes de prédiction de mots dépendent des décisions prises par l’utilisateur concernant le moment de recherche dans la liste, le nombre de mots candidats proposés au sujet, la position, forme et ordre de la liste et le temps de recherche.

II.1. Nombre de propositions

Un des facteurs les plus importants et qui peut rendre la liste plus utilisable est le nombre de propositions affichées par le système. La plupart des travaux visant ce sujet mentionnent que plus le nombre de propositions est large, plus les résultats de la prédiction sont meilleurs, et plus le pourcentage de réduction du nombre de clics est grand (Figure 25), [Swiffin 1987], [Koester 1998], [Garay-Victoria 1997], [Copestake 1997].

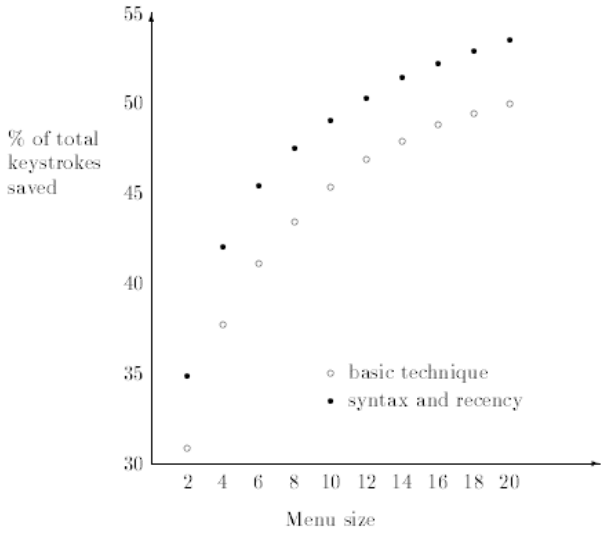


Figure 25: Réduction du nombre d'opérations avec l'augmentation de la taille de la liste [Copestake 1997]

Cependant, un grand nombre de propositions du système de prédiction augmentera le temps de recherche et l'effort cognitif nécessaire pour sélectionner un candidat. Landauer et Nachbar [Landauer 1985] ont étudié l'effet de la longueur de la liste et le nombre de propositions dans celle-ci. Les chercheurs ont trouvé que le temps de recherche dans une liste de 5 mots prend entre 1.0 et 1.5 secondes et augmentera d'une façon logarithmique si d'autres mots candidats sont ajoutés. Swiffin et ses collègues [Swiffin 1987] ont mené une simulation pour proposer un nombre optimal de mots dans une liste. Ils ont trouvé qu'une liste de 5 mots peut économiser le nombre de frappe tout en minimisant la charge et l'effort cognitifs. Inversement, [Venkatagiri 1999] a conduit une étude empirique sur l'effet du nombre de propositions sur le taux de saisie et conclut qu'une liste de 15 mots peut réduire au minimum le nombre d'opérations mais augmentera significativement le temps de recherche d'un mot spécifique. Koester et Levine [Koester 1998] ont montré que pour chaque proposition ajoutée, le temps de recherche d'un mot déterminé dans la liste augmentera de 150 ms. La lecture et le parcours de la liste, surtout pour les personnes souffrant d'un handicap, peut poser un problème et augmente la charge de l'utilisateur qui doit toujours déplacer son regard et sa tête entre le clavier et la liste. Le temps de ce déplacement, estimé par Koester et Levine de 0.8 s pour chaque option, peut alors diminuer d'une manière significative la vitesse de saisie du message [Koester 1996]. Pour cette

raison, différents auteurs ont visé de minimiser autant que possible la distance entre le clavier et la liste de prédiction [Newell 1992]. D'autre part, malgré la performance présentée dans le cas d'un grand nombre de propositions et malgré les expérimentations menées sur ce cas (par exemple jusqu'à 20 mots sont proposés dans [Copestake 1997]), les chercheurs ne présentent généralement que des listes contenant entre cinq et sept mots. En effet, certains auteurs comme [Heinisch 1993], [Newell 1992], [Swiffin 1987] ont estimé que c'est le nombre qu'un utilisateur peut percevoir dans un seul regard.

II.2. Distribution et position des propositions

Séparément du nombre de propositions, la distribution et la présentation de ces propositions peuvent aussi jouer un rôle dans la vitesse du processus de sélection. En effet, les utilisateurs doivent toujours transférer leur regard entre le clavier et la liste de prédiction. Dans la tâche de copie, le changement du regard nécessite aux utilisateurs un effort additionnel [Anson 1993], [Treviranus 1987], [Vanderheiden 1987].

La position de la liste est considérée comme un des facteurs affectant les charges visuelle et cognitive. [Swiffin 1987] et d'autres auteurs ont proposé que la liste de mots doit être présentée à proximité du curseur d'écriture, pour permettre à l'utilisateur de percevoir à la fois le texte et les mots candidats. En projetant cette proposition aux claviers logiciels, [Masui 1999] a conçu un système de saisie, le PO-Box de façon à ce que la liste de mots soit affichée au même endroit de la dernière saisie. A ce jour, il n'existe pas d'étude empirique qui compare l'effet des différentes positions sur le taux de saisie.

Concernant la présentation des mots, les auteurs de [Klund 2001], [Newell 1992] et [Swiffin 1987] ont démontré que les efforts de recherche et de sélection dans une liste verticale sont inférieurs à ceux commis en utilisant une liste horizontale. En fait, la recherche de mots dans une liste horizontale nécessite des mouvements additionnels de tête et de regard pour balayer toutes les propositions. Quand ils sont exposés verticalement, la longueur et la forme des mots peuvent fournir quelques indices pour orienter la recherche visuelle [Swiffin 1987].

Ajoutons que l'ordre dans lequel les mots sont présentés peut être important dans l'efficacité du système de prédiction. Les propositions peuvent être classées par ordre alphabétique [Sad 2009a], ou suivant leur taille, ou par ordre de fréquence d'apparition [Woltosz 1990]. La règle la plus fréquente est de trier les propositions par ordre de probabilité. L'avantage de cette présentation est la réduction de temps de recherche comme ayant les mots les plus probables en haut de la liste. D'autre part, la taille des mots peut guider les utilisateurs dans leur recherche. Mais, quand le nombre de propositions est grand, le classement par ordre alphabétique est plus performant [Garay-Victoria 2006]. Cette dernière technique est plus fréquemment utilisée avec des utilisateurs ayant des difficultés d'apprentissage[11]. Quelques utilisateurs préfèrent que les propositions restent toujours à leur place, nécessitant ainsi moins d'effort pour les mémoriser [Koester 1998].

II.3. Synthèse

Ces nombreux travaux effectués sur les listes montrent l'importance de la position de la liste à l'écran, la taille de l'espace d'affichage et le nombre d'items dans la liste. Cependant, si ces travaux permettent d'optimiser l'utilisation de la liste en faisant varier ces paramètres, ils n'expliquent pas pourquoi l'utilisation de la liste reste préjudiciable pour l'utilisateur dès que celui-ci a une vitesse de saisie un peu rapide sur le clavier logiciel.

Notre étude porte sur l'influence de la liste quant à la saisie sur clavier logiciel et s'il est possible de dégager des règles sur le comportement de l'utilisateur durant sa saisie. Pour notre étude, nous avons choisi de prendre une liste de prédiction avec les caractéristiques qui lui sont normalement favorables d'après les études que nous venons de présenter. Ainsi notre liste de prédiction est une liste verticale de sept items où les mots y sont classés par rapport à leur fréquence d'apparition. De plus, pour effectuer notre étude, nous avons donc choisi de comparer la saisie de texte au moyen d'une liste de prédiction avec celle effectuée sur un clavier logiciel classique.

[11] Auron [Computer Software], Burnaby. BC: Aurora Systems. Inc., 1999.

Section III - Protocole d'évaluation

III.1. Participants

Huit personnes (six hommes et deux femmes) ont participé à cette expérimentation. Agés de 21 à 43 ans, les participants sont tous des utilisateurs réguliers de l'ordinateur et utilisent quotidiennement le clavier AZERTY. Ils n'ont pas pour habitude d'utiliser un clavier logiciel ni une liste de prédiction.

III.2. Materiel utilisé

L'expérience s'est déroulée dans le laboratoire des usages de l'IRIT. Elle a nécessité l'utilisation de deux ordinateurs de bureau. Le premier sert à exécuter l'expérimentation il est celui avec lequel l'utilisateur interagit. Cet ordinateur est un HP Compaq doté d'un processeur AMD Athlon(tm) 64 X2 4200+ 2,1GHz et d'une mémoire de 2048Mo et a Windows XP 32bit comme système d'exploitation. Les participants utilisaient une souris optique Logitech pour interagir avec le clavier logiciel. Le tout était affiché sur un écran DELL de 19 pouces avec une résolution de 800 pixels par 600 pixels et un rafraichissement de 160 Hz.

Le second ordinateur est utilisé juste pour piloter le système de suivi du regard et sauvegarder les données issues de ce système de suivi. Cet ordinateur Windows98 avec un processeur AMD Duron(tm) 650MHz et d'une mémoire de 128Mo lancé en mode Dos. Le système de suivi du regard utilisé est EyeLink2. Il utilise deux petites caméras positionnées à proximité des yeux de l'utilisateur. Pour faciliter le suivi, l'utilisateur doit positionner sa tête sur une mentonnière et appuyer son front contre une barre (cf. Figure 26).

L'expérimentation se déroule au moyen de la plate-forme E-Assiste. L'ensemble des outils logiciels utilisés (bandeau de présentation, clavier, liste de prédiction) sont implémentés en Java 6. La liste de prédiction a été paramétrée de manière à afficher 7 mots au maximum

dans celle-ci. Seuls des mots de plus de 6 caractères y sont affichés. Enfin, les mots affichés sont ordonnés par rapport à leur fréquence d'apparition (et non par ordre alphabétique).

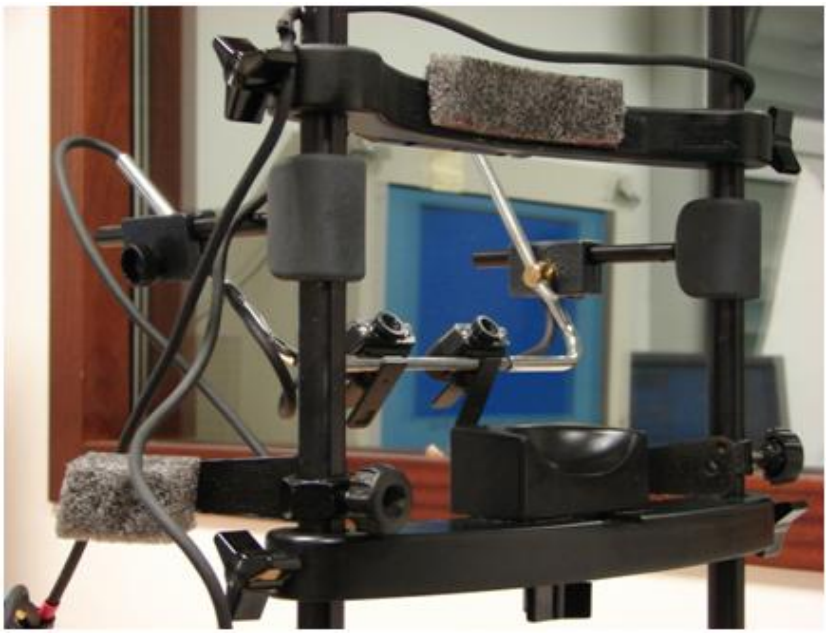


Figure 26 : Dispositif de suivi du regard

III.3. Tâche et stimulii

La tâche demandée aux participants était de recopier le plus rapidement possible et en faisant le moins d'erreurs possibles le mot qui était affiché en haut de l'écran. Pour accéder au premier mot à saisir, l'utilisateur devait appuyer sur la barre espace du clavier logiciel. A la fin de chaque mot, il devait appuyer sur cette même barre espace afin de valider sa saisie et passer au mot suivant. L'utilisateur avait face à lui, sur l'écran, le clavier centré verticalement sur l'écran et le bandeau d'affichage au-dessus du clavier (cf. Figure 27). Le bandeau est divisé en deux parties : sur la partie la plus haute est affiché le mot que l'utilisateur doit saisir ; sur la partie inférieure du bandeau sont affichés les caractères déjà saisis par l'utilisateur. Si ce dernier saisit un caractère erroné, celui-ci n'est pas affiché et l'utilisateur est averti de sa faute à la fois par un feedback visuel rouge et un feedback sonore. L'utilisateur doit alors saisir le bon caractère avant de pouvoir continuer sa saisie.

Un exercice consistait à saisir 40 mots. Les mots ont été choisis de manière à être affichés sur la liste de prédiction à différents moments de la saisie du mot et apparaître dans

différentes positions dans la liste. Ainsi, les deux variables que nous avons pris en compte pour choisir ces mots sont d'une part, le nombre de caractères qu'il faut saisir sur le clavier avant que le mot désiré apparaisse dans la liste de prédiction et d'autre part, la position à laquelle le mot apparait dans la liste. Nous avons donc choisi des mots qui apparaissent après 2, 4, 6 et 8 caractères saisis et aux première, troisième, cinquième et septième positions dans la liste. Pour chacun de ces couples « nb de caractères » / « position », nous avons choisi deux mots. En plus de ces 32 mots, nous en avons choisi huit de plus qui n'apparaissent pas du tout dans la liste.

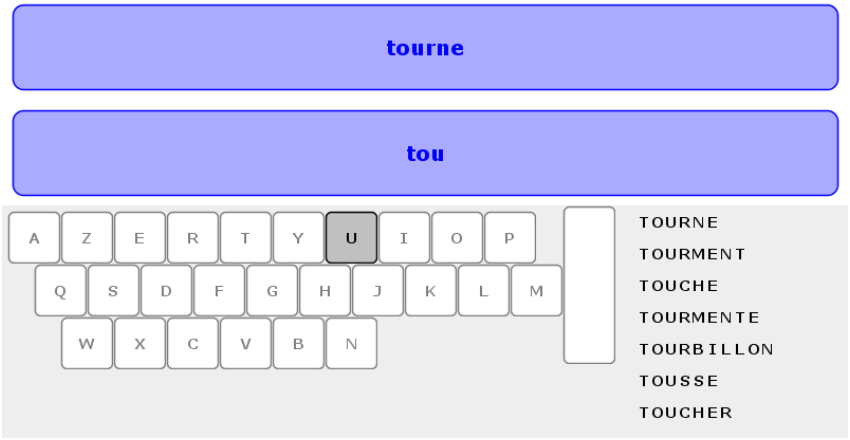


Figure 27 : Plateforme utilisée pour l'expérimentation

III.4. Procédure

Les huit participants ont eu deux exercices à réaliser. Chaque exercice correspondait à une tâche de recopie de 40 mots. Un exercice était réalisé avec le clavier logiciel AZERTY seul et l'autre exercice était effectué avec le même clavier augmenté de la liste de prédiction. Les participants ont été répartis dans deux groupes de quatre. Un groupe commençait avec le clavier seul puis réalisait ensuite l'exercice avec la liste, alors que l'autre groupe réalisait les exercices dans l'ordre inverse de manière à éviter les effets d'apprentissage des mots ou encore la fatigue.

Avant chaque exercice, les participants étaient soumis à une phase de calibrage du système de suivi du regard. Puis tous les 8 mots saisis, 5 pastilles s'affichaient successivement sur

un fond noir. L'utilisateur devait les fixer tour à tour de manière à recalibrer rapidement le système de suivi du regard.

III.5. Données recueillies

La plate-forme E-Assiste permet de recueillir toutes les données relatives à l'interaction entre l'utilisateur et la plate-forme. Ainsi, sont stockés dans un fichier les événements relatifs aux déplacements de la souris, aux pressions et relâchements des boutons de la souris, aux caractères saisis au moyen du système de saisie (clavier et liste de prédiction). La plate-forme permet aussi de sauvegarder d'autres données telles que les mots affichés sur la liste de prédiction, les mots qui sont demandés de recopier à l'utilisateur ou encore le début et la fin d'un exercice. L'ensemble de ces données est conservé dans un format XML décrit dans [Raynal 2005a].

Les données recueillies au moyen du suivi du regard sont stockées dans un autre fichier. Elles décrivent les fixations et les saccades oculaires. De la même manière que le premier fichier, ces données sont stockées dans un format XML qui permet ainsi de les parser et les analyser rapidement.

Section IV - Analyse des données

Les résultats présentés dans cette section ont été obtenus par des tests paramétriques d'analyse de variance. Nous avons utilisé l'outil ANOVA de Scott Mackenzie[12].

Avant de présenter les résultats, nous pouvons déjà dire que les groupes et l'ordre dans lesquels ont été réalisés les exercices n'ont pas eu d'effet significatif sur les résultats ($F_{1,6}$=2,496 et p>0,1).

[12] http://www.yorku.ca/mack/RN-Anova.html

IV.1. Erreur

La Figure 28 présente le taux d'erreur de chaque participant avec chaque dispositif. Une erreur est comptabilisée à chaque fois que le participant saisit un caractère autre que celui qu'il doit entrer. Le taux d'erreur est obtenu en divisant le nombre d'erreurs de saisie par le nombre de caractères à saisir. Tous les taux d'erreurs sont inférieurs à 3%, ce qui implique que le nombre d'erreurs est assez faible et n'aura pas d'impact sur les résultats présentés par la suite. En moyenne, les participants ont réalisé 1,19% d'erreurs avec le clavier seul et 1,08% avec le clavier augmenté de la liste de prédiction. Cette différence entre les deux dispositifs de saisie de texte n'est pas significative ($F_{1,6}$<1).

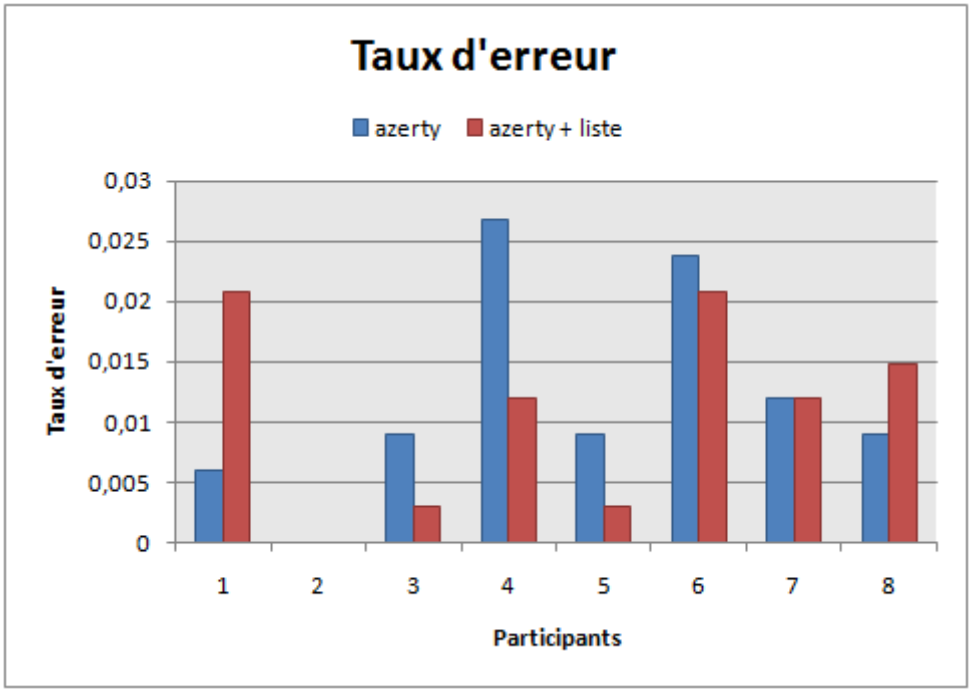


Figure 28 : Taux d'erreur

IV.2. Taux d'utilisation de la liste

Le principal avantage d'une liste de prédiction est de permettre de réduire le nombre d'actions à réaliser pour saisir du texte. Les participants de notre expérimentation ont bien

utilisé cet avantage (cf. Figure 29). En effet, ils ont effectué en moyenne 0,885 actions pour saisir un caractère. Celui qui l'a utilisé le plus n'a effectué en moyenne que 0,82 actions pour saisir un caractère alors que le participant ayant le moins utilisé la liste n'a réalisé que 0,93 actions par caractère saisi.

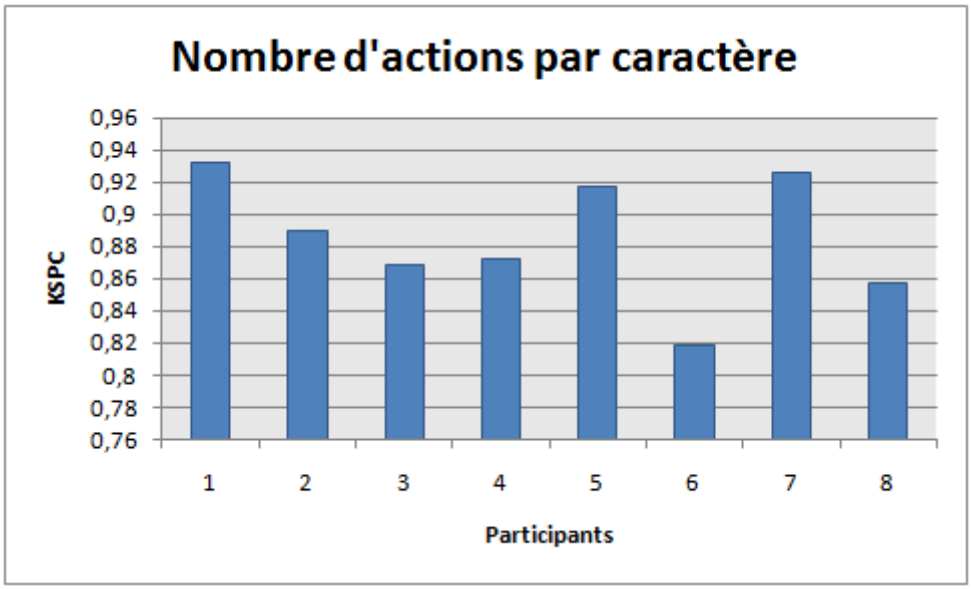


Figure 29 : Nombre d'actions effectuées

IV.3. Vitesse de saisie

En premier lieu, nous avons souhaité confirmer notre hypothèse principale qui est que nous sommes meilleurs avec le clavier AZERTY seul qu'avec le clavier accompagné de la liste. La Figure 30 présente deux mesures de vitesse :

- La première mesure concerne la durée moyenne pour saisir un caractère. Cette mesure est calculée en divisant la durée totale pour saisir l'ensemble des mots par le nombre total de caractères à saisir.
- La seconde mesure présente la durée moyenne pour effectuer une action. Une action est ici définie comme étant une frappe sur une touche du clavier ou la sélection d'un item de la liste. La durée moyenne pour effectuer une action est donc

la durée totale pour saisir l'ensemble des mots divisé par le nombre total d'actions effectuées.

Pour les deux mesures, la durée totale pour saisir l'ensemble des mots est obtenue en sommant l'ensemble des durées de saisie des mots. La durée pour saisir un mot est mesurée entre le moment où le premier caractère du mot est saisi et le moment où le dernier caractère est saisi. Ainsi le temps séparant la saisie de deux mots n'est pas pris en compte.

A noter que la durée de saisie d'un mot étant prise entre le premier et le dernier caractère du mot, nous avons considéré une taille de n-1 caractères pour un mot de n caractères[13]. De même, nous n'avons pris en compte que n-1 actions pour un mot qui en avait nécéssité n.

La première mesure confirme notre hypothèse : les participants ont été plus rapides avec le clavier seul qu'avec le clavier et la liste. En effet, ils ont saisi en moyenne un caractère toutes les 815 millisecondes contre une durée de 911 millisecondes pour saisir un caractère avec la liste. Cet écart entre les deux dispositifs est par ailleurs significatif ($F_{1,6}$=21,987 avec p<0,005).

[13] http://www.yorku.ca/mack/RN-TextEntrySpeed.html

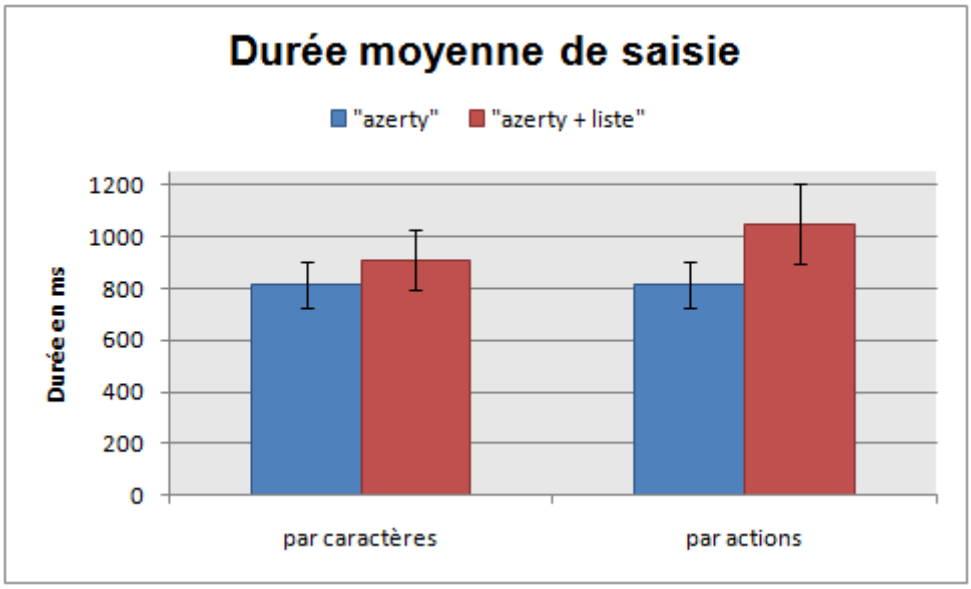


Figure 30 : Comparaison des vitesses obtenues avec les deux claviers

Ce résultat peut paraître troublant car, comme nous l'avons vu précédemment, le clavier augmenté de la liste permet aux utilisateurs de diminuer le nombre d'actions à réaliser pour saisir un caractère. Cette réduction du nombre d'actions ne s'est donc pas traduite par une diminution de la durée de saisie.

La seconde mesure de la Figure 30 montre ainsi l'écart qui sépare la durée pour effectuer une action sur le clavier seul (815 ms) de la durée pour effectuer une action sur le clavier avec la liste (1051 ms). Cet écart est aussi significatif ($F_{1,6}$=136,113 avec p<0,001). A noter que la durée pour effectuer une action sur le clavier seul est la même que celle pour saisir un caractère. Ceci est normal puisque chaque action entraîne la saisie d'un seul caractère. Si l'utilisateur met plus de temps à saisir avec le clavier et la liste c'est donc avant tout car il lui faut beaucoup plus de temps pour effectuer une action avec ce dispositif. L'utilisateur met en moyenne 236 ms en plus, soit 29% de temps supplémentaire par rapport au clavier AZERTY seul.

IV.4. Distance

Une première explication de cette augmentation du temps avec le clavier AZERTY et la liste pourrait être la distance parcourue par le pointeur du dispositif de pointage. De la même manière que pour le temps, nous avons donc calculé pour les distances parcourues deux distances moyennes (cf. Figure 31). D'une part, la distance moyenne parcourue par caractère ; et d'autre part, la distance moyenne parcourue pour effectuer une action.

Les résultats montrent que la distance moyenne parcourue pour effectuer une action est plus élevée lorsqu'il s'agit d'effectuer cette action sur le clavier complété de la liste (147,87 pixels) que pour réaliser une action sur le clavier seul (123,96 pixels). Cette différence de 24 pixels est significative ($F_{1,6}$=87,502 avec p<0,001).

Cependant, cet écart ne peut pas suffire à expliquer seul la différence de temps exposée précédemment. En effet, le dispositif avec la liste permet d'effectuer moins d'actions pour saisir un caractère ; cette distance moyenne par action est réalisée moins souvent qu'avec le clavier AZERTY seul. Si on calcule la distance parcourue en moyenne pour saisir un caractère, on peut s'apercevoir que les participants effectuent quasiment la même distance pour saisir un caractère (123,96 pixels avec le clavier seul contre 128,39 pixels pour le clavier avec la liste. $F_{1,6}$=8,044 avec p<0,05).

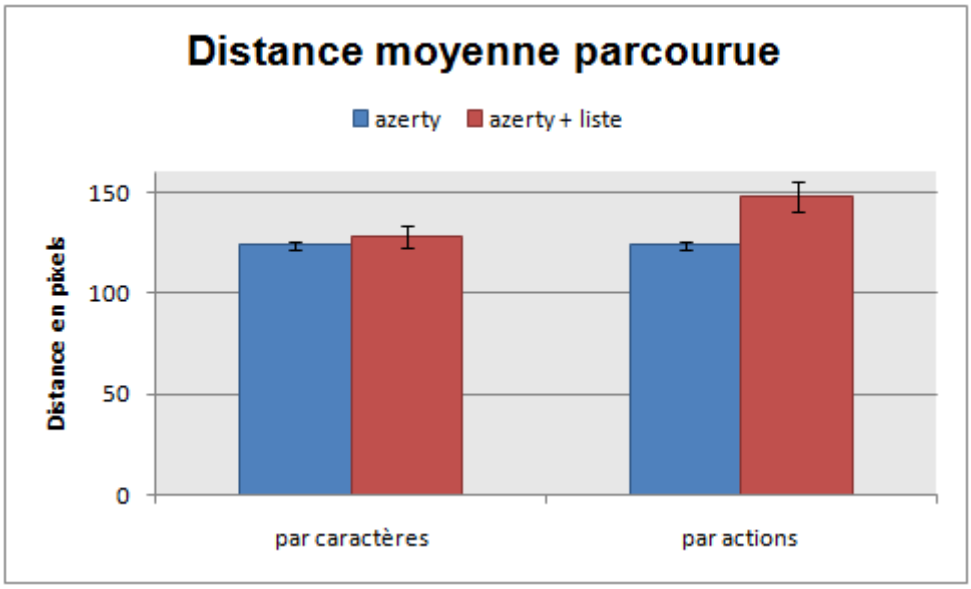


Figure 31 : Distance parcourue

En conclusion de cette analyse classique, nous pouvons donc confirmer le constat général qui est que la liste de prédiction ralentit la saisie sur un clavier logiciel de type AZERTY. Ce ralentissement n'est pas du à une augmentation du nombre d'erreurs, ni à une augmentation des distances à parcourir avec le pointeur du dispositif de pointage. Afin d'essayer de comprendre l'origine de ce problème, nous allons maintenant proposer une analyse du comportement oculaire de l'utilisateur.

IV.5. Impact de la présence de la liste sur la saisie sur clavier logiciel

IV.5.a. Comparaison des temps moyens de saisie d'un caractère

Pour essayer de comprendre ce qui impacte la vitesse de saisie lorsqu'il y a une liste de prédiction, nous avons réalisé une étude plus précise en analysant les résultats caractère par caractère. Dans un premier temps, nous avons regardé les déplacements effectués entre chaque bigramme. Pour cela, nous avons distingué les actions qui se déroulent exclusivement sur le clavier logiciel de celles effectuées en partie sur la liste de prédiction.

Sur le clavier logiciel augmenté de la liste, la saisie de caractères sur le clavier logiciel représente environ 89% des actions effectuées. Dans un premier temps, nous avons donc voulu voir si le temps de pointage d'une touche sur le clavier logiciel était différent entre le clavier seul et le clavier accompagné de la liste. Pour cela, nous avons classé l'ensemble des tâches de pointage en fonction de leur indice de difficulté, et fait la moyenne des temps de saisie pour chacun des indices de difficulté.

La Figure 32 présente donc les temps de saisie d'un caractère pour les deux dispositifs et ce en fonction de la difficulté de la tâche. De la même manière que pour la loi de Fitts, nous avons pu calculer les droites de regression linéaire. Les deux droites ont des coefficients de correlation respectivement très bon pour les temps de saisie du clavier seul (R^2=0,96) et correct pour le clavier avec la liste (R^2=0,834). Même si le degré de corrélation est moins bon pour la liste, ces droites nous donnent une information très claire : les deux droites ont quasiment la même pente, mais la droite de régression du clavier avec liste a un temps fixe supérieur d'environ 100 ms par rapport à celle du clavier logiciel seul. Ceci montre donc clairement que le simple fait d'avoir la liste à côté du clavier logiciel, perturbe l'utilisateur dans sa saisie sur le clavier.

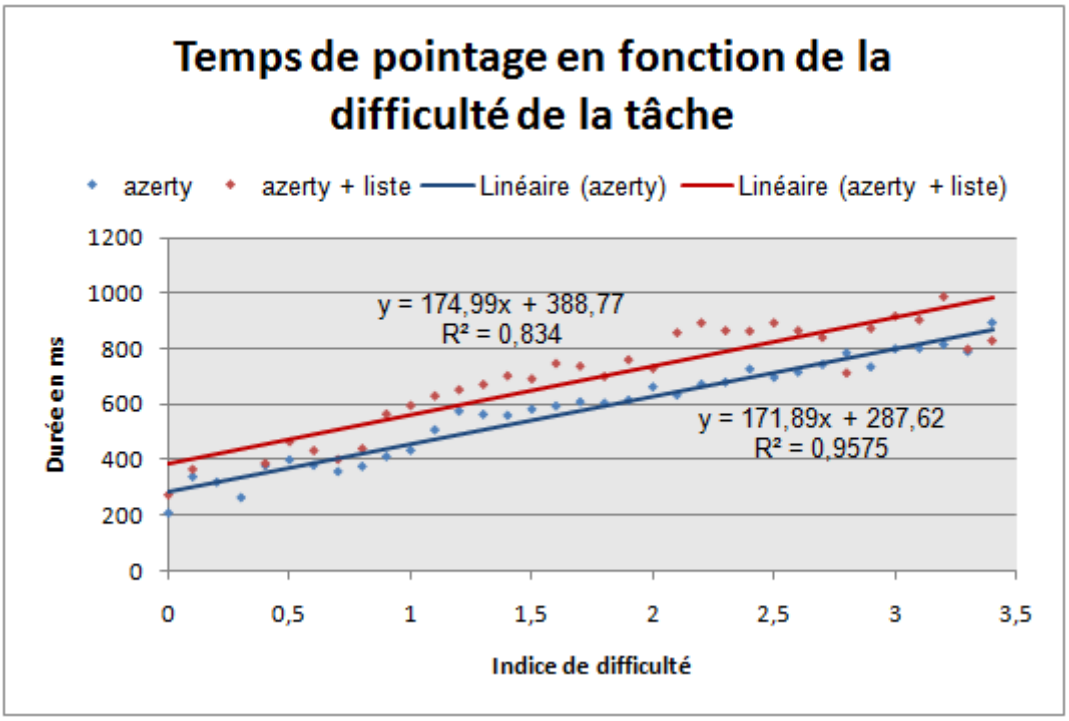


Figure 32 : Temps de saisie d'un caractère en fonction de la difficulté pour l'atteindre

IV.5.b. Temps de fixation sur la liste

Pour essayer d'expliquer ce temps fixe supplémentaire, une des hypothèses est que l'utilisateur porte une attention à la liste, et ce même s'il saisit le caractère sur le clavier logiciel. Pour vérifier cette hypothèse, nous avons regardé l'impact temporel du regard sur la liste. Les fixations et les saccades enregistrées par le système de suivi du regard sont datées avec un temps de début et un de fin. Ainsi nous avons pu déterminer la durée d'une fixation ou d'une saccade. Grâce à ces données, nous avons donc compté le nombre de fixations qui étaient portées sur la liste durant la saisie d'un caractère. En plus de cela, nous avons mesuré le temps que l'utilisateur passe à regarder la liste pendant sa saisie (soit sous la forme de saccade, soit sous la forme de fixation). De même, nous avons mesuré le temps qu'il passe à regarder le clavier.

La Figure 33 présente les résultats obtenus. Les temps (en ordonnée) sont donnés en fonction du nombre de fixations (en abscisse). Les courbes verte et rouge représentent

respectivement le temps où l'utilisateur regarde le clavier logiciel et celui où il regarde la liste. S'il est difficile de dégager une tendance pour le temps passé à regarder le clavier logiciel, la droite de régression linéaire correspondant à la courbe de temps passé à regarder la liste a un très fort coefficient de corrélation. Nous pouvons donc affirmer que chaque fixation sur la liste prend environ 250ms d'attention à l'utilisateur. Ce temps se répercute sur le temps de saisie d'un caractère. La courbe bleue représente le temps moyen de saisie d'un caractère en fonction du nombre de fois où l'utilisateur va fixer la liste de prédiction. La droite de régression linéaire a aussi un fort taux de corrélation avec la courbe (R^2=0,96). Nous pouvons ainsi estimer qu'une fixation sur la liste coûte environ 325 ms à l'utilisateur lors de la saisie d'un caractère.

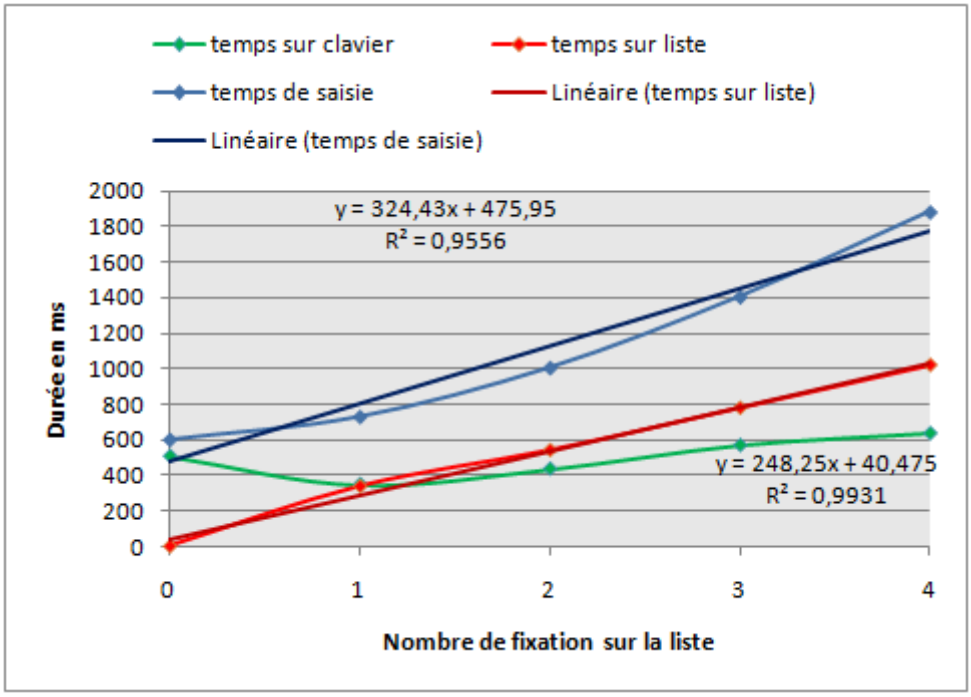


Figure 33 : Temps passé à regarder la liste

IV.5.c. Fréquence des regards portés sur la liste

Ce temps est largement supérieur aux 100 millisecondes évoquées sur la Figure 32. Ceci est du au fait que l'utilisateur ne regarde pas la liste lors de chaque saisie de caractères. En effet, la Figure 34 montre que pour 66,7% des caractères saisis sur le clavier logiciel, l'utilisateur ne regarde pas la liste. Ainsi, les 325 ms d'attention sur la liste ne sont donc appliqués qu'environ une fois sur trois, ce qui revient donc à impacter chaque saisie d'environ 110 ms.

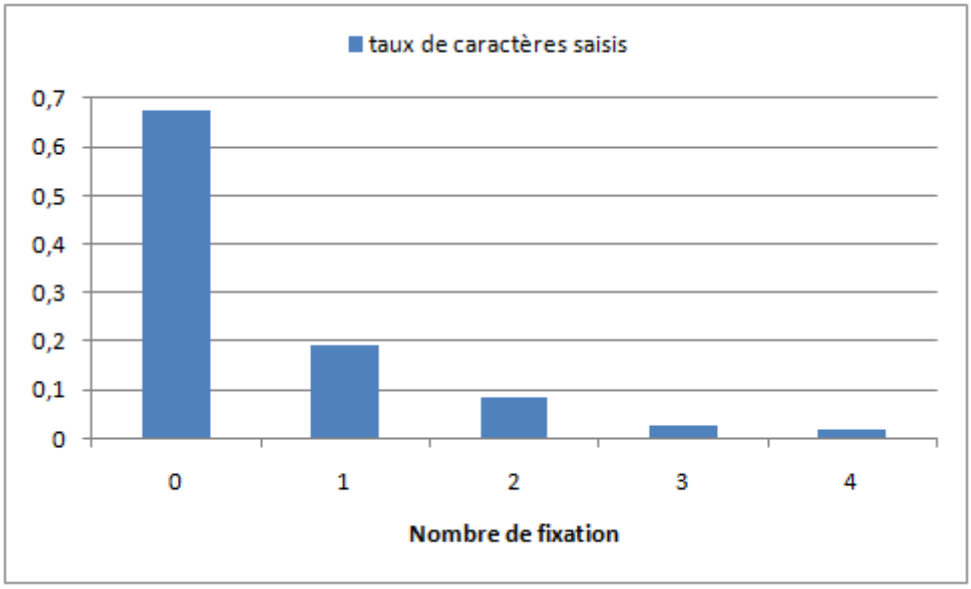


Figure 34 : Taux de caractères saisis en fonction du nombre de fixation

La Figure 35 nous donne une information sur le moment où l'utilisateur regarde la liste. En effet, cette courbe donne le taux de fixation sur la liste en fonction du nombre de caractères déjà saisis. On peut ainsi voir que plus l'utilisateur avance dans le mot, et plus il regarde en direction de la liste.

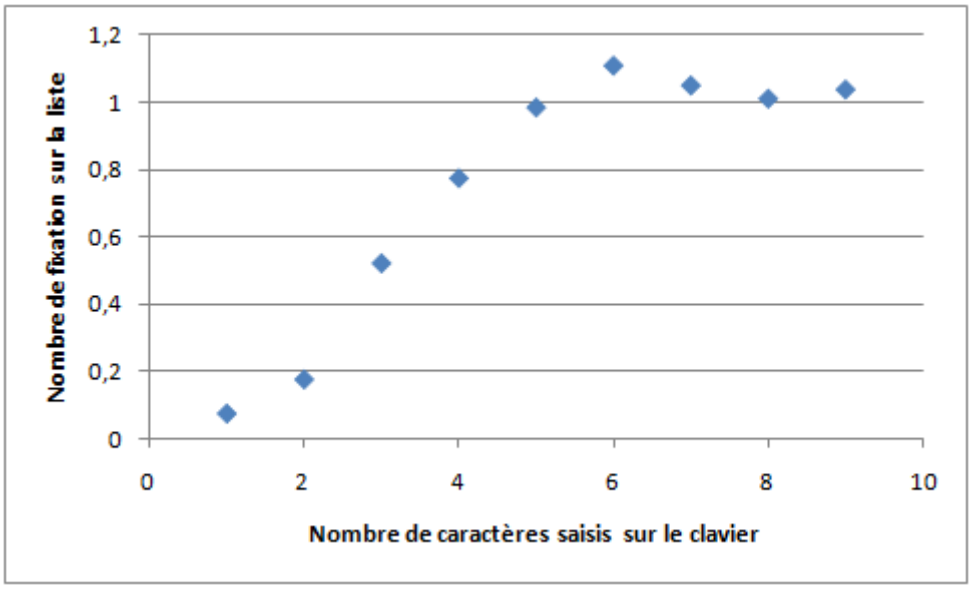


Figure 35 : Taux de fixation en fonction de l'avancée dans le mot

Ceci se traduit par une augmentation du temps moyen nécessaire pour saisir un caractère en fonction du nombre de caractères saisis. Plus l'utilisateur avance dans la saisie du mot, plus le temps pour saisir un caractère augmente.

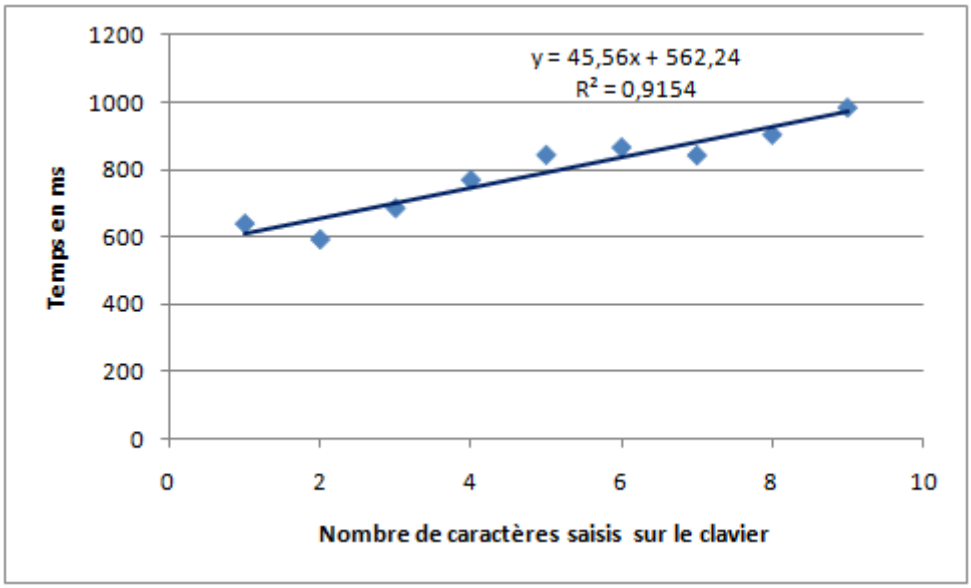


Figure 36 : Temps de fixation en fonction de l'avancée dans le mot

IV.6. Saisie sur la liste de prédiction

Après avoir regardé l'impact sur la saisie sur le clavier logiciel, nous allons maintenant regarder l'impact direct de l'utilisation de la liste de prédiction.

IV.6.a. Temps de saisie dans la liste en fonction de l'avancée dans le mot

Une sélection dans la liste prend en moyenne 1260 ms. Ceci est donc plus long que de saisir un caractère sur le clavier. Cependant cette sélection peut correspondre à la saisie de plusieurs caractères d'un coup. Cette sélection peut donc être bénéfique pour l'utilisateur si elle intervient assez tôt dans le mot. Deux raisons principales à cela : d'une part, comme nous l'avons vu précédemment, plus l'on avance dans le mot sans utiliser la liste et plus la saisie d'un caractère est pénalisée par l'attention portée à la liste. D'autre part, on peut imaginer que plus nous l'utilisons tôt dans la saisie du mot, plus nous sélectionnons un nombre important de caractères d'un coup. La Figure 37 confirme ce sentiment en montrant que le temps moyen pour saisir un caractère augmente avec le moment où l'on utilise la liste. Chaque caractère saisi en plus avant d'utiliser la liste pénalise le temps moyen de près de 30 ms.

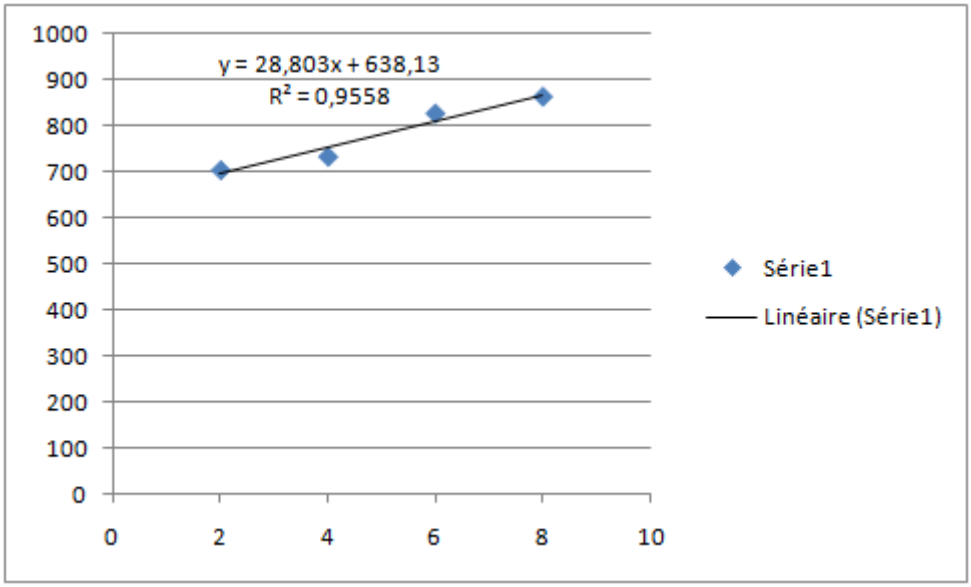


Figure 37 : Temps de saisie moyen d'un caractère en fonction de l'utilisation de la liste

IV.6.b. Utilisation de la liste en fonction de la position du mot dans la liste

Une autre information intéressante est que l'utilisateur effectue la majorité de ses sélections dans la liste dans les trois premiers items de la liste. La Figure 38 présente ce résultat. Les histogrammes en bleu sont le nombre de fois où le mot à saisir a été positionné dans la liste. Il s'agit ici de sa première apparition dans la liste. Si l'utilisateur a continué sa saisie au clavier, il se peut que le mot à saisir soit par la suite remonté dans la liste. Les histogrammes en rouge présentent le nombre de fois où l'utilisateur a sélectionné un mot en fonction de sa position dans la liste au moment où il l'a saisit. On peut donc clairement voir que l'utilisateur soit n'utilise pas la liste, soit attend que le mot soit dans les premières positions de la liste pour le sélectionner.

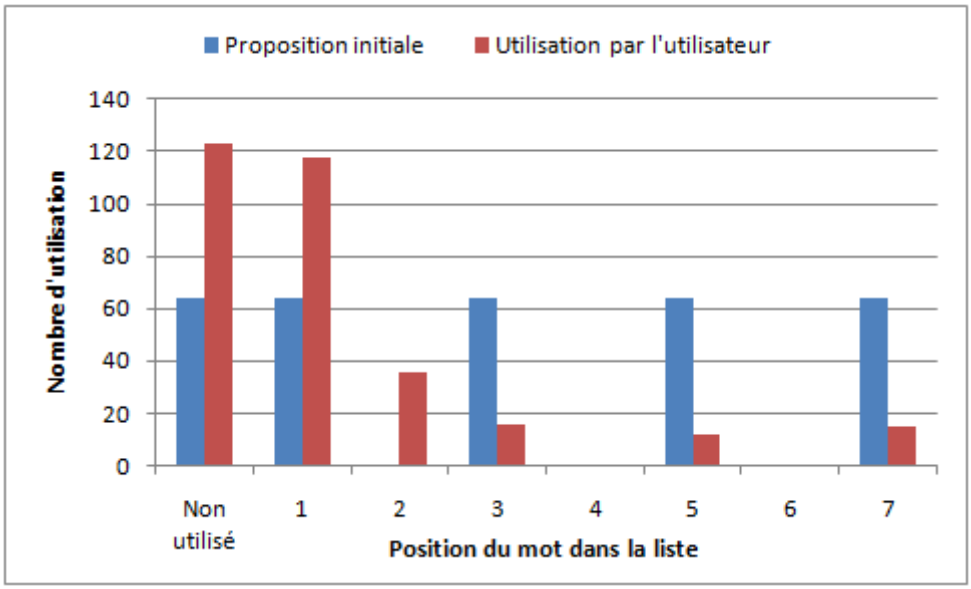


Figure 38 : Utilisation de la liste en fonction de la position du mot

Pour expliquer le fait que l'utilisateur ne sélectionne quasiment que des items dans le haut de la liste, nous avons mesuré le nombre de fois où l'utilisateur regardait dans la partie haute de la liste et inversement le nombre de fois où il regardait en bas de cette même liste. La Figure 39 présente le taux de regard en haut et en bas de la liste en fonction du nombre de

fixations que l'utilisateur fait sur la liste au moment d'aller sélectionner un item dans cette dernière. On peut ainsi voir que la majorité de ces regards se portent sur la partie haute de la liste. Ceci peut donc expliquer le fait qu'il ne prenne en compte quasiement que les items du haut.

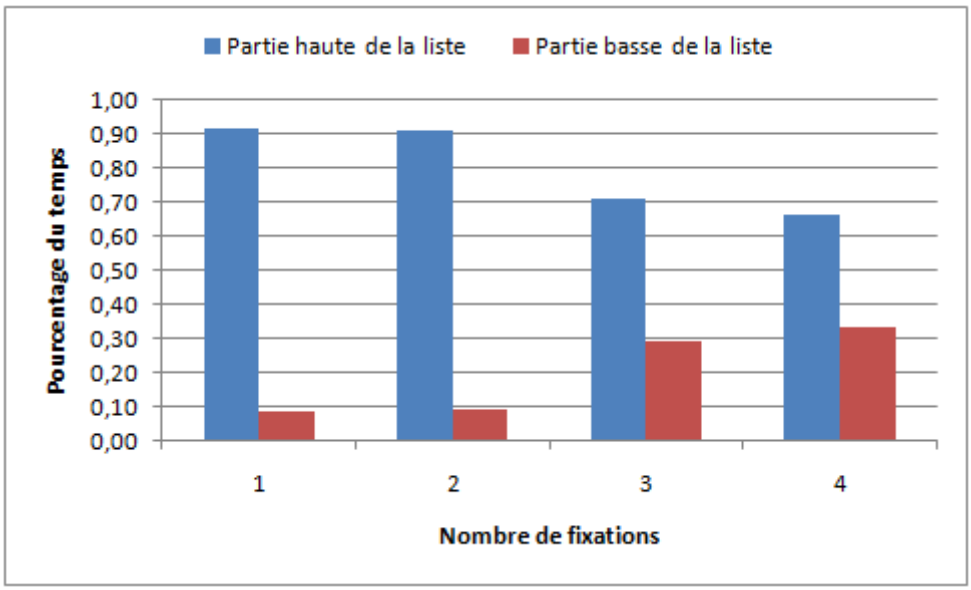


Figure 39 : Position du regard dans la liste

IV.6.c. Utilisation de la liste en fonction de l'avancée dans le mot

Le fait que l'utilisateur ne saisisse quasiment que dans les trois premiers items de la liste a aussi un autre impact : il saisit plus de caractères au clavier avant d'aller sélectionner dans la liste. La Figure 40 vient soutenir cette hypothèse. Les mots proposés à l'utilisateur avaient été choisis de manière à apparaitre de manière uniforme après la saisie de 2, 4, 6 ou 8 caractères (histogramme bleu). Or, on peut voir que l'utilisateur (histogramme rouge) a, en réalité, effectué une sélection dans la liste après avoir saisi entre 4 et 9 caractères. En moyenne, l'utilisateur a saisi au clavier 2 caractères en plus après l'apparition du mot dans la liste avant d'aller sélectionner le mot dans la liste.

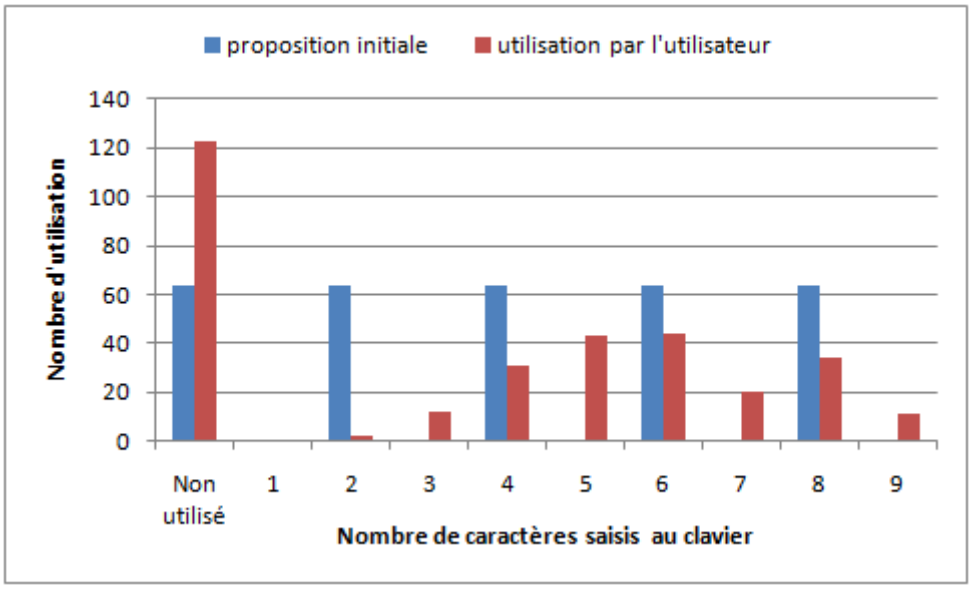


Figure 40 : Sélection dans la liste en fonction du nombre de caractères déjà saisis

Section V - Synthèse

Pour résumer les résultats que nous venons de présenter, nous pouvons donc dire qu'en général l'utilisateur ne regarde que le haut de la liste. Par conséquent, il n'utilise pas la liste de prédiction de manière optimale et saisit plus de caractères sur le clavier logiciel. Cependant, même la saisie sur le clavier logiciel est pénalisée par la présence de la liste à côté et l'attention que lui porte l'utilisateur. Cette utilisation sous optimale de la liste couplée à la saisie moins rapide sur le clavier logiciel entraîne donc une vitesse de saisie moins rapide sur le dispositif clavier logiciel auquel nous avons ajouté une liste.

Les modèles théoriques actuels ne prennent pas en compte ces constats dans leur modélisation. C'est ce que nous allons essayer de mettre en place dans la section suivante.

Chapitre 3 - Evaluation théorique d'une liste de prédiction

Une expérimentation avec des utilisateurs est une phase importante pour prouver qu'un système interactif est efficace et adapté aux besoins des utilisateurs. Cependant, cette étape est assez coûteuse en temps et elle nécessite le recrutement des participants. Ceci est encore plus compliqué lorsqu'il s'agit de faire une expérimentation avec des personnes ayant un handicap moteur. Ces personnes ont généralement un équipement spécialisé ce qui rend leur déplacement difficile. Une expérimentation peut s'avérer lourde à mettre en place. Il est préferable d'avoir une première estimation de la pertinence du système avant de mettre en œuvre une expérimentation.

Une méthode généralement utilisée consiste à prédire théoriquement les performances du système. Des modèles de prédiction performants existent pour des tâches telles que le pointage ou la saisie de texte sur clavier logiciel statique. En revanche, il existe peu de modèles permettant de prédire les performances d'un utilisateur qui utiliserait un clavier logiciel couplé à une liste de prédiction. En nous basant sur les résultats obtenus et présentés au chapitre précédent, nous proposons dans ce chapitre un modèle théorique prédisant le temps moyen nécessaire pour saisir un caractère sur ce type de clavier. De manière à pouvoir utiliser facilement le modèle présenté, nous proposons un outil permettant de simuler l'utilisation d'un clavier logiciel augmenté d'une liste de prédiction et de connaître les performances théoriques de ce clavier.

Section I - Revue des modèles théoriques existants

L'étude théorique se base sur une loi prédictive uniquement basée sur un calcul et ne nécessite ni la mise en œuvre d'expérimentation ni l'intervention des utilisateurs. Elle permet

de calculer les performances théoriques d'un système sur des critères tels que la vitesse de saisie de texte, la recherche visuelle d'une touche, etc.

I.1. Temps de recherche visuelle d'une touche

Le temps de recherche visuelle d'une cible fut abordé pour la première fois en 1952 par Hick [Hick 1952]. Cette loi prédit le temps nécessaire pour sélectionner un item parmi un ensemble de choix. En l'appliquant sur les claviers, c'est le temps que prend l'utilisateur pour trouver ou sélectionner une touche. Ce temps dépendait de la quantité d'information et le nombre de choix proposés. Un an plus tard, Hyman [Hyman 1953] étend la loi de Hick et suppose que les choix ou « stimulii » ont une probabilité d'occurrence.

La loi est ensuite reformulée afin de déterminer le temps de réaction nécessaire pour amorcer une action de pointage. Ce temps est fonction du nombre de cibles dont il est possible de choisir en considérant équiprobable l'accès aux cibles et en prenant en considération le fait que l'utilisateur n'est pas familiarisé avec la disposition des items :

$$TR = a + b \times \log_2(N+1)$$

Où :

- **TR** représente le temps de recherche d'une cible (en secondes) ;
- **N** le nombre de cibles dont il est possible de choisir (le +1 est une extension de la loi proposée par Card et all [Card 1983] et matérialise, dans le processus de décision, l'alternative d'agir ou non) ;
- **a** et **b** sont des constantes qui sont calculées empiriquement par régression linéaire[14]. Ces constantes dépendent du contexte général d'utilisation.

[14] La courbe de régression linéaire est calculée à partir d'un ensemble de points obtenus expérimentalement, où l'axe des abscisses et celui des ordonnées représentent respectivement le nombre de cibles et le temps nécessaire pour en rechercher une.

- **b** : la pente de la droite de régression, est exprimée en secondes et est évaluée à environ 0,15 s pour un individu normal.
- **a** : est l'ordonnée à l'origine de cette même droite et est généralement évaluée à 0.

I.2. Temps de pointage d'une cible

I.2.a. La loi de Fitts

L'étude du temps de pointage d'une cible remonte à la fin de XIX^{e} siècle avec Woodworth qui démontra en 1899 qu'il est impossible voire incompatible d'associer à la fois une grande rapidité du mouvement et une haute précision dans le pointage d'une cible [Woodworth 1899]. En 1954, Fitts [Fitts 1954] proposa une série d'expérimentations d'acquisition de cibles avec un stylet. L'utilisateur devait pointer de droite à gauche en touchant avec son stylet des cibles variables en taille et en distance (Figure 41). Cette expérience a conduit l'auteur à proposer une loi permettant d'évaluer les performances motrices d'un utilisateur. Une loi qui met en relation la rapidité du mouvement et la précision du pointage en se basant sur la formulation de Shannon [Shannon 1949]. Cette théorie définit la capacité **C** de transmission d'un signal **D** avec du bruit **W**.

$$C = log_2\left(\frac{W + D}{W}\right) = log_2\left(\frac{D}{W} + 1\right)$$

Fitts utilisa cette formule dans le domaine perceptuo-moteur, et définit alors une formule qui prédit le temps du mouvement en fonction de la distance entre les cibles et leur taille :

$$MT = a + b \times ID$$

Avec :

$$ID = log_2\left(\frac{2D}{W}\right)$$

Où :

- D est la distance à parcourir pour aller pointer la cible ;

- W est la taille de la cible ;
- ID est l'indice de difficulté en bits ;
- MT est le temps prédit pour effectuer le mouvement ;
- a et b sont des valeurs empiriques qui décrivent le profil moteur d'un utilisateur. Elles se déterminent par régression linéaire.

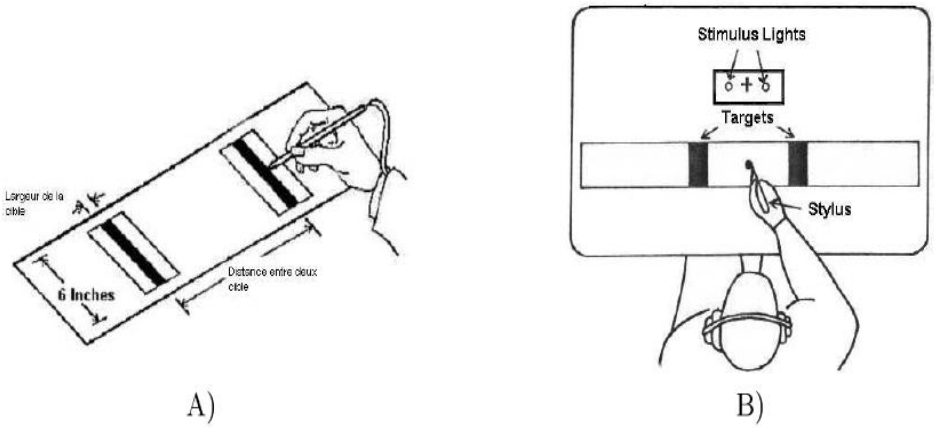


Figure 41: Expérimentations menées par Fitts : A) Expérience « mouvements de va-et-vient » [Fitts 1954]; B) Expérience « atteinte d'une cible » [Fitts 1964]

I.2.a. Développement de la loi de Fitts pour l'IHM

La loi de Fitts a été proposée suite à l'étude des mouvements de va et vient du stylet de l'utilisateur entre 2 cibles physiques. A partir des années 70, la loi de Fitts est adaptée pour l'Interaction Homme-Machine. Ce sont Card et al. [Card 1978], qui ont appliqué cette loi dans le cas de cible virtuelle. Ils ont montré que le temps de pointage sur un écran d'ordinateur à l'aide d'une souris suivait la loi de Fitts.

Dans le même contexte, MacKenzie essaya de rendre la formule de Fitts plus robuste et la modifia. Ce temps de pointage établi par [MacKenzie 1992a] peut être défini par la formule suivante :

$$T = a + b \times \log_2(\frac{D}{W} + 1)$$

Où T est le temps exprimé en secondes, D est la distance à parcourir, W la taille de la cible, a et b sont des variables identiques aux variables de l'équation originale de Hick-Hyman.

I.2.b. Le modèle de Soukoreff et MacKenzie

Dans une interaction sur un clavier logiciel, l'utilisateur utilise la souris ou n'importe quel dispositif de pointage pour cliquer sur un caractère. Il s'agit de la technique de pointage et de sélection. Sur un clavier logiciel, les touches du clavier sont considérées comme des cibles. La loi de Fitts a été adaptée pour la saisie sur un clavier logiciel en tenant compte des probabilités d'apparition de chaque lettre. Cette adaptation a été mise au point en 1995 par Soukoreff et MacKenzie [Soukoreff 1995].

Ils proposèrent une nouvelle formule pour prédire « *a priori* » les performances de saisie d'un utilisateur au moyen d'un clavier logiciel. Le temps moyen MT (exprimé en secondes par caractère) pour sélectionner un caractère sur un clavier logiciel limité aux 26 lettres de l'alphabet et le caractère espace, est calculé à partir de la formule suivante :

$$MT = \sum_{i=1}^{27} \sum_{j=1}^{27} P_{ij} T_{ij}$$

Où :

- P_{ij} : La probabilité d'apparition du bigramme C_iC_j où C_i et C_j sont deux caractères appartenant à l'alphabet ;
- T_{ij} est exprimé par la formule :

$$T_{ij} = a + b \times \log_2(\frac{D_{ij}}{W_j} + 1)$$

Avec :

- D_{ij} : La distance séparant les touches associées aux caractères C_i et C_j ;
- W_j : La largeur de la touche associée au caractère C_j ;

Notons que ce temps est largement dépendant du profil des utilisateurs par rapport au clavier logiciel et à la disposition des caractères. On distingue principalement deux profils d'utilisateur :

- Les experts : ceux sont les utilisateurs qui connaissent la disposition des touches et par conséquent ne prennent presque pas de temps pour retrouver la touche cible. Dans ce cas, nous pouvons considérer comme nul le temps de recherche.
- Les novices, qui connaissent mal la disposition des caractères sur le clavier et qui peuvent prendre un temps important dans la recherche d'une touche donnée.

Ces deux profils d'utilisateur sont les extrêmes que l'on peut rencontrer dans la saisie de texte. On peut estimer qu'un utilisateur l'ambda aura des performances qui seront comprises entre ces deux bornes :

$$CPS = \begin{cases} \dfrac{1}{\overline{MT} + TR} & pour\ un\ novice \\ \dfrac{1}{\overline{MT}} & pour\ un\ expert \end{cases}$$

Pour un novice, le temps de saisie est égal à la somme du temps de recherche (TR) et du temps de mouvement ou de sélection $(\overline{MT})$. Le CPS calculé, CPS_{min} est la limite inférieure du temps espéré avec une personne utilisant le clavier pour la première fois. Pour un expert, le temps de recherche TR étant considéré comme nul, nous obtenons la vitesse maximale ou CPS_{max}.

I.2.c. Tables d'actions empiriques

Le déplacement entre les touches du clavier est modélisé par la loi de Fitts pour les modèles de Soukoreff et MacKenzie. Selon la loi de Fitts, le temps de mouvement ne

dépend que de l'indice de difficulté : le rapport entre la distance à la cible et sa largeur. Cela signifie que pour un clavier rectangulaire ordinaire, le temps de circulation ne change pas avec la taille du clavier. Par conséquent, pour un clavier considérablement petit, la vitesse entrée prédite sera la même que pour un clavier de grande taille.

Les expériences ont montré que la taille du clavier n'a pas d'effet sur la vitesse de frappe, mais dans une certaine limite [Soukoreff 1995]. La méthode de tables d'actions [Hughes 2002] prend en compte les dimensions réelles du clavier lors de la prédiction de la vitesse de saisie pour un expert.

Au lieu d'utiliser la loi de Fitts comme un modèle pour le mouvement moteur en tapant sur un clavier, des tables bi-actions empiriques sont utilisées. Ces tableaux sont construits en considérant uniquement les aspects physiques du système de saisie de texte. Ces aspects physiques influent sur le temps requis pour exécuter certaines actions juste après avoir terminé une autre (bi-action). Par exemple, le fait de saisir le caractère 'j' juste après avoir tapé la lettre «i» sur un clavier est considéré comme une bi-action.

La saisie de texte peut être alors vue comme un processus d'exécution de séquences consécutives. Pour chaque saisie de caractère, il y a logiquement une lettre qui va succéder lors de l'exécution de cette action. C'est pourquoi Hughes et ses collègues ont proposé une méthode d'évaluation plus fine. Cette méthode consiste à remplacer le temps de déplacement d'une touche à une autre par un temps concret obtenu lors d'une expérimentation. Lors de l'expérimentation, les sujets devaient réaliser tous les déplacements possibles sur le clavier en test (cf. Figure 42). A chaque couple de touches, on associe un temps de déplacement et l'ensemble de ces couples est sauvegardé sous la forme d'une table d'actions.

A5	B5	C5	D5	E5	F5
A4	B4	C4	D4	E4	F4
A3	B3	C3	D3	E3	F3
A2	B2	C2	D2	E2	F2
A1	B1	C1	D1	E1	F1

Figure 42: Interface utilisée pour réaliser la table d'actions pour des claviers comprenant 5 lignes et 6 colonnes (extraite de [Hughes 2002]**)**

L'avantage de cette méthode est qu'elle propose une évaluation théorique plus précise et moins coûteuse que celle de Soukoreff et MacKenzie. Cependant, il faut recommencer l'expérimentation chaque fois que l'on change la disposition des caractères à la différence du modèle de Soukoreff et MacKenzie.

I.3. Prédiction de performance

Les modèles que nous venons de présenter ne s'occupent que d'une partie de la tâche de saisie : la recherche ou la sélection d'un caractère. Pour considérer la tâche dans son ensemble et prendre tous les paramètres en compte, un modèle plus complet peut être nécessaire.

Le modèle Keystroke-Level Model (KLM) [Card 1980] [Card 1983] a été développé, pour prédire le temps nécessaire pour réaliser une tâche sur un ordinateur. Les auteurs supposent que l'utilisateur dévie de l'opération postulée quand il fait des erreurs. Ce modèle ne prédit que les opérations qui s'effectuent sans erreur ni interruption. Il décompose la tâche de saisie en six actions élémentaires. Le temps de chaque action dépend de l'expérience de l'utilisateur. Trois types d'utilisateur ont été définis : expert, averti, débutant [Kieras 1993]. La prédiction du temps (noté $\mathbf{T_{EXEC}}$) est basée sur l'addition des temps estimés pour réaliser les six actions élémentaires suivantes :

- **T_K** : (« Keystroking ») le temps de pression ou de relâchement d'une touche du clavier. Ce scénario dépend de la disposition des lettres et de l'expérience de l'utilisateur.
- **T_P** : (« Pointing ») le temps de pointage. (valeur moyenne de 1.1 secondes). ce temps peut aussi être calculé au moyen de la loi de Fitts ;
- **T_H** : (« Homing ») le temps de déplacement de la main pour saisir le clavier ou la souris (considéré de 0.4 secondes).
- **T_D** : (« Drawing ») le temps de dessin de lignes et de courbes avec une souris. Ce temps dépend du nombre et de la taille des lignes.
- **T_M** : (« Mental ») temps de l'activité mentale, ou le temps de préparation avant d'exécuter les opérations physiques. Comme toutes les opérations physiques ne nécessitent pas une activité mentale, ce temps n'est pris en considération que quand c'est nécessaire (1.35 secondes).
- **T_R** : (« Response ») temps de réponse du système (0 seconde).

Le temps T_{EXEC} est calculé comme suit :

$$T_{EXEC} = T_K + T_P + T_H + T_D + T_M + T_R$$

Certains de ces temps peuvent être ignorés suivant le scénario de la tâche. Par exemple dans [Dunlop 2000], l'auteur ne considère que trois de ces temps : T_K, T_H et T_M pour prédire le temps de saisie sur une clavier téléphonique. D'autres temps peuvent être calculés par d'autres modèles comme par exemple le temps de pointage qui peut être prédit par les lois de Fitts.

Dans le modèle de KLM le temps de pointage est considéré comme un temps fixe indépendamment de sa complexité. Cependant, ce temps varie avec la distance de la cible et sa taille.

I.4. Prédiction du temps de recherche et de sélection dans une liste

Dans le même thème, nous trouvons des recherches et des études essayant de modéliser et calculer le temps de recherche dans un menu. Dans [Landauer 1985], l'auteur a conçu une

étude empirique sur la recherche dans un menu, et a trouvé que la sélection d'une option peut être modélisée par la loi de Hick-Hyman et le temps de pointage par les lois de Fitts. De même, Cockburn et al, ont proposé dans [Cockburn 2007] un modèle prédictif de perfomance de la sélection dans un menu. Ce modèle incorpore en même temps les lois de Hick-Hyman et Fitts et intègre les transitions en fonction de l'expérience de l'utilisateur. En effet, un utilisateur expert n'examine plus tous les éléments du menu.

La sélection et la recherche d'une option dans un menu est différente de celle dans une liste de mots de prédiction. Le menu contient des items statiques qui ne changent pas durant l'interaction avec l'utilisateur, tandis qu'une liste de mots est dynamique. Cette dernière est mise à jour par le système de prédiction à chaque fois que l'utilisateur saisit une nouvelle lettre. Par conséquent, contrairement à un menu, l'utilisateur a besoin de plus de temps pour chercher un mot dans la liste en changement continu [Hornof 1997]. Le modèle de recherche et sélection dans un menu ne peut pas être utilisé directement dans l'évaluation de la liste de mots. Dans les travaux de [Koester 1994b], les chercheurs ont essayé d'étudier les performances d'un sujet utilisant une liste de prédiction pour trouver sous quelles conditions la prédiction de mot peut accélérer la vitesse de saisie. Ils ont trouvé que c'est la charge cognitive ajoutée lors d'une saisie avec une liste qui pénalise le gain de temps initialement obtenu par la réduction du nombre d'opérations. En effet, les utilisateurs, indépendamment de leur statut médical, passaient plusieurs centaines de millisecondes à chercher le mot dans la liste par rapport à la saisie à l'aide des caractères seuls. Ils ont indiqué que ce temps dit temps de recherche enferme aussi le temps de vérification du mot ou caractère choisi, le temps des mouvements des yeux et de la tête etc. Dans [Koester 1998], l'utilisation de la liste de prédiction dépend directement du temps de recherche. Le temps nécessaire pour générer un caractère est modélisé comme étant le nombre de recherches dans la liste multiplié par le temps de recherche, ajouté au nombre de frappes par caractère multiplié par le temps de frappe. D'où :

$$T_{wp} = (S)(t_s) + (1 - Ksav)\big((t_k)_{wp}\big) \text{ sec/car}$$

Avec :

- S le nombre de recherche dans la liste

- 1-Ksav, le nombre de frappes par caractère
- ts, le temps de recherche
- $(tk)_{wp}$, le temps de frappe

Cependant, les recherches de Koester et al, se déroulaient sur des claviers physiques et non pas sur des claviers logiciels et des techniques de pointage. Les utilisateurs avaient à saisir des caractères sur un clavier standard et à séléctionner des mots dans la liste « logicielle ». Ils avaient à manipuler deux dispositifs différents et perdaient du temps entre ces deux dispositifs. Ce qui ne coïncide pas avec nos travaux de recherche.

Sad [Sad 2009b] a modélisé la recherche et la sélection d'un mot dans une liste. Il a tout d'abord construit un modèle KLM pour calculer le temps de sélection dans une liste. Ce temps est l'intervalle entre le clic du bouton « select » de l'interface et le clic ou la sélection du mot de la liste (cf.Figure 43). Ce temps est la somme de :

- temps d'exécution de l'opération mentale t_M, avec une moyenne de 1.35 secondes ;
- temps d'exécution de l'opération de pointage t_P, de 1.1 secondes en moyenne;
- et t_K, le temps d'exécution du clic avec une valeur moyenne de 0.2 seconde.

Le temps de sélection proposé est alors de :

$$T=t_M + t_P + t_K = 1.35 + 1.1 + 0.2 = 2.65 \text{ secondes.}$$

Figure 43: Interface de sélection de mots dans une liste (repris de [Sad 2009a]).

Sad a aussi essayé de trouver une formule basée sur une étude empirique pour modéliser le temps de recherche dans une liste. Pour cela, il a mené une expérimentation avec des utilisateurs sélectionnant des mots dans une liste triée par ordre alphabétique et une autre dans une liste avec un ordre arbitraire. La liste contenait 20 mots dans les deux cas.

Après avoir mémorisé le mot à chercher, l'utilisateur appuie sur le bouton « *Select* ». La liste est alors affichée, le chronometer est déclenché et l'utilisateur cherche le mot dans la liste avec la possibilité de défilement si nécessaire. Le temps est arrêté quand l'utilisateur sélectionne le mot dans la liste. Les données récupérées sont le temps de sélection et la position du mot dans la liste. A partir de ces données, les chercheurs ont pu représenter ces résultats dans des équations linéaires de la forme :

$$T = a \times n + b$$

Avec :

- T le temps moyen de sélection en millisecondes (ms) ;
- a et b des constantes empiriques ;
- n l'indice du mot dans la liste.

I.5. Synthèse

Les modèles que nous venons de présenter et qui traitent des listes de prédiction ne sont pas utilisables pour les claviers logiciels accompagnés d'une liste de prédiction. En effet, les travaux de Koester portent sur l'utilisation d'une liste de prédiction avec un clavier physique. Ceci est complètement différent de notre problématique car l'utilisateur a généralament deux dispositifs à manipuler : le clavier physique et la souris. Le fait de passer d'un dispositif à un autre représente déjà une perte de temps. De plus, le focus d'attention n'est pas le même : quand on utilise un clavier physique, on ne va pas forcément toujours regarder l'écran, à moins de connaitre parfaitement le clavier physique. Ainsi, les résultats trouvés par Koester et al. sont difficilement applicables à la problématique des claviers logiciels.

Les travaux de Sad et Poirier se limitent à une sélection dans une liste seule. Ils ne sont pas corrélés avec une saisie au clavier. L'attention portée par l'utilisateur sur la liste seule est differente que s'il saisit sur un clavier et sélectionne dans une liste. Si leur modèle permet de définir un temps de sélection d'un item dans la liste, il peut s'appliquer à notre cas, où un utilisateur saisit un mot au clavier logiciel couplé à une liste de prédiction. Cependant ce modèle ne prendrait pas bien en compte la transition entre la saisie au clavier et la sélection dans une liste, ce qui constitue un manque au modèle proposé. A noter encore que dans notre cas la liste est dépourvue de la possibilité de défilement. Tous les mots de la liste apparaissent dans une seule « fenêtre ».

Section II - Modèle proposé

II.1. Proposition de modèle

Le modèle que nous proposons a pour objectif de prédire le temps moyen nécéssaire pour saisir un caractère lorsque nous effectuons une saisie au moyen d'un clavier logiciel couplé à une liste de prédiction. La saisie des mots peut être dépendante de la liste de mots présents. Notre modèle est forcément dépendant du système de prédiction proposé. Pour pouvoir proposer un temps moyen de saisie d'un caractère, nous nous appuyons sur le temps mis en moyenne pour saisir un ensemble de mots d'un corpus. Pour chaque mot,

nous calculons le temps moyen qu'il faut pour saisir ce mot. Le temps moyen pour saisir un caractère avec le système évalué est la somme des temps prédits pour chaque mot divisée par le nombre total de caractères composant l'ensemble des mots du corpus.

Le temps moyen est ainsi obtenu au moyen de la formule suivante :

$$T = \frac{\sum_{m \in D} T(m)}{\sum_{m \in D} |m|}$$

Où :

- D est le corpus de mots considéré,
- |m| est la taille en caractères du mot m,
- T(m) est le temps moyen mis pour saisir le mot m.

Saisir un mot avec un clavier logiciel muni d'une liste de prédiction peut être saisi de différentes manières ; il faudrait vérifier si le mot apparait dans la liste de prédiction et quand il apparait dans la liste. Par exemple, avec notre système de prédiction et une liste de prédiction de 7 mots, le mot « bonjour » apparaît après la saisie des quatre premiers caractères. Les alternatives pour saisir le mot bonjour sont :

- Saisir 'b', 'o', 'n' et 'j' au clavier puis sélectionner le mot dans la liste ;
- Saisir 'b', 'o', 'n', 'j' et 'o' au clavier puis sélectionner le mot dans la liste ;
- Saisir 'b', 'o', 'n', 'j', 'o' et 'u' au clavier puis sélectionner le mot dans la liste ;
- Saisir intégralement le mot « bonjour » au clavier ;

Le temps moyen nécessaire pour saisir un mot doit prendre en compte ces différentes alternatives. Nous proposons de faire la moyenne des temps prédits pour ces différentes alternatives.

Nous obtenons donc :

$$T(m) = \frac{1}{1 + |m| - P(m)} \sum_{i=P(m)}^{|m|} T(m)_i$$

Où

- |m| est la taille du mot en caractères ;
- P(m) est la fonction qui renvoie le nombre de caractères après lequel le mot apparait dans la liste et est en mesure d'être saisi.
- $T(m)_i$ est le temps nécessaire pour saisir le mot m en considérant que l'utilisateur utilisera la liste de prédiction après le $i^{ème}$ caractère saisi au clavier. Dans ce cas, le temps de saisie est obtenu en sommant l'ensemble des tâches de pointage sur le clavier logiciel ainsi que le temps nécessaire pour sélectionner l'item dans la liste s'il est choisi. On obtient la formule suivante :

$$T(m)_{\boldsymbol{n}} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{T_i(m)} + \mathrm{E(i, |m|)} \times \mathrm{TL}$$

Où, TL est une constante représentant le temps de sélection d'un item dans la liste de prédiction. Ce temps a été fixé à 1,26 s et correspond au temps moyen de sélection dans la liste que nous avons trouvé dans l'expérimentation présenté au chapitre précédent. Ti(m) correspond au temps nécessaire pour aller pointer le ième caractère du mot m sachant que l'on vient du $(i-1)^{ème}$ caractère. Ce temps est obtenu au moyen de la loi de Fitts :

$$T_i = a + b \times \log_2\left(\frac{D_i(m)}{W} + 1\right)$$

Où Di(m) est la distance qui sépare le ième caractère du mot m au $(i-1)^{ème}$ caractère du même mot. Les constantes a et b ont respectivement été fixées à 0,390 et 0,175s conformément à la droite de régression linéaire présentée sur la Figure 32.

Enfin, pour définir si la liste est utilisée ou non, nous utilisons :

$$E(i, |m|) = \begin{cases} 1, & i < |m| \\ 0, & i = m \end{cases}$$

II.2. Un algorithme plus qu'un modèle mathématique

Les claviers logiciels augmentés d'une liste de prédiction sont dépendants du système de prédiction qu'ils utilisent pour afficher les résultats de prédiction. Par conséquent, pour pouvoir évaluer théoriquement un clavier de ce type, il nous faut connaitre le système de prédiction utilisé. De plus, à la différence du modèle de Soukoreff et Mackenzie, la fonction P(m) est plus qu'une simple table de fréquences de bigramme. En effet, il parait assez difficile d'avoir une table qui représente l'ensemble des fréquences d'apparition des mots en fonction du préfixe saisi.

De ce fait, notre modèle ne peut pas être vu comme une équation mathématique mais doit plus être considéré comme un modèle décrivant l'algorithme à mettre en place pour connaitre le temps moyen de saisie. Cet algorithme, pour pouvoir fonctionner, a besoin d'avoir un accès au système de prédiction dont l'utilité serait de pouvoir prédire les mots qui apparaitront et le moment où ils apparaitront dans la liste. Cet algorithme simule donc la saisie de différents mots en explorant l'ensemble des possibilités qui sont offertes à l'utilisateur.

Evaluer théoriquement les performances d'un clavier logiciel accompagné d'une liste de prédiction oblige le concepteur du clavier à passer par une phase supplémentaire de programmation pour permettre la simulation du mot sur son clavier logiciel. Cette étape peut paraître longue et peu avantageuse pour un concepteur et peut le pousser à supprimer cette étape pour passer directement à une phase d'évaluation expérimentale avec les risques que cela comporte.

Pour éviter de passer sous silence cette phase d'évaluation théorique, nous proposons un outil de simulation permettant aux concepteurs d'évaluer leur système sans avoir à développer d'algorithmes supplémentaires.

Section III - Outils de simulation

Notre idée est de proposer un outil qui permet aux concepteurs de prototyper rapidement leur système de saisie et de pouvoir évaluer théoriquement les performances de saisie de leur prototype.

III.1. Prototypage rapide de clavier logiciel

La première fonction de notre système est de permettre de modéliser rapidement un clavier logiciel ayant un fonctionnement dynamique (cf. Figure 44). Il est ainsi possible de créer rapidement un clavier en positionnant des touches associées à un caractère fixe et d'autres dites dynamiques. Les touches dynamiques se verront associées lors de la simulation aux résultats du système de prédiction. La première touche dynamique recevra le premier mot proposé par le système de prédiction, la seconde touche le second mot et ainsi de suite jusqu'à épuisement des touches ou des résultats du système de prédiction. Chaque touche (fixe ou dynamique) est caractérisée par une position (x,y) et une taille (longueur et hauteur), ce qui permet de calculer toutes les données nécessaires pour notre modèle.

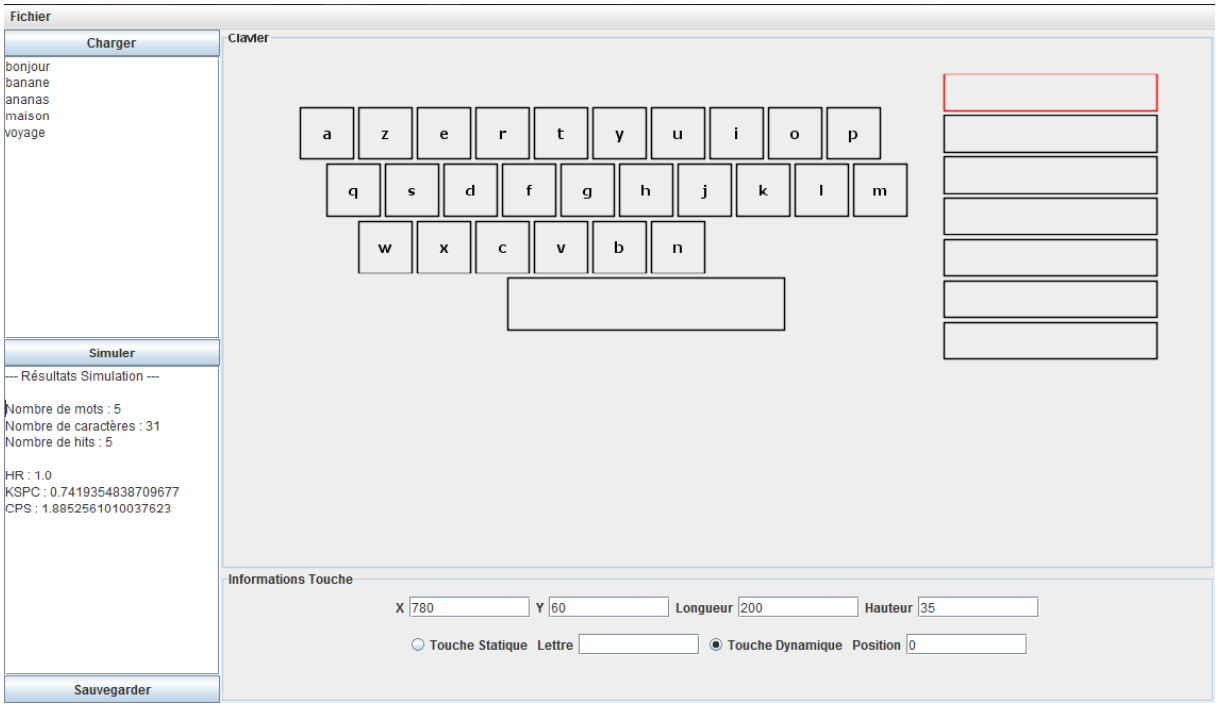


Figure 44 : Interface de prototypage et d'évaluation de clavier logiciel et liste de prédiction

III.2. Simulation de performances

Une fois le clavier prototypé, nous pouvons simuler la saisie de mots sur ce clavier et ainsi obtenir une prédiction des performances qu'espère obtenir un utilisateur avec le clavier logiciel prototypé.

III.2.a. Une simulation parfaite

La simulation d'une saisie parfaite est une technique permettant de tester *« a priori »* les performances théoriques d'un système d'entrée de texte. On l'appelle « parfaite » parce qu'elle ne prend pas en considération la présence de l'utilisateur, sa fatigue et les erreurs commises durant la saisie. Cette méthode permet d'évaluer toutes les formes d'interactions possibles sur un clavier logiciel avec ou sans systèmes de prédiction. Elle permet la collection et l'étude des données concernant les mécanismes d'interaction, les frappes, le temps de réaction, le type et la forme du clavier, etc. en lançant des applications qui testent tous ces paramètres. Elle produit toujours les meilleurs résultats possibles et économise le

temps de l'évaluation car le programme de simulation « interagit » avec le système le plus rapidement possible. Par ailleurs cette méthode est objective et les données recueillies sont analysées et discutées pour récupérer les performances théoriques des systèmes étudiés, améliorer les points faibles et ajouter les points forts. Plusieurs auteurs ont utilisé cette méthode d'évaluation comme [Hunnicutt 1987], [Higginbotham 1992], [Garay 1994], [Wood 1996]. D'auteurs comme dans [Digiovanni 1996] et [Copestake 1998] préfèrent cette technique car elle n'est pas limitée par les cas des utilisateurs et peut aider les chercheurs à prendre des décisions dans la phase de conception du système.

La simulation que nous proposons consiste à dérouler la saisie d'un mot en mesurant la distance qui sépare chaque caractère à saisir. Cette distance est définie par la longueur du segment qui sépare le centre de la touche du caractère précédent et le centre de la touche du caractère à saisir. A partir de cette distance et de la taille des touches, nous pouvons calculer, au moyen de la loi de Fitts, le temps théorique nécessaire pour saisir chaque caractère au clavier. Le calcul de ces temps nous permet de calculer le temps théorique pour chaque alternative possible pour saisir le mot (cf. section précédente). Cette simulation est dite parfaite car celle-ci ne prévoit pas d'erreur de saisie.

III.2.b. Performances calculées pour un corpus de mot

De manière à avoir des performances représentatives de la saisie que pourrait faire l'utilisateur, nous proposons sur l'interface la simulation d'un corpus de mot complet. Ainsi, il est possible de donner à simuler n'importe quel mot ou divers type de corpus, par exemple des corpus de mots d'un domaine spécifique.

Lors de la simulation d'un corpus, l'interface donne des performances moyennes pour l'ensemble du corpus et propose un fichier CSV avec le détail des performances mot à mot.

En plus de la vitesse de saisie théorique moyenne (mesurée en Caratères Par Seconde, CPS), nous donnons aussi le taux d'utilisation possible de la liste (Hit rate, HR), et le taux d'actions nécéssaires pour saisir un caractère (KeyStroke Per Character, KSPC)

Section IV - Discussions

Nous venons de proposer dans ce chapitre un modèle de prédiction de la vitesse de saisie de texte sur un clavier logiciel couplée à une liste de prédiction. Pour aider les concepteurs à utiliser ce modèle, nous proposons en complément un outil de simulation qui utilise ce modèle pour prédire les performances d'un prototype de clavier logiciel.

IV.1. Un modèle perfectible

Le modèle que nous proposons est une première base pour la prédiction de la vitesse de saisie sur ce type de clavier logiciel. Elle prend en compte le fait que l'utilisateur a plusieurs alternatives possibles pour saisir un même mot. Cependant, deux paramètres de notre modèle peuvent certainement être améliorés.

IV.1.a. Quelle est la probabilité d'utiliser chaque alternative ?

La première limite de notre modèle concerne la probabilité que l'utilisateur choisisse une alternative plutôt qu'une autre. Actuellement, dans notre modèle, nous prenons en compte chaque alternative de manière équiprobable. Les résultats que nous avons présentés au chapitre précédent montrent que l'utilisateur ne va pas forcément prendre l'alternative la plus avantageuse, mais va attendre un ou deux caractère(s) supplémentaire(s) pour utiliser la liste. Nous avons pu constater que l'utilisation de la liste dépend en partie de la position du mot dans la liste.

Les alternatives ne sont certainement pas utilisées équiprobablement. Il faudrait pondérer la moyenne de ces temps par une probabilité d'utilisation. Cependant, nos données d'expérimentation ne suffisent pas pour proposer un modèle probabiliste qui prendrait en compte la position du mot dans la liste, et le nombre de caractères saisis avant l'apparition du mot dans la liste. Pour réaliser ce modèle et ainsi l'affiner, il faudra collecter un ensemble de données d'utilisation de la liste. Cette collecte pourra être faite à long terme.

IV.1.b. Modélisation du temps de sélection dans la liste

La seconde limitation de notre modèle est le fait que nous utilisons une constante pour déterminer le temps de sélection d'un item dans la liste. Cette sélection étant principalement un pointage, nous devrions calculer le temps de sélection d'un item en utilisant la loi de Fitts. Cependant les données collectées précédemment ne nous ont pas permis d'obtenir une droite de regression linéaire avec un bon coefficient de corrélation. De ce fait les constantes a et b que nous pourrions proposer seraient biaisées par cette mauvaise corrélation entre les données mesurées et la droite proposée.

Par conséquent, à défaut d'utiliser un modèle précis qui pourrait faire varier de quelques dizaines de millisecondes notre prédiction, nous avons choisi de prendre la moyenne des temps mesurés lors de la sélection d'item dans la liste.

IV.2. Une interface extensible

Notre interface remplit la fonction pour laquelle nous l'avons initialement conçue, à savoir modéliser rapidement un clavier avec une liste de prédiction et prédire les performances que nous pourrions avoir avec ce système modélisé.

Cependant, cet outil a un intérêt limité en l'état car il ne permet de modéliser que les claviers logiciels avec une liste de prédiction et utilisés avec un dispositif de pointage. Pour en faire un outil plus intéressant, il faudrait maintenant permettre de modéliser tout type de clavier logiciel : à la fois au niveau des interactions proposées (défilement automatique, geste, etc.) et de l'utilisation des résultats de prédiction (réagencement des caractères par exemple).

Ceci n'est pas le cadre de cette thèse. Il est au croisement avec les travaux de Bruno Merlin qui propose un langage de description de claviers logiciels (KeySpec). Ainsi, notre interface peut être vue comme une première brique d'une interface de construction de claviers logiciels et qui pourrait permettre de générer la description KeySpec du clavier. Une fois cette description obtenue, nous pourrions alors appliquer notre modèle de prédiction sur le

clavier généré et ainsi proposer une simulation des performances théoriques des claviers modélisés avec KeySpec.

Chapitre 4 - Comment optimiser l'utilisation de la liste de prédiction ?

Le chapitre 3 a mis en évidence qu'un des principaux problèmes des listes de prédiction, était qu'un regard en direction de la liste de prédiction était très coûteux en temps pour l'utilisateur, et ce même s'il ne déplaçait pas son curseur sur la liste. Nous avons également pu voir, qu'il effectuait plusieurs contrôles sur la liste sans pour autant toujours l'utiliser. Ceci est du au fait que pour pouvoir l'utiliser, il faut que la liste contienne le mot exact que l'utilisateur souhaite saisir. Si aucun des mots de la liste ne correspond à la suite de la saisie de l'utilisateur, alors ce dernier ne peut pas utiliser la liste et doit effectuer au moins la saisie du caractère suivant sur le clavier logiciel. Cependant, le temps de recherche sur la liste l'aura pénalisé de plusieurs centaines de millisecondes sans que cela ne lui apporte quoique ce soit.

Afin de diminuer le nombre de fois où l'utilisateur va regarder la liste sans pouvoir interagir avec, nous proposons dans ce chapitre, une variante de la liste de prédiction classique. L'idée est de permettre à l'utilisateur de ne sélectionner qu'une partie du mot. Ainsi, si le préfixe d'un mot de la liste peut lui permettre d'accélerer sa saisie, l'utilisateur pourra alors le sélectionner sans pour autant prendre la fin du mot qui ne l'intéresse pas.

Section I - Principe du système

Au lieu de voir chaque mot de la liste comme un seul item cliquable, nous proposons de considérer chaque caractère de chaque mot comme un objet à part entière. Chaque caractère a donc une représentation graphique propre et est un objet graphique cliquable. Quand un caractère est choisi, le système sélectionne tous les caractères qui se trouvent entre la racine du mot et le caractère sélectionné (cf. Figure 45). Ce principe correspondant à une structure sous forme d'arbre, nous avons nommé notre système « WordTree ».

Figure 45 : Interface de WordTree après la saisie de « bo »

Ainsi, lors de la saisie de l'utilisateur, si le mot désiré n'est pas dans la liste mais qu'un autre mot présent dans cette liste contient une partie du mot à saisir, l'utilisateur peut cliquer sur le caractère le plus avancé dans ce mot de manière à sélectionner la chaîne de caractères qui l'intéresse et lui permet de compléter au mieux sa saisie. Cette méthode devrait ainsi permettre d'accélérer la saisie de texte. Par exemple, si l'utilisateur souhaite écrire le mot « applicable », il commence à saisir au clavier les caractères 'a', 'p' et 'p'. Après ces trois premiers caractères, le mot « applicable » n'apparait pas encore dans la liste de prédiction. Cependant, le mot « application » est présent. L'utilisateur peut alors cliquer sur le deuxième 'a ' de « application » pour sélectionner la chaîne « applica » et l'insérer dans le texte. De cette manière, l'utilisateur ajoute quatre caractères supplémentaires à sa saisie et n'aura pas porté attention à la liste pour rien.

Section II - La liste de prédiction WordTree

II.1. Architecture

De manière à pouvoir faire évoluer facilement WordTree, nous avons conçu notre système de manière modulaire. En effet, nous l'avons divisé en trois modules principaux

communiquant entre eux au moyen du bus logiciel IVY [Buisson 2002]. Les trois modules sont :

- **Le clavier logiciel :** Pour notre étude, nous avons choisi un clavier AZERTY réduit ne comportant que les 26 caractères de l'alphabet latin et la barre espace. Cependant, pour pouvoir utiliser ce système quotidiennement, il faudrait le coupler à un clavier logiciel plus complet. Ceci est facilement réalisable à partir du moment où le nouveau clavier logiciel suit le protocole de communication mis en place pour discuter entre les agents.

- **Le système de prédiction :** celui-ci reçoit au fur et à mesure les caractères saisis par l'utilisateur. En fonction de ce qui a déjà été saisi, il propose les mots qui ont la plus grande probabilité de compléter la saisie de l'utilisateur. Il met à jour ses résultats après chaque saisie (sur le clavier ou la liste) de l'utilisateur. Ses résultats sont envoyés à la liste. De même que pour le clavier logiciel, le système de prédiction est interchangeable avec un autre, à partir du moment où ce dernier reçoit et envoie les informations comme indiqué dans le protocole.

- **La liste de mots :** celle-ci affiche les mots envoyés par le système de prédiction. Ces mots sont affichés les uns sous les autres. Chaque caractère est sélectionnable individuellement. Cependant, la sélection d'un caractère entraine la sélection de tous les caractères qui le précède dans le mot. De la même manière que la saisie d'un caractère sur le clavier, la sélection d'un ou plusieurs caractères sur la liste entraine leur envoi à la fois à l'application cible (par exemple, une messagerie ou un éditeur de texte) mais aussi au système de prédiction. Ce dernier les traite comme ceux venant du clavier logiciel et propose donc une nouvelle liste de mots à afficher.

L'originalité de notre système se situe dans l'interaction proposée sur la liste de mots. Cette modularité nous permet de pouvoir utiliser notre système avec un autre clavier logiciel que celui que nous proposons ici. Cela peut aussi nous permettre de choisir un autre système de prédiction.

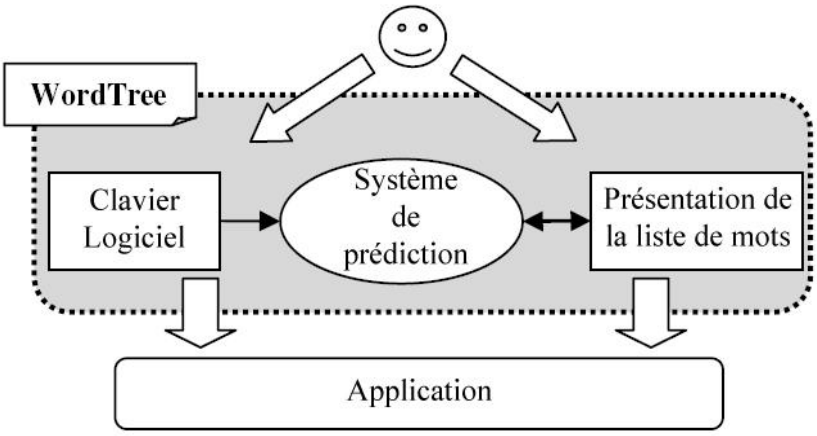


Figure 46 : Architecture de WordTree

II.2. Le système de prédiction

Afin de pouvoir prédire les mots qui pourraient venir compléter la saisie de l'utilisateur, WordTree est allié à un système de prédiction basé sur un arbre lexicographique. L'arbre est construit à partir d'un corpus de mots de plus de 130 000 mots. Chaque mot est représenté par un chemin de la racine à une feuille de l'arbre, et où chaque caractère est représenté par un nœud. Deux mots ayant le même préfixe utilisent le même chemin initial pour éviter de créer de nouveaux nœuds. Pour les mots qui sont des préfixes d'autres mots, une feuille de « fin mot » a été créée.

De plus, nous avons choisi la méthode utilisée dans [Ménier 2001]. Chaque transition entre deux noeuds de l'arbre est associée à un poids. A chaque mot ajouté, les transitions entre les différents nœuds traversés sont incrémentées de 1. Ainsi, les transitions des préfixes régulièrement utilisés ont un poids beaucoup plus élevé que celles des préfixes quasiment jamais utilisés. Les caractères qui terminent une transition dont le poids est élevé présentent plus de probabilité d'être les caractères saisis par l'utilisateur.

Pour effectuer son classement, ce système prend en compte le début du mot qui vient d'être entré : il traverse l'arbre en fonction de ce qui a déjà été écrit. Une fois positionné à la fin du préfixe déjà saisi, il classe les caractères qui peuvent suivre ce préfixe en fonction de leur

probabilité d'apparition (plus le poids d'une transition est grand, plus le caractère a la chance d'apparaître). Cette méthode est reproduite par récursivité sur les autres caractères de manière à obtenir les mots qui ont la plus forte probabilité.

Section III - Evaluations

III.1. Hypothèses

Si nous offrons à l'utilisateur la possibilité de saisir qu'une partie d'un mot, il va ainsi pouvoir utiliser plus souvent la liste de prédiction. Même au cas où mot désiré n'est pas saisi complètement, utiliser plus régulièrement la liste pour saisir de petites chaines de caractères devrait contribuer à réduire le nombre d'actions à réaliser pour saisir un mot. Notre première hypothèse est que le système WordTree devrait permettre de réduire le nombre d'actions moyen par caractère.

Notre seconde hypothèse est que si l'utilisateur commence à utiliser la liste de prédiction pour compléter son mot, et qu'il n'a pas pu finir son mot, il va automatiquement regarder de nouveau la liste pour finir sa saisie. Il ne repartira sur le clavier logiciel que s'il ne réussit pas à améliorer sa saisie sur la liste. De cette manière, nous espérons aussi réduire le nombre d'aller-retour entre le clavier logiciel et la liste. Cette réduction du nombre de déplacements devrait alors permettre d'augmenter la vitesse de saisie.

Pour vérifier nos hypothèses, nous avons procédé en trois étapes. Dans un premier temps, nous avons simulé le fonctionnement de notre système de manière à avoir de premières données théoriques nous permettant notamment de savoir si le système pouvait effectivement diminuer le nombre d'actions à réaliser. La seconde étape a consisté à expérimenter ce système avec un ensemble de personnes valides. Cette étape avait pour but de voir comment les utilisateurs interagissaient avec ce nouveau système, et si la nouvelle interaction proposée était bénéfique ou non par rapport à la liste de prédiction classique. Enfin, lors d'une dernière expérience, nous avons expérimenté ce système avec des personnes présentant un handicap des membres supérieurs.

III.2. Evaluation théorique

Avant de tester notre système avec des utilisateurs, nous avons donc voulu déterminer, « *a priori* », les performances de notre système par la réalisation d'une simulation par ordinateur. Ceci consiste à simuler la saisie d'un ensemble de mots du dictionnaire en prenant à chaque fois la solution de saisie la plus avantageuse en vue de performances de saisie.

III.2.a. Protocole

Procédure

Afin de vérifier la pertinence de notre système, nous avons calculé ses performances théoriques optimales, et nous les avons comparées à celles d'un système classique où l'utilisateur serait obligé de choisir le mot entier dans la liste. Cette simulation consiste à simuler, via un algorithme, la saisie de l'ensemble des mots de l'officiel du Scrabble qui est composé de 364 370 mots français (soit 3 641 140 caractères). Cette simulation, théorique, est dite « idéale » car les résultats de prédiction du système WordTree ou de la liste classique de prédiction sont utilisés dès que possible. Notons que WordTree et le système classique utilisent le même algorithme de prédiction décrit précédemment.

Variables mesurées

Notre évaluation théorique se base sur deux variables présentées au Chapitre 2 :

- D'une part le KeyStroke Per Character (KSPC) [MacKenzie 2002b], qui est le nombre d'actions nécessaires, en moyenne, pour saisir un caractère utilisant une technique de saisie. Cette métrique est calculée en divisant le nombre total d'actions de l'utilisateur par le nombre total de caractères à saisir. Le KSPC peut être défini comme suit :

$$\mathrm{KSPC} = \frac{\text{nombre total d'opérations}}{\text{nombre de caractères}}$$

A noter que le KSPC est aussi utilisé pour mesurer le taux d'erreur effectuée par l'utilisateur pendant sa saisie. Il permet en effet de prendre en compte, dans les actions les saisies erronées de l'utilisateur et les actions de correction de ce dernier. Les erreurs corrigées sont invisibles à la fin, mais font quand même augmenter le KSPC du fait des actions supplémentaires à réaliser. Dans le cas de notre simulation, celle-ci étant « parfaite », il n'y aura pas d'erreur de saisie ni de corrections. Le KSPC obtenu nous donnera donc comme seule indication le nombre d'actions minimum à réaliser en moyenne pour saisir un caractère.

- D'autre part, nous mesurons le Hit Rate (HR) [Fazly 2003], qui est le pourcentage d'apparition du mot désiré dans la liste. Puisque notre système permet à l'utilisateur de sélectionner une partie d'un mot prédit, nous redéfinissons le HR comme étant le nombre de fois où une chaîne de caractères (de plus de un caractère) apparait dans la liste et pourrait venir compléter la saisie de l'utilisateur.

III.2.b. Résultats

Les résultats de cette simulation montrent une réduction importante du nombre moyenne d'actions à effectuer pour saisir un caractère (cf. Figure 47). En considérant une saisie de texte idéale, sans erreur et en utilisant au mieux le système de prédiction proposé, nous pourrions n'avoir à faire que 0,71 actions à faire pour saisir un caractère alors qu'une liste de prédiction classique nécessite 0,92 actions par caractère. Ainsi une diminution théorique de 22% est obtenue en utilisant le nouveau système. En comparaison, un clavier logiciel standard nécessite, lui, une action par caractère.

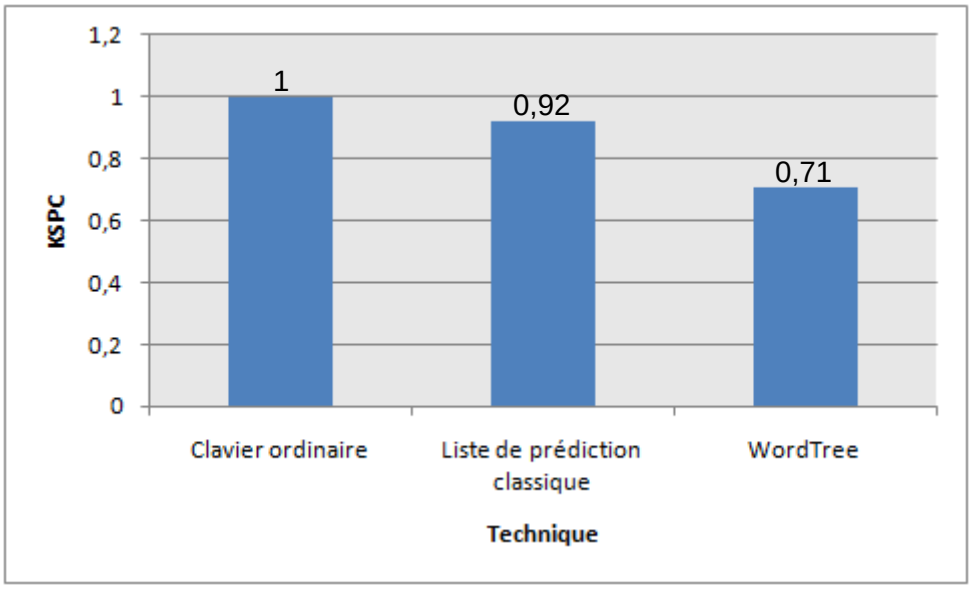


Figure 47: KSPC théorique en fonction du dispositif utilisé

Cette réduction du nombre d'actions est du au fait que la liste de prédiction peut être beaucoup plus utilisée avec notre système. La Figure 48 montre que grâce à notre système, le taux d'utilisation de la liste passe ainsi de 3,7% avec l'interaction classique à 18,7% avec notre système. Le taux d'utilisation augmente alors de 15%.

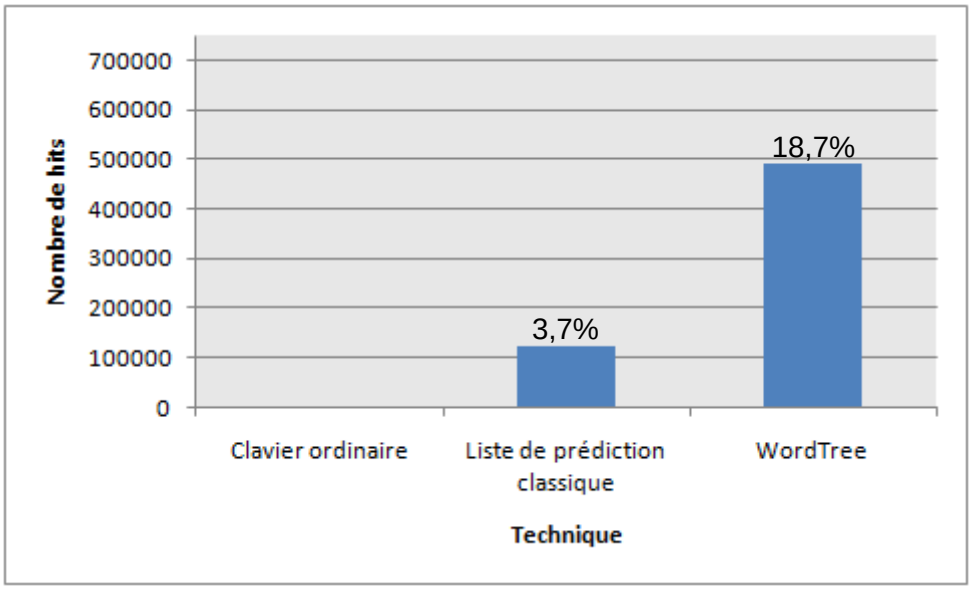


Figure 48: Comparaison de nombre de hits pour les 3 systèmes

Cette simulation confirme notre hypothèse principale : la sélection de n'importe quel caractère des mots de la liste de prédiction permettrait d'utiliser plus régulièrement les résultats de la liste. Même si l'utilisation de la liste ne permet pas de saisir le mot complet, elle permet de saisir plusieurs caractères d'un coup.

Théoriquement, notre système permet donc de diminuer le nombre d'actions à réaliser pour saisir du texte. Par conséquent, nous pouvons espérer augmenter la vitesse de saisie de texte avec notre système par rapport à une liste de prédiction classique. Notre simulation ne calculant que le nombre d'actions, elle ne prend pas en compte les aspects moteurs et cognitifs qui peuvent intervenir lors de la saisie. Nous avons donc réalisé une expérience utilisateur pour évaluer la pertinence de notre système et son efficacité sur les performances de l'utilisateur.

III.3. Evaluation avec des personnes valides

III.3.a. Protocole

L'expérience a été menée sur un ordinateur portable Toshiba fonctionnant sous Microsoft Windows XP, avec 1 Go de mémoire vive. Les sujets, au nombre de 24, utilisaient une souris ordinaire pour interagir avec le clavier logiciel.

Dans cette expérience, nous avons limité notre choix à un clavier logiciel qui ne contient que les 26 caractères de l'alphabet latin et la barre d'espace. Nous avons choisi le clavier QWERTY avec lequel les utilisateurs étaient familiers.

L'ensemble du logiciel comprenant le clavier, le système de prédiction, les listes classiques et WordTree a été réalisé en Java SE 6 (cf.Figure 49). Le recueil des données a été réalisé avec la plate-forme E-Assiste [Raynal 2005a]. Pour chaque session d'expérimentation, l'ensemble des données recueillies était enregistré dans un fichier en respectant un formalisme XML.

Les sujets étaient divisés en 2 groupes de 12 personnes chacun. Chaque groupe avait à accomplir deux exercices de copie sur le clavier QWERTY[15]: le premier avec la liste classique et le second avec WordTree. Les sujets pouvaient choisir entre la copie de 22 mots (soit 157 caractères) ou de 41 mots (218 caractères en total) selon le temps qu'ils avaient à accorder à l'expérimentation. Les sujets avaient les mêmes mots pour les deux exercices.

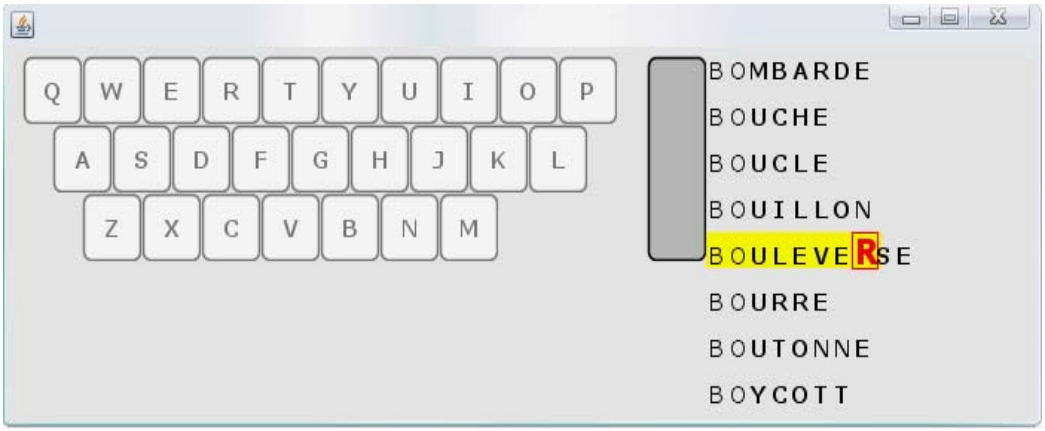


Figure 49: Clavier logiciel utilisé dans l'évaluation

La tâche demandée aux participants était de saisir le plus rapidement possible et en évitant au maximum les erreurs de saisie. Les mots à recopier étaient présentés sur une ligne et les mots recopiés par les sujets apparaissaient sur une ligne en dessous (cf. Figure 50). Pour minimiser le problème de focus d'attention, les erreurs de frappe n'étaient pas affichées. A la place, il y avait un retour sonore et visuel qui se répétait tant que le bon caractère n'était pas saisi. A la fin de chaque mot, l'utilisateur devait appuyer sur la barre espace.

Des consignes et des explications étaient présentées avant le début de chaque test pour faire comprendre aux utilisateurs le principe du système et le déroulement des évaluations.

[15] La disposition QWERTY est la disposition de caractères habituellement utilisée au Liban, lieu où les évaluations se sont déroulées.

Deux petites phases d'apprentissage précédaient l'expérience. Ces phases permettaient aux utilisateurs de se familiariser avec les claviers logiciels et aux listes de prédiction proposées.

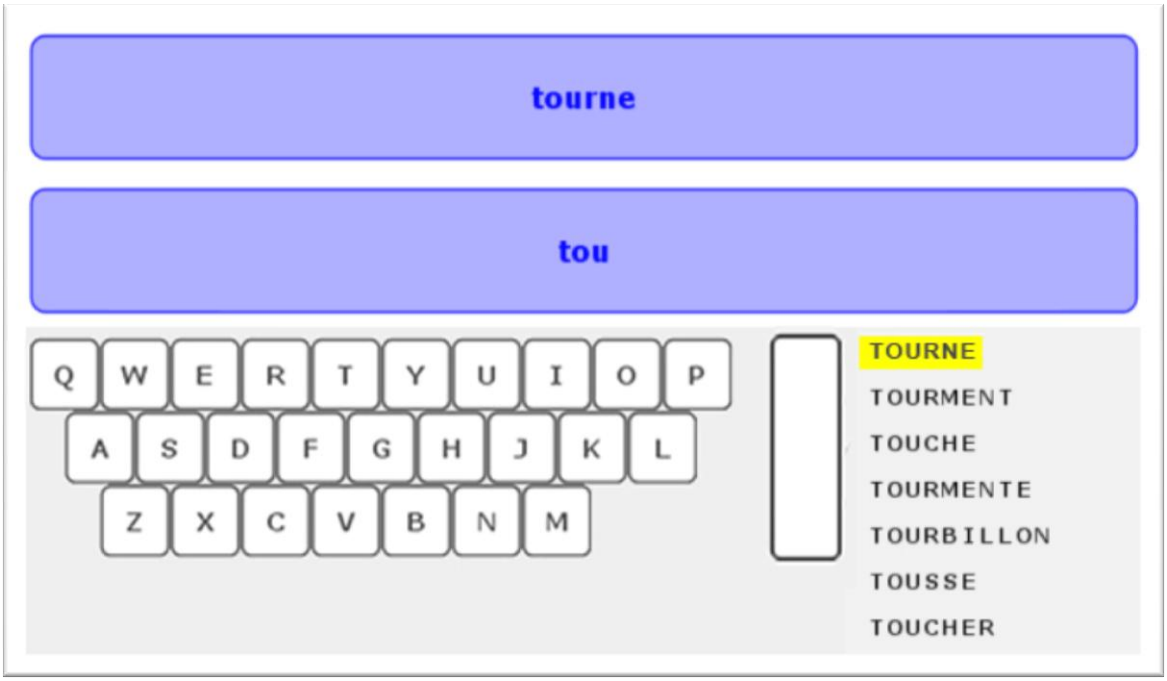


Figure 50: Ecran de l'évaluation avec la liste classique

En plus des deux métriques déjà présentés pour la simulation, deux autres étaient évaluées dans ces expérimentations :

- Le nombre de caractères saisis par seconde, CPS (« Characters Per Second »), calculé en divisant le nombre de caractères saisis par le temps mis pour faire entrer tous les caractères.
- Le taux d'économie de saisie, KSR (« Keystroke Saving Rate »), calculé en soustrayant le nombre de clics sans liste du celui en utilisant la liste puis en divisant le tout sur le nombre de clics faits sans utiliser la liste.

III.3.b. Résultats

Voici la présentation des résultats obtenus à partir des données recueillies lors des expérimentations des sujets qui avaient réalisé l'évaluation jusqu'au bout et dans de bonnes conditions. Les résultats statistiques présentés dans cette section ont été obtenus par des tests paramétriques d'analyse de la variance (ANalysis Of Variance, ANOVA). Ces traitements ont été réalisés avec l'API développé par S. MacKenzie [MacKenzie].

Vitesse de saisie

Les résultats de cette expérimentation confirment nos hypothèses. Les utilisateurs ont eu moins d'actions à faire avec le système WordTree qu'avec la liste de prédiction classique. La Figure 51 montre les résultats du KSPC pour les participants ayant saisi 22 ou 41 mots et ce avec les deux dispositifs. Dans les deux cas, nous pouvons voir que le KSPC est moins élevé avec le système WordTree qu'avec la liste classique. Les mots ne sont pas les mêmes dans les deux cas (22 ou 41 mots) : ceci peut expliquer la différence importante qu'il peut y avoir entre les deux jeux de données. (Diminution du KSPC de 9% pour les utilisateurs saisissant 41 mots et 25% pour l'autre cas.)

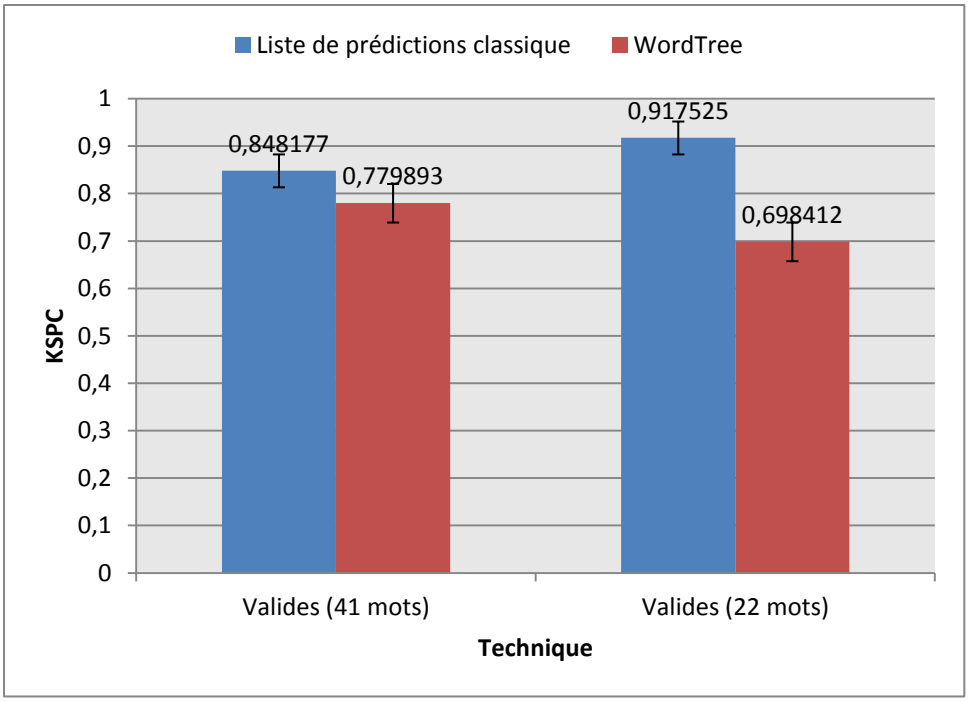


Figure 51: Comparaison du KSPC de WordTree par rapport à la liste classique dans les 2 exercices

Cette diminution du KSPC s'est aussi traduite par une augmentation de la vitesse de saisie de texte (cf. Figure 52). Ce résultat est important car il montre que notre système, même en diminuant le nombre d'actions effectuées par l'utilisateur, ne détériore pas la saisie de l'utilisateur mais au contraire lui permet de l'améliorer.

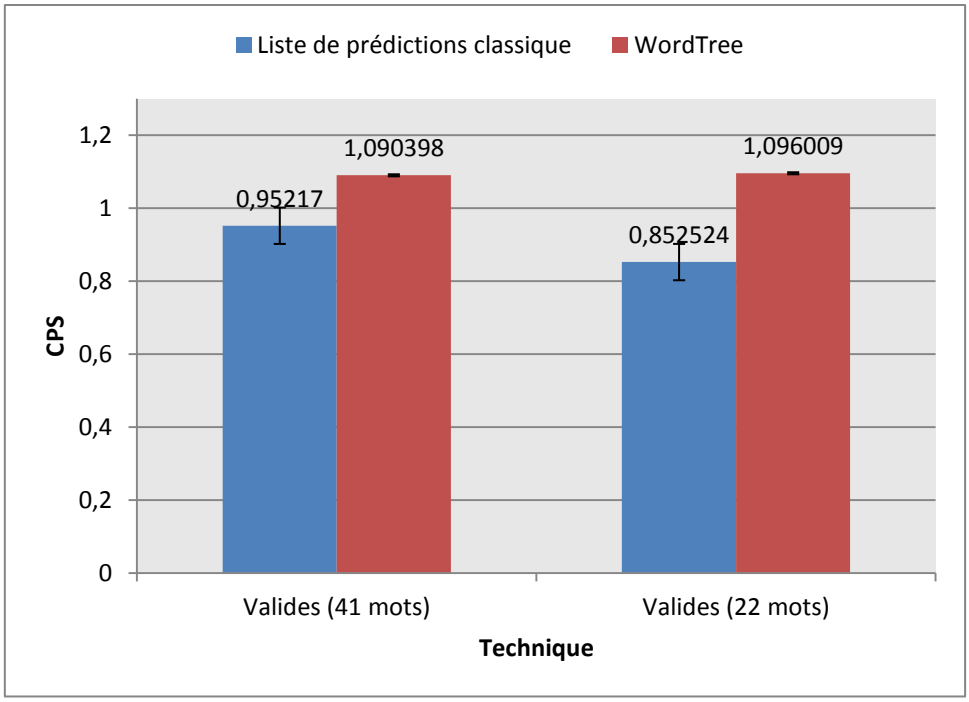


Figure 52: Comparaison du nombre de caractères saisis par seconde avec les deux dispositifs

Ces résultats montrent une augmentation significative de 14% et 28% dans le cas de 41 et 22 mots respectivement.

Précision

En revanche, un bémol peut être mis à ces résultats : le taux d'erreur est supérieur avec notre système pour les utilisateurs qui ont saisi 41 mots avec les deux dispositifs.

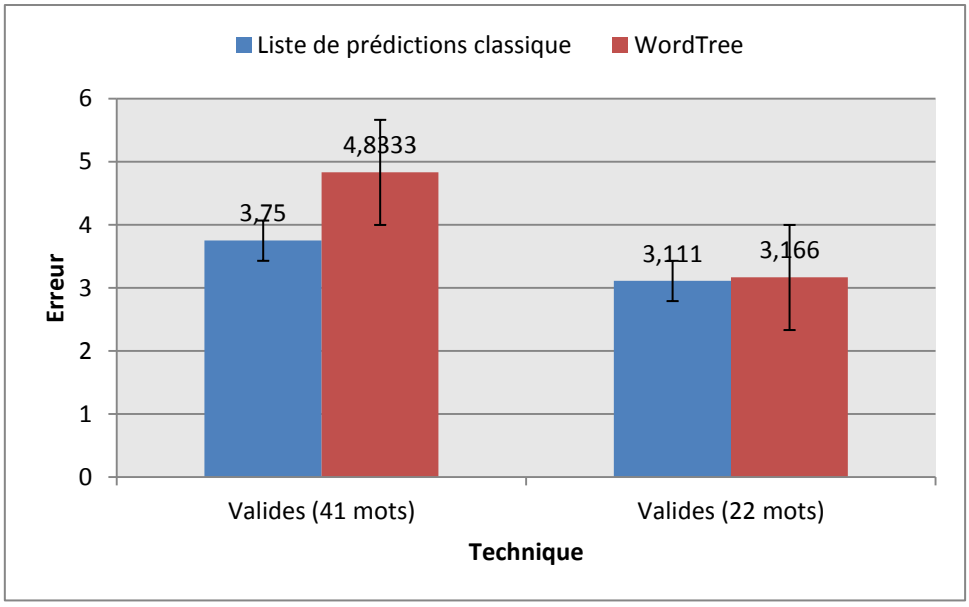


Figure 53: Nombre moyen des erreurs commises avec les deux dispositifs

III.4. Evaluation avec des personnes en situation d'handicap moteur

Les claviers logiciels couplés à une liste de prédiction sont souvent utilisés par des personnes souffrant d'une faible motricité des membres supérieurs. C'est pourquoi nous avons choisi de réaliser une deuxième expérimentation avec ces personnes de manière à conforter la validité de notre système.

III.4.a. Protocole

Pour cette évaluation, le même protocole a été utilisé qu'avec les personnes valides. Le clavier logiciel, les listes de prédiction et l'affichage des mots étaient identiques. La population était constituée de 8 sujets souffrant d'un handicap des membres supérieurs. Ils étaient tous familiers avec l'utilisation d'ordinateur et manipulaient généralement un clavier logiciel QWERTY. Pour les sujets à mobilité restreinte des membres supérieurs n'ayant pas tous la même motricité, les dispositifs de pointage étaient variés: trois sujets utilisaient une souris classique, et les cinq autres pointaient avec un trackball. Chaque sujet exécutait l'évaluation sur son propre ordinateur et avec son dispositif de pointage personnel.

Dans cette expérimentation, les sujets avaient à recopier 30 mots (soit 215 caractères). Les mots étaient utilisés dans les 2 exercices : une fois avec une liste classique et une autre en utilisant WordTree. A la différence de l'expérimentation précédente, chaque sujet devait répéter ces deux exercices pendant 10 sessions et sur une période de 8 à 10 jours. Deux sessions successives devaient être espacées d'au moins 3 heures et pas de plus de 2 jours.

Les variables analysées étaient les mêmes que dans l'expérimentation précédente.

III.4.b. Résultats

Les résultats que nous avons obtenus lors de cette deuxième expérimentation viennent confirmer ceux de la première évaluation. La Figure 54 montre que le système WordTree a permis de diminuer de 19,7% le nombre d'actions réalisées. Le KSPC de WordTree n'est que de 0,66 lorsque celui de la liste de prédiction est de 0,825.

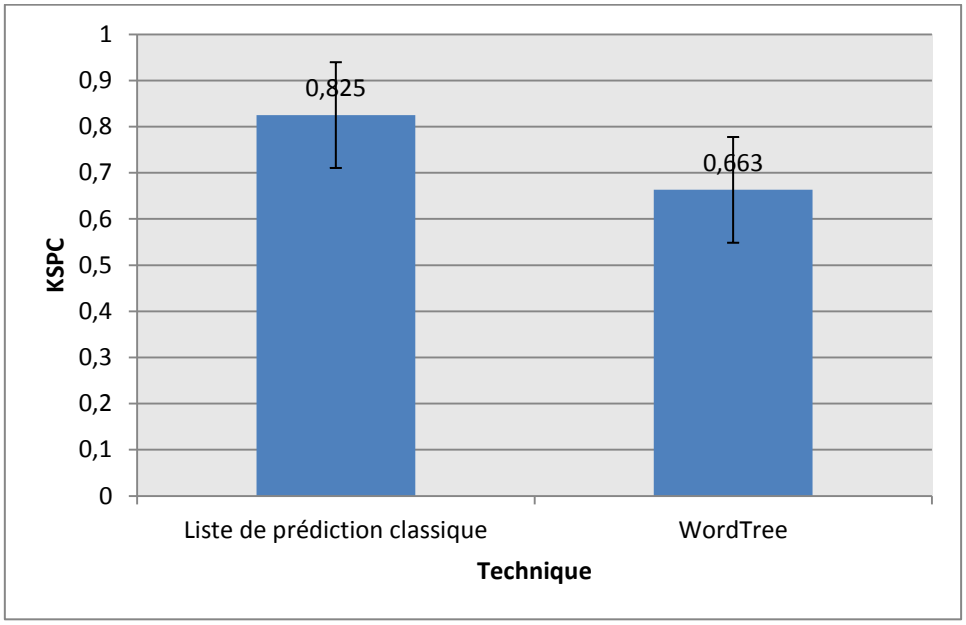


Figure 54: Diminution de KSPC avec WordTree

La réduction du KSPC provoque ainsi une augmentation de la vitesse de saisie avec WordTree. Les participants ont entré en moyenne 0,566 caractères par seconde en utilisant une liste classique et 0,604 caractères par seconde avec WordTree (cf. Figure 55). Le gain en vitesse est alors de 6.7% comparé à la liste ordinaire ($F_{1,158}$=7,7142, p=0,00614).

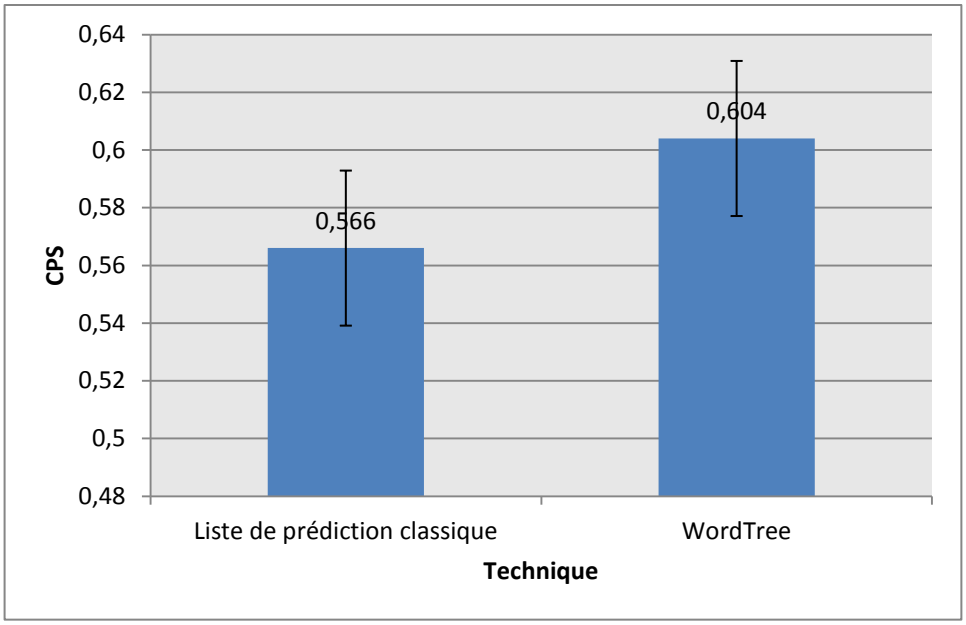


Figure 55: Augmentation du CPS de WordTree % à la liste classique

La répétition des sessions a montré que les participants évoluaient de la même manière avec les deux dispositifs évalués (cf. Figure 56). Leurs performances de saisie augmentent pour les deux systèmes. On peut penser que cette augmentation est due à l'apprentissage des mots qui étaient les mêmes pour les dix sessions. Au bout de 10 sessions, il est possible que les participants sachent quand les mots apparaissaient dans la liste et donc puissent un peu plus anticiper leur saisie vers la liste. Néanmoins, l'écart entre les deux dispositifs reste inchangé au cours des 10 sessions. On peut donc imaginer que les performances resteront à l'avantage de Wordtree même dans un usage quotidien.

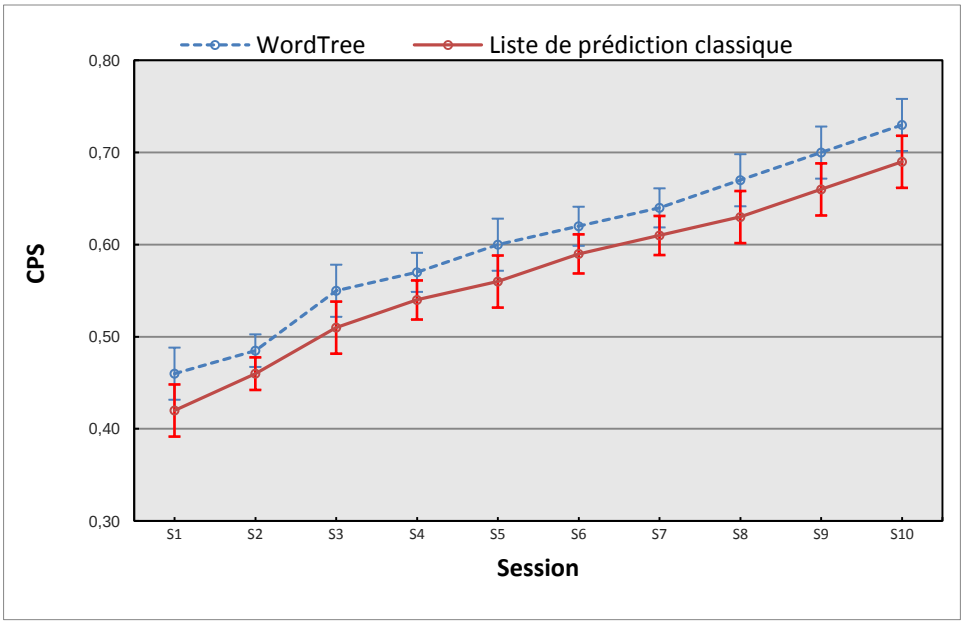


Figure 56: Vitesse de saisie par session et technique d'interaction

Enfin, à la différence de la première expérimentation, les huit participants ont réalisé en moyenne sur les 10 sessions moins d'erreurs avec le système WordTree. Les participants ont réalisé moins de 3% d'erreurs avec les deux dispositifs (cf. Figure 57) ce qui est dans la normale. On peut donc penser que le taux élevé d'erreurs lors de la première expérimentation était soit du à une trop forte précipitation des participants ou alors à un temps d'apprentissage nécessaire pour bien prendre en main la nouvelle interaction.

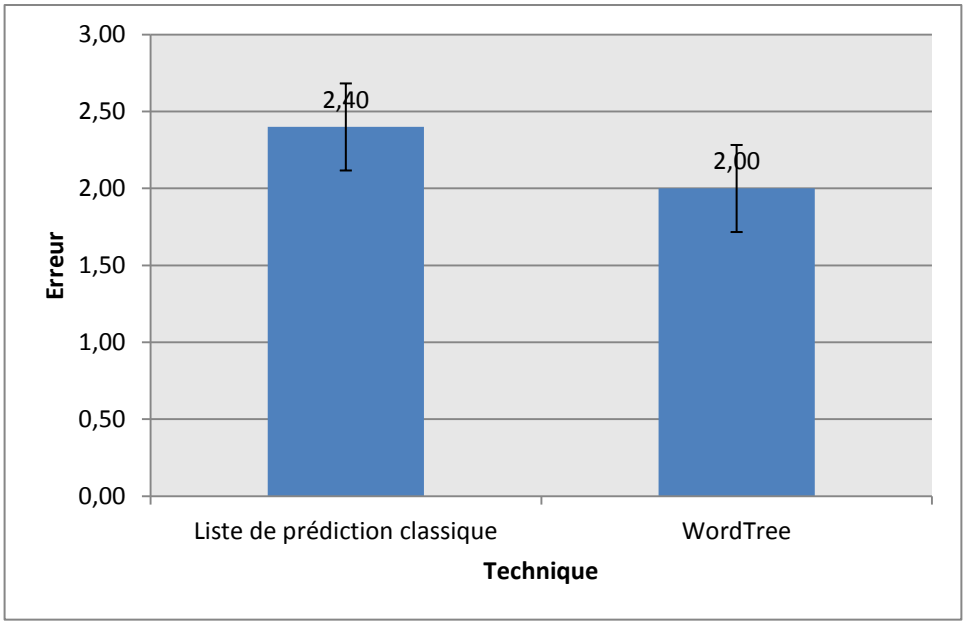


Figure 57: Diminution du taux d'erreur avec WordTree

III.5. Synthèse

L'interaction que nous proposons permet bien d'améliorer les performances de saisie avec une liste de prédiction. L'utilisateur se sert plus régulièrement de la liste de prédiction et par conséquent, diminue de manière significative le nombre d'actions à réaliser pour saisir du texte. Notre système étant basé sur la sélection de caractères plutôt que de mot, il nécessite le pointage d'une cible plus petite. En effet, les cibles à atteindre sont des touches englobant les caractères des mots alors que dans une liste classique, les items sont des touches englobant le mot complet. Donc dans le second cas, la taille de la touche est bien plus grande que dans notre cas. Cette sélection de cibles plus réduites aurait pu présenter un problème de vitesse d'exécution ou de précision car comme l'indique la loi de Fitts, plus la tâche de pointage est difficile plus le temps de pointage est long. Cependant, les expériences ont montré que malgré la plus grande précision à avoir dans la sélection des cibles sur la liste de prédiction, les utilisateurs étaient quand même plus rapides pour saisir du texte avec WordTree qu'avec la liste de prédiction classique.

Enfin, les petites cibles sont généralement un problème pour les personnes en situation d'handicap car leur manque de précision dans les mouvements les empêche de sélectionner

facilement ce type de cible. Cependant, les cibles que nous leur avons proposées n'ont posé de problème ni en précision (puisqu'ils ont réalisé moins d'erreurs avec WordTree qu'avec la liste de prédiction classique) ni en rapidité (car ils ont aussi été plus rapides avec WordTree).

Chapitre 5 - Comment focaliser l'attention de l'utilisateur sur la liste ?

Dans la plupart des systèmes d'aide à la saisie reposant sur une liste de prédiction, les mots prédits se situent soit à droite ou à gauche du clavier [Guenthner 1992], soit en haut ou en bas des touches [Tzimas] (cf. Figure 58 à gauche en bas). La liste est généralement « délocalisée » par rapport au clavier, ce qui cause certainement des problèmes d'attention à l'utilisateur qui est obligé de déplacer son regard pour pouvoir prendre connaissance des mots affichés dans la liste. POBox, conçu par [Masui 1999], propose la liste des mots à l'endroit où l'utilisateur vient de cliquer, là où son attention est portée à ce moment. Dans ce cas, l'utilisateur peut regarder plus rapidement les mots. En revanche, ce type d'affichage cache une partie du clavier, ce qui peut être contraignant dans le cas où le mot n'est pas dans la liste et que le caractère à saisir se trouve sous la liste.

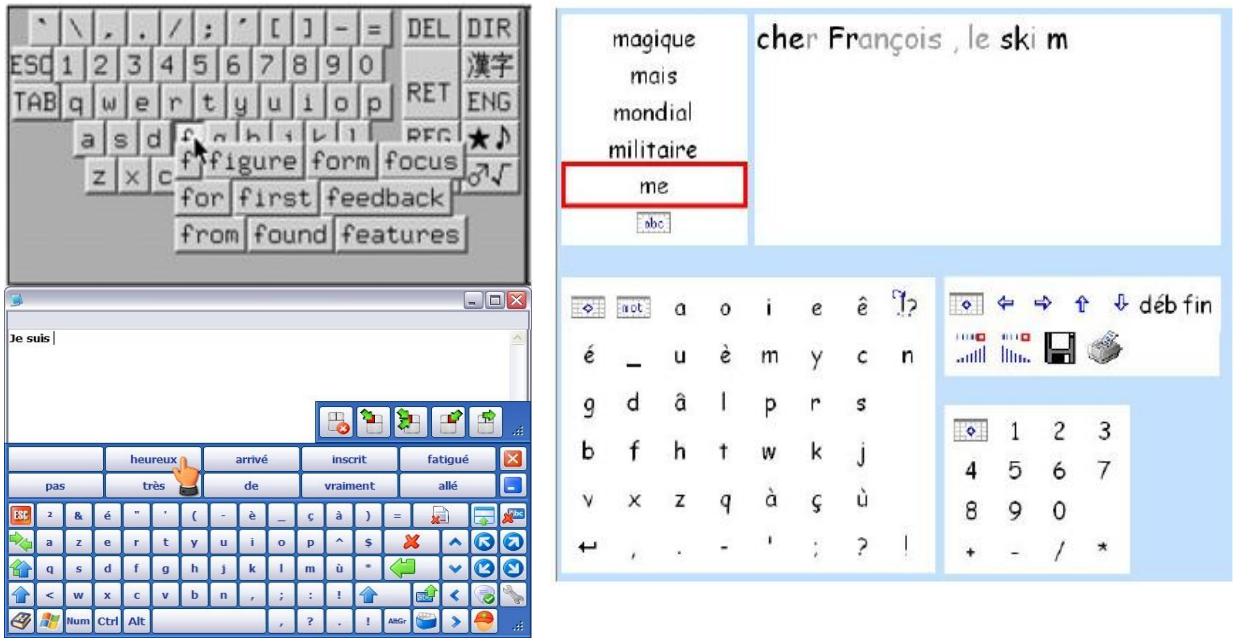


Figure 58: Exemples de claviers logiciels présentant une liste de mots prédits : A gauche en haut le clavier POBox [Masui 1999], en bas le clavier de [Tzimas], à droite le clavier SibyMot [Schadle 2004]

Section I - Problématique

Le problème est de savoir comment mettre la liste de prédiction au centre de l'attention de l'utilisateur sans pour autant masquer le clavier. Disperser le clavier autour de la liste pose plusieurs problèmes :

- D'une part, l'utilisateur est habitué à utiliser un clavier de type AZERTY (ou QWERTY) et connait la disposition des caractères. Cette connaissance lui permet d'anticiper des déplacements et ainsi d'accélérer sa saisie. Si la liste vient s'insérer au milieu du clavier, il faudrait alors garder une disposition de caractères que l'utilisateur pourrait facilement retrouver.

- D'autre part, insérer la liste de prédiction au milieu du clavier écarterait encore un peu plus les touches les plus éloignées. Cette solution augmenterait la distance à parcourir sur le clavier logiciel. Sur le long terme, cette solution risquerait d'être plus fatigante pour les personnes présentant un handicap moteur au niveau des membres supérieurs.

Section II - Le clavier logiciel CentraList

Notre objectif était de placer la liste de prédiction au centre de l'attention de l'utilisateur sans pour autant cacher les touches du clavier logiciel. Pour cela, nous proposons de positionner les touches du clavier tout autour de la liste de manière à ce que tous les caractères soient à proximité de la liste (cf. Figure 59). Cette disposition de touches fait perdre tout repère par rapport au clavier AZERTY classique : c'est pourquoi, nous avons décidé de disposer les caractères par ordre alphabétique tout autour de la liste. Cette disposition a l'avantage de donner plus facilement des repères à l'utilisateur. Comme la touche espace est la plus utilisée lors de la saisie de texte, nous en avons ajouté quatre comme dans [MacKenzie 1999a]. Ces barres espace sont positionnées sur chaque côté du clavier.

L'architecture utilisée est la même que pour WordTree où le système est divisée en trois modules indépendants : clavier logiciel, liste de prédiction et système de prédiction. La liste de prédiction WordTree présentée au chapitre précédent étant plus performante que la liste

classique, nous avons utilisé celle-ci pour ce clavier nommé Centralist. De même, nous avons aussi repris le système de prédiction par arbre lexicographique. Par conséquent, seul le clavier logiciel est modifié par rapport au système précédent.

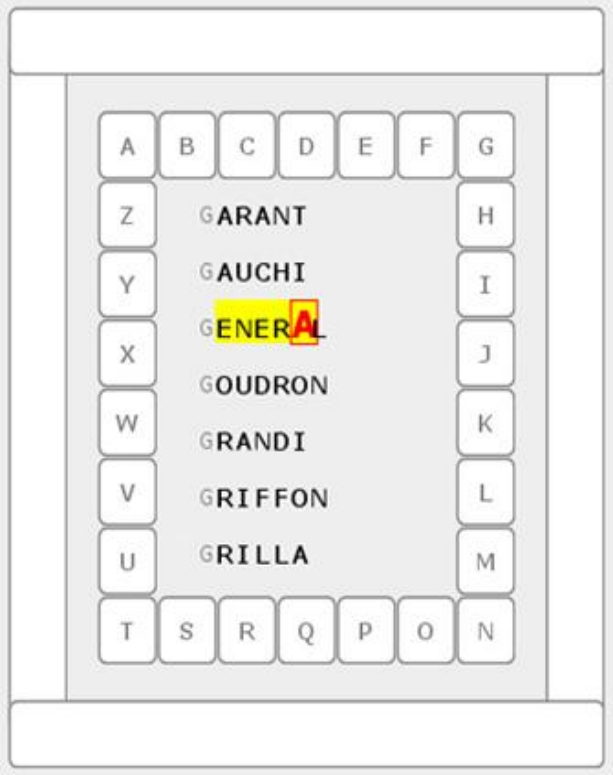


Figure 59 : Disposition des touches sur le clavier centralist

Section III - Evaluations

III.1. Hypothèses

Notre principale hypothèse est qu'en ayant la liste au centre du clavier, l'utilisateur regardera plus souvent et surtout plus rapidement la liste des mots proposés. Nous avons vu au Chapitre 2 que chaque regard sur la liste de prédiction pénalisait l'utilisateur d'un temps assez conséquent (environ 325 ms). Si le temps de recherche dans la liste n'est pas à négliger, nous pensons qu'une partie de ce temps est du au déplacement du regard du clavier à la liste et de la liste au clavier. En insérant la liste au milieu du clavier, nous

pensons donc que ce temps de déplacement sera réduit et que par conséquent le temps de recherche dans la liste sera lui aussi réduit.

Notre seconde hypothèse est que ce positionnement de la liste devrait aussi permettre à l'utilisateur de réduire les distances à parcourir avec le curseur du dispositif de pointage. Cette réduction des distances devrait, « a priori » (selon la loi de Fitts), augmenter la vitesse de saisie de texte

.

III.2. Evaluation

III.2.a. Protocole

L'évaluation a été menée sur un ordinateur portable de marque Toshiba, fonctionnant sous Microsoft Windows 7, avec 2.5 Go de mémoire vive.

Comme dans l'évaluation du premier système, nous nous sommes limités à un clavier ne contenant que les 26 caractères de l'alphabet latin et la barre espace. Les claviers étaient développés avec JAVA SE. L'architecture est similaire à celle du clavier contenant WordTree mais la disposition des touches et de la liste est différente.

Pour tester l'effet de la distribution des lettres du clavier CentraList, nous avons comparé CentraList à un clavier alphabétique ordinaire, dont les touches sont réparties sur trois lignes comme pour le clavier AZERTY (cf.Figure 60). Ce clavier alphabétique était lui aussi couplé au système WordTree disposé à droite du clavier.

Figure 60: Le clavier alphabétique utilisé dans l'évaluation

Cette évaluation a été réalisée en deux parties. D'une part avec 18 participants volontaires ne présentant pas de handicap moteur (participants que nous nommerons participants valides lors de la présentation des résultats) ; et d'autre part, 7 participants qui ont un handicap moteur au moins au niveau des membres supérieurs). Tous utilisaient une souris ordinaire comme dispositif de pointage sauf un des participants avec des déficiences motrices qui utilisait un joystick.

La plateforme d'évaluation utilisée est la même que celle prise pour les évaluations concernant WordTree (cf. Figure 61). La tâche proposée était la même, à savoir saisir le plus rapidement et précisément possible les phrases qui apparaissaient sur l'interface.

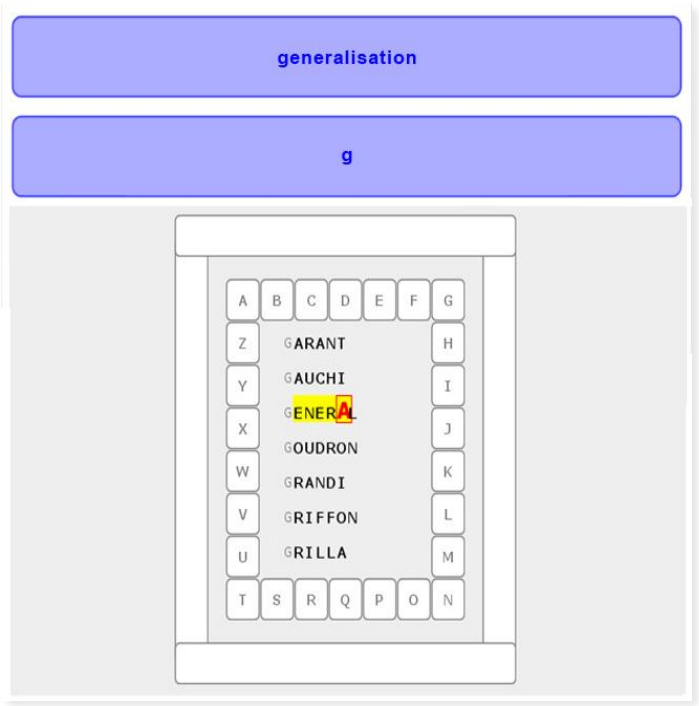


Figure 61: Plateforme d'évaluation de CentraList

Les utilisateurs valides avaient à saisir 25 phrases courtes (soit 984 caractères) et les sujets déficients moteurs n'en avaient que 15 à entrer (soit 460 caractères à saisir). Ces phrases étaient à saisr avec les deux claviers. Les participants de chaque type de population ont été répartis dans deux groupes. Un groupe commençait par le clavier alphabétique, et faisait

ensuite l'exercice avec Centralist. L'autre groupe faisait les exercices dans l'ordre inverse pour ainsi éviter les effets d'apprentissage et réduire les effets de la fatigue.

Les métriques mesurées dans ces expérimentations sont :

- le nombre de caractères par seconde (CPS),
- le nombre de touches nécessaires pour saisir un caractère (KSPC),
- le taux d'utilisation de la liste (HR),
- le nombre d'erreurs,
- la distance moyenne parcourue par le curseur durant la saisie.

III.2.b. Résultats

Les résultats sont sensiblement différents entre les personnes valides et les personnes à faible motricité. Cependant, les différences trouvées entre les deux claviers évalués sont les mêmes pour l'ensemble des participants. C'est pourquoi nous avons choisi de présenter conjointement les résultats des deux populations.

En premier lieu, l'analyse des distances parcourues par le pointeur montre que les participants ont eu moins de distance à parcourir en moyenne avec Centralist qu'avec le clavier alphabétique. En effet, la distance moyenne entre 2 opérations est de 336.78 pixels pour les participants valides sur le clavier classique, tandis qu'elle n'est que de 304.40 sur CentraList. Cette réduction de près de 10% de la distance est significative ($F_{1,34}$ = 14.846, p = 0.001 < 0.05). Les distances parcourues par les participants à faible motricité sont plus élevées (respectivement 460,54 pixels et 353,51 pixels pour le clavier alphabétique et Centralist) mais le gain escompté avec Centralist est bien plus important (environ 23%). Dans les deux cas, ces résultats viennent donc confirmer notre seconde hypothèse.

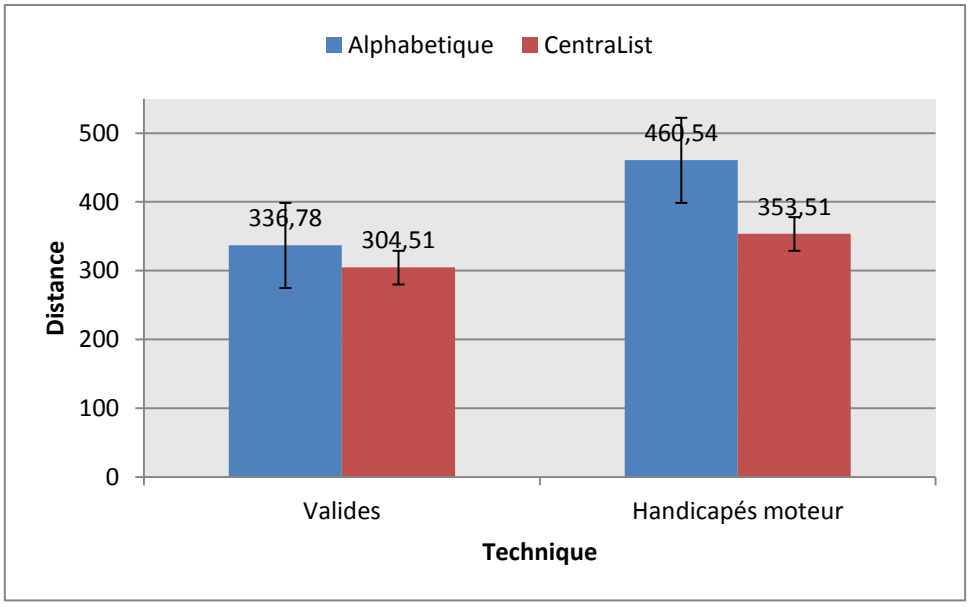


Figure 62 : Diminution de la distance entre 2 opérations avec CentraList.

La position de la liste au centre du regard de l'utilisateur a comme effet d'augmenter l'utilisation de la liste par les utilisateurs. En effet, l'analyse du taux d'utilisation de la liste dans CentraList confirme notre hypothèse. Les participants ont sélectionné des mots de la liste avec CentraList plus qu'avec l'autre clavier. Ce constat est schématisé dans la Figure 63.

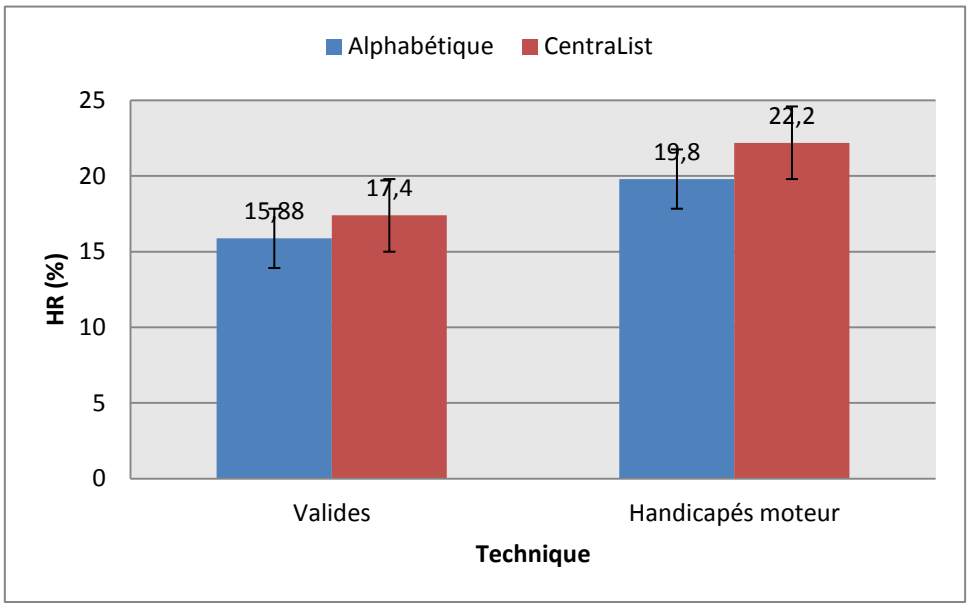


Figure 63: Augmentation du taux d'utilisation de la liste

D’autre part, l’analyse du nombre d’actions effectuées pour saisir un caractère est aussi en accord avec notre hypothèse. L’utilisation davantage de la liste permet de réduire le nombre d’actions nécessaires pour saisir un caractère. Les participants ont réalisé moins d’actions avec CentraList (0,82 pour les personnes valides et 0,77 pour les personnes avec un handicap moteur) qu’avec le clavier Alphabétique classique (0,84 pour les participants valides et 0,8 pour les autres participants).

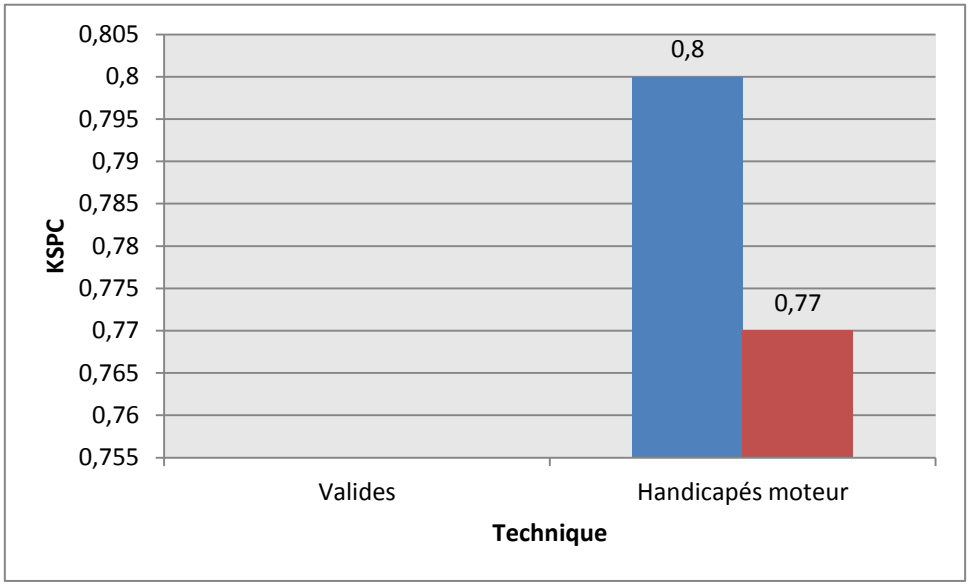


Figure 64 : Nombre d'actions par caractère

La liste de prédiction et le système de prédiction étant les mêmes pour les deux claviers, cela veut donc dire que les participants ont eu plus souvent recours à sélectionner de la liste avec Centralist que sur le clavier alphabétique avec la liste sur le côté.

Ces deux facteurs (diminution des distances et augmentation de l’utilisation de la liste) ont eu un effet bénéfique sur la vitesse de saisie de texte. Les participants valides ont saisi en moyenne 0,90 caractères par seconde avec Centralist contre 0,81 avec le clavier alphabétique, soit une augmentation de 11,1%. Pour les participants en situation d’handicap, la vitesse est passée de 0,46 caractères par seconde à 0,52 avec Centralist soit une hausse de 13%.

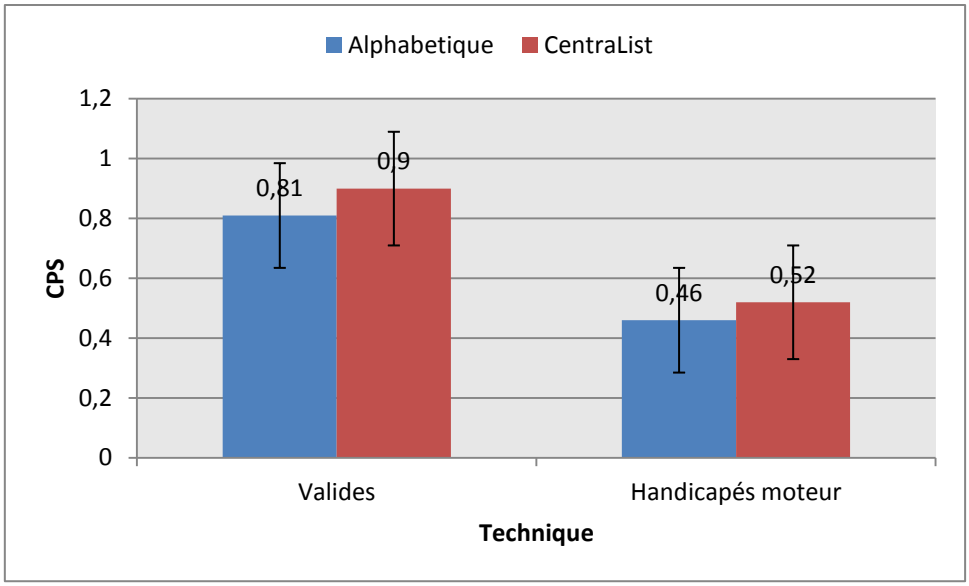


Figure 65 : Vitesse de saisie

Enfin, les deux systèmes présentent des taux d'erreurs normaux (cf. Figure 66). Tous les taux d'erreurs sont inférieurs à 3% et sont à peu près similaires pour les deux systèmes. Les participants valides effectuent un peu moins d'erreurs (1,76% avec Centralist et 1,93% avec Alphabétique) que leurs homologues déficients moteurs (2,45% et 2,58%). Ceux-ci peuvent être dus à la petite taille des touches à atteindre qui peut être source d'erreur de pointage pour les personnes dont la mobilité du pointeur est moins précise.

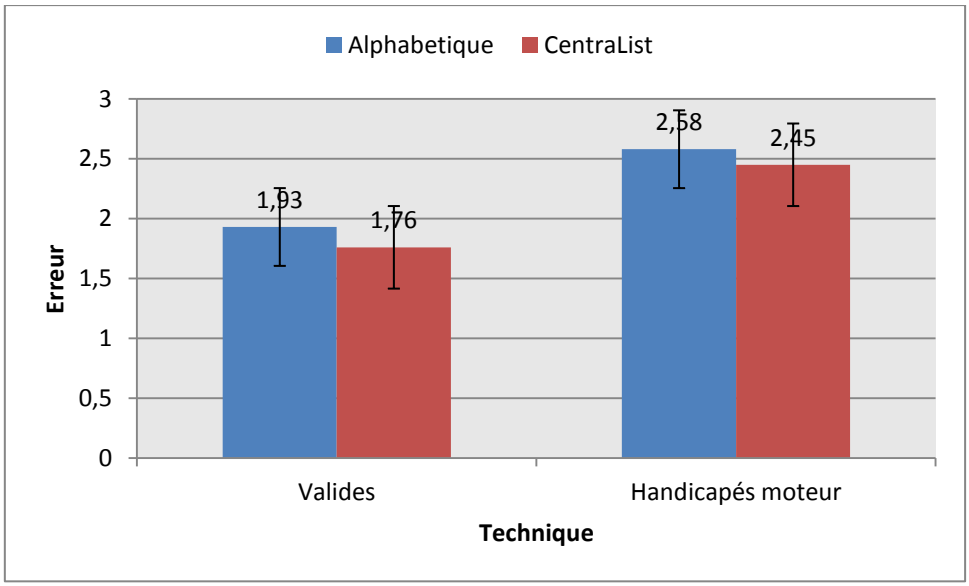


Figure 66 : Taux d'erreur

Section IV - Discussion

Ce clavier présente l'avantage d'attirer l'attention de l'utilisateur sur la liste. Cette attention plus grande permet ainsi à l'utilisateur de voir plus fréquemment les mots ou les chaînes de caractères de la liste qui peuvent l'aider dans sa saisie. Ainsi, il utilise plus souvent la liste, ce qui a pour avantage de diminuer le nombre d'actions à réaliser. Ceci, couplé au fait que les distances à parcourir avec le pointeur sont moins longues, entraîne une augmentation de la vitesse de saisie de texte.

Ces deux derniers facteurs sont notamment très importants pour les personnes ayant une faible motricité des membres supérieurs car la diminution de la distance à parcourir leur permet de diminuer la fatigue motrice. D'autre part, une augmentation de la vitesse de saisie est toujours bienvenue pour ces personnes dont la vitesse de saisie est généralement largement inférieure à celle que nous pouvons avoir sur un clavier physique.

Conclusion

La recherche sur la saisie de texte et l'accessibilité à la communication écrite est un domaine assez récent. Initialement (dans les années 80), ces recherches concernaient en grande partie les personnes souffrant d'un handicap moteur et la possibilité de leur donner accès à l'outil informatique. Par la suite, ces recherches se sont considérablement développées avec l'émergence du Web et des dispositifs mobiles. Le but est de palier l'impossibilité d'utiliser un clavier physique tout en gardant des performances de saisie qui soient comparables.

Malgré la disparition des claviers physiques, la saisie de texte reste néanmoins possible grâce à des moyens logiciels (claviers logiciels, système de reconnaissance d'écriture) avec lesquels les utilisateurs interagissent à l'aide de différents dispositifs : stylet, doigt, souris ou autres dispositifs de pointage. Ainsi, ces différentes méthodes d'interaction, ajoutées aux profils des utilisateurs rendent difficile la conception d'un système généraliste, utile et utilisable par tous.

Cependant, quelque soient les dispositifs d'interaction et la motricité des utilisateurs, les performances d'un clavier logiciel restent inférieurs à celles d'un clavier physique, sur lequel l'interaction peut se faire à l'aide de 10 doigts et non un seul pointeur. Des recherches se sont accrues pour remédier alors à ce défaut. Des systèmes de prédiction sont couplés aux claviers dans le but de faciliter la tâche de saisie à l'utilisateur. Malgré l'intégration de ces techniques, le taux de texte entré reste faible et le temps de saisie reste long. De plus, lors d'une saisie de texte long avec ces aides, les utilisateurs ressentent une fatigue motrice et visuelle.

C'est pourquoi nous avons choisi de focaliser ces travaux de thèse sur l'étude de ces systèmes et plus particulièrement sur les listes de prédiction ajoutées en complément du clavier logiciel. Nos travaux ont consisté dans un premier temps à étudier plus finement l'utilisation d'une liste de prédiction avec un clavier logiciel. Pour cela, nous avons mené une expérimentation comparant l'utilisation de ce dispositif avec l'utilisation d'un clavier logiciel

classique. Pour affiner notre analyse, nous avons appareillé cette expérimentation d'un système de suivi du regard, ce qui nous a permis d'avoir un complément d'information sur la manière dont les personnes utilisent la liste durant leur saisie. Les principaux résultats montrent que le simple fait d'avoir la liste de prédiction à côté du clavier coûte 80 millisecondes à l'utilisateur par caractères saisis sur le clavier logiciel. Ceci nous a aussi permis d'apprendre que l'utilisateur préférait attendre un peu plus longtemps en saisissant un caractère ou deux de plus sur le clavier pour avoir ensuite le mot dans les 3 premiers items. Ce constat a été confirmé par la position du regard sur la liste. Ainsi, nous avons pu constater que dans près de 80% des cas, l'utilisateur ne regarde que le haut de la liste. Enfin, ce suivi du regard nous a permis aussi de voir que l'utilisateur ne regarde en direction de la liste que lors de la saisie d'un caractère sur trois.

A partir de l'analyse des résultats de cette expérimentation, nous avons pu proposer un modèle de prédiction du temps moyen de saisie d'un caractère prenant en compte les différentes alternatives proposées à l'utilisateur pour saisir un mot. Pour que ce modèle soit plus facilement utilisable, nous avons implémenté un outil permettant de prototyper un clavier logiciel avec une liste de prédiction et qui permet de prédire le temps moyen de saisie à l'aide de notre modèle.

Enfin, nous avons proposé deux systèmes qui répondent chacun à un constat de notre étude : le premier consiste à essayer de maximiser le nombre d'utilisation de la liste de prédiction. Pour cela, nous proposons une interaction qui permet de sélectionner n'importe quelle chaine de caractère de la liste. Ainsi l'utilisateur peut utiliser plus régulièrement la liste, ce qui lui permet de réduire le nombre d'actions à effectuer et par conséquent augmente sa vitesse de saisie. Le second clavier proposé consiste à placer la liste au centre de l'attention de l'utilisateur. Ainsi, l'utilisateur regarde plus souvent la liste et utilise plus rapidement les résultats qui y sont proposés.

Les deux solutions que nous avons proposées ont permis d'augmenter la vitesse de saisie par rapport à une liste de prédiction classique. Cependant, nous n'avons pas évalué ces propositions par rapport à un clavier logiciel seul, et notre clavier ne comportait que les lettres de l'alphabet latin. Donc les résultats de nos propositions restent à confirmer par un clavier plus complet et sur une utilisation quotidienne.

Perspectives

Les travaux de cette thèse peuvent être continués. Nos perspectives pour ces travaux se positionnent sur plusieurs points :

En premier lieu, nous avons proposé un modèle théorique pour prédire les performances de saisi sur un clavier logiciel couplé à une liste de prédiction. Nous avons vu que l'utilsateur peut avoir plusieurs alternatives pour saisir un mot sur le clavier avec la liste. Ces alternatives n'étant pas équiprobables, une des perspectives est d'améliorer ce modèle en pondérant d'une manière efficace ces alternatives. D'autre part, nous pourrons améliorer ce modèle en modélisant le temps de sélection sur la liste, que nous avons considéré comme fixe. Nous pourrons s'inspirer du modèle élaboré dans Sad [Sad 2009a], soit en trouvant une nouvelle valeur de ce temps en analysant toutes les variables que nous n'avons pas considérer (indice du mot dans la liste, ordre des mots dans la liste, etc.)

En second lieu, les résultats des évaluations des solutions présentées montrent que ce type des solutions peut être bénéfique pour les utilisateurs valides ainsi que pour les utilisateurs en situation d'handicap moteur des membres supérieurs. Cependant ces claviers sont réduits aux 26 caractères de l'alaphabet latin. Or dans une utilsation quotidienne, ces claviers ne peuvent pas être utilisés. Notre deuxième perspective se dessine alors autour d'une répétition de nos évaluations à l'aide de clavier complet contenant les chiffres, les accents, la ponctuation, etc. Même si ces caractères secondaires sont moins utilisés que ceux de l'alphabet, il ne faut pas que ceux-ci viennent détériorer complètement les effets bénéfiques montrés. Cela peut être facilement exécuté sur notre premier paradigme (WordTree) puisque la disposition des touches dans le clavier est la même que AZERTY ou QWERTY. Cependant pour CentraList, des solutions doivent être présentées. Une solution envisageable, serait de proposer une troisième « couronne » qui viendrait s'ajouter autour des barres espaces. Néanmoins cette solution a aussi ses inconvénients car la liste permet généralement de saisir la fin d'un mot. Les caractères de ponctuation sont généralement utilisés à la fin des mots. Donc s'ils se trouvent trop loin de la liste, cela risque d'augmenter

fortement les distances à parcourir et donc par conséquent réduire la vitesse de saisie. L'idée de mettre la liste au centre parait donc être une bonne idée. Il faut maintenant voir comment placer au mieux l'ensemble des caractères autour. Une solution pour résoudre ce problème complexe pourrait être l'utilisation d'algorithme d'optimisation de disposition de caractères tels que GAG [Raynal 2005b] ou Metropolis [Zhai 2000]. Une autre solution consiste à adopter la disposition optimisée « thersan » utilisée dans un clavier edge multipage [Wobbrock 2003].

Enfin, pour mieux affirmer nos premiers résultats, nous pourrons comparer nos propositions avec des claviers sans prédiction pour confirmer nos hypothèses de solution. Une évaluation de nos propositions peut se faire avec des disposifs de petite taille (dispositif mobile) ainsi qu'avec des écrans tactiles et des dispositifs multi-touch. L'utilisation du système de suivi oculaire avec nos paradigmes peut encore être le sujet d'un prochain travail pour affirmer nos résultats.

Bibliographie

[AL Faraj 2009] K. AL FARAJ , M. MOJAHID, N. VIGOUROUX, BigKey : A Virtual Keyboard for Mobile Devices, In Proceedings of HCII 2009, Springer-Verlag, LNCS, pp. 3-10, 2009

[Allen 1987] J. ALLEN, Natural Language Understanding, *Benjamin Cummings Publishing Company*. 1987.

[Anson 1993] D. ANSON, The effect of word prediction on typing speed, *The American Journal of Occupational Therapy*, 47 (11), 1039-1042, 1993.

[Antoine 2007] J.Y. ANTOINE, D. MAUREL, Aide à la communication pour personnes handicapées et prédiction de texte (Problématique, état des lieux et retour sur trente ans de recherche en communication augmentée), *article de revue TAL* vol 48, pages 3-40, 2007.

[Arnott 1992] J.L. ARNOTT, M.Y. JAVED, Probabilistic character disambiguation for reduced keyboard using small text samples. *Dans Actes de Augmentative and alternative communication*, vol. 8, pages 215-223, 1992.

[Aulagner 2010] G. AULAGNER, R. FRANCOIS., B. MARTIN, D. MICHEL, et M. RAYNAL, FloodKey: Increasing Software Keyboard Keys by Reducing Needless Ones without Occultation, *Proceedings of 10th WSEAS International Conference on APPLIED COMPUTER SCIENCE. (ACS '10)*, Iwate (Japon), 2010.

[Badr 2009] G. BADR, M. RAYNAL, WordTree: Results of a Word Prediction System Presented Thanks to a Tree. *In Proceedings of the 13th International Conference on Human-Computer Interaction (HCII09),* 19-24 July 2009.

[Badr 2011] G. BADR, M. RAYNAL, CentraList, *In Proceedings of the 11th European Conference for the Advancement of Assistive Technology, (AAATE 2011),* 31 Août-2 Septembre 2011.

[Beck 2004] C. BECK, G. SEISENBACHER, G. EDELMAYER, et W.L ZAGLER, First User Test Results with the Predictive Typing System FASTY. *In Proceedings of 9th Computer Helping People with Special Needs*, pages 813-819, 2004.

[Bellman 1998] T. BELLMAN, I.S. MACKENZIE, A probabilistic character layout strategy for mobile text entry. *Dans Actes de Graphics Interface*, pages 168-176, 1998

[Bérard 2004a] Ch. BERARD, *Clavier-écran: Concevoir avec les utilisateurs* In Handicap 2004, pp. 83-88, 2004.

[Bérard 2004b] Ch. BERARD and D. NIEMERIJER. *Evaluating effort reduction through different word processing systems.* In IEEE SMC 2004: International Conference on Systems, Man and Cybernetics, pages 2658-2663, 2004.

[Biard 2006] N. BIARD, C. DUMAS, J. BOUTEILLE, D. POZZI, F. LOFASO, I. LAFFONT, Apports de l'évaluation en situation de vie à partir d'une étude sur l'intérêt de la prédiction de mots auprès d'utilisateurs de synthèse vocale, *Dans Actes Handicap 2006*, Paris, pages 145-148. 2006.

[Bier 2009] J. BIER, I. PECCI, WeGliss : clavier virtuel réduit pour Wiimote, *Dans 21ème Conférence sur l'Interaction Homme-Machine (IHM 09)*, ACM, Boston, USA, 4-9 Avril 2009.

[Boissière 1996] P. BOISSIERE, D. DOURS, VITIPI: Versatile Interpretation of Text Input by Persons with Impairments, *5ème Actes de ICCHP*, pages 165-172. Linz, July 1996.

[Boissière 2000] P. BOISSIERE, VITIPI : Un système d'aide à l'écriture basé sur un principe d'auto apprentissage et adapté à tous les handicaps

moteurs, *Handicap 2000*, pages 81-86, Paris, 2000.

[Buisson 2002] M. BUISSON, A. BUSTICO, S. CHATTY, F. COLIN, Y. JESTIN, S. MAURY, C.P. MERTZ, and P. TRUILLET, Ivy: un bus logiciel au service du développement de prototypes de systèmes interactifs, In Proceedings of *IHM 2002*, pp.223-226, 2002.

[Cantegrit 2001] B. CANTEGRIT, J.M. TOULOTTE, Réflexions sur l'aide à la communication des personnes présentant un handicap moteur, *Actes atelier Ingénierie de la langue et handicap*, TALN'2001, pages 193-202, Tours, France, 2001.

[Card 1978] S.K. CARD, W. ENGLISH and B.J. BURR; Evaluation of mouse, rate controlled isometric joystick, step keys and test keys for text selection on an CRT, Ergonomics, 21: 601-613, 1978.

[Card 1980] S.K. CARD, T.P. MORAN, and A. NEWELL, The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM*, pages 396-410, July, 1980.

[Card 1983] S.K. CARD, T.P. MORAN and A. NEWELL, The psychology of human-computer interaction, *Hillsdale, NJ: Lawrence Erlbaum Associates*, Inc, 1983.

[Card 1986] S.K CARD, T.P. MORAN, and A. NEWELL, The Model Human Processor: An Engineering Model of Human Performance, *In K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), Handbook of Perception and Human Performance. Vol. 2: Cognitive Processes and Performance*, 1986, pages 1–35.

[Carlberger 1997] J. CARLBERGER, Design and Implementation of a Probabilistic Word Prediction Program, *Master's Thesis Dept. of Speech*, Music and Hearing, KTH, SE-100 44 Stockholm, Sweden, 1997.

[Chubon 1988] R.A. CHUBON and M.R. HESTER, An enhanced standard computer keyboard system for single-finger and typing-stick typing,

Rehabilitation Research and Development, 25 (4): 17-24, 1988.

[Clawson 2008] J.CLAWSON, K. LYONS, A. RUDNICK, R.A.J. IANNUCCI, T. STARNER, Automatic whiteout++: correcting mini-QWERTY typing errors using keypress timing, *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing system,* pages 573-582, Florence, Italy, 2008.

[Cockburn 2007] A. COCKBURN, C. GUTWIN, S. GREENBERG, A predictive model of menu performance, *Proc. CHI*, pp. 627-636, 2007.

[Colad 2008] S. COLAD, N. MONARCHE, P. GAUCHER, M. SLIMANE, Artificial Ants for the Optimization of Virtual Keyboard Arrangement for Disabled People, *In Proceedings of the 8th international conference on Artificial evolution EA'07*, Springer-Verlag, pp. 89-99, 2008.

[Copestake 1997] A. COPESTAKE, Augmented and alternative NLP techniques for augmentative and alternative communication. *In Proceedings of the ACL workshop on Natural Language Processing for Communication Aids*, 1997.

[Copestake 1998] A. COPESTAKE, D. FLICKINGER, Evaluation of NLP technology for AAC using logged data, In: *Isaac 98 research symposium proceedings*, Whurr Publishers, London. http://www-csli.stanford.edu/_aac/evaluation-dsp1.htm, 1998.

[Dabbagh 1985] H.H. DABBAGH, R.I. DAMPER, Average selection length and time as predictors of communication rate, *Proceeding of RESNA 1985 Annual Conference*, Washington. DC: RESNA Press, pp. 404-406, 1985.

[Damper 1984] R.I. DAMPER, Test composition by the physically disabled: A rate prediction model for scanning input, *Applied Ergonomies*, 15, 289-296. 1984.

[Darragh 1999] JJ. DARRAGH, IH. WITTEN, MJ. JAMES, The Reactive Keyboard: A Predictive Typing Aid, *IEEE Computer, 23(11),* pages 41-49,

November 1990.

[Digiovanni 1996] JM. DIGIOVANNI, Word prediction software as a tool to improve writing literacy, http://san183.sang.wmich.edu/Sped603fall96/Digiovannipaper.html, 1996.

[Dominowska 2002] E. DOMINOWSKA, D. ROY, and R. PATEL, An adaptive context-sensitive communication aid. *Dans Actes de the 17th Annual Technology and Persons with Disabilities Conference,* Northridge, CA, 2002.

[Dunlop 2000] M.D. DUNLOP and A. CROSSAN, Predictive Text Entry Methods for Mobile Phones, Personal Technologies, 4(2): 134-143, 2000.

[Dunlop 2009] M. DUNLOP, F. TAYLOR, Tactile Feedback for Predictive Text Entry, *In Proceedings of CHI 2009*, Boston, USA, 4-9 Avril 2009.

[Fazly 2003] FAZLY, G. HIRST, Testing the efficacy of part-of-speech information in word completion, *in Proceedings of Workshop on Language Modeling for Text Entry Methods, EACL'2003*. Budapest, Hongrie. 2003.

[Felzer 2005] T. FELZER, R. FISCHER, T. GRÖNSFELDER, and R. NORDMANN. Alternative control system for operating a PC using intentional muscle contractions only. *In Online-Proc. CSUN Conf.,* 2005

[Felzer 2006] T. FELZER, R. NORDMANN, Alternative Text Entry Using Different Input Methods, *In Proceedings of The Eighth International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS'06)*, pages 22–25, Portland, Oregon, USA. October, 2006.

[Fitts 1954] P.M. FITTS, The information capacity of the human motor system in controlling the amplitude of movement, *Journal of Experimental Psychology*, vol 47 pages 381-391, 1954.

[Fitts 1964] P.M. FITTS, and J.R. PETERSON, Information capacity of discrete

motor response, *Journal of Experimental Psychology*, vol 67, pages 103-112, 1964.

[Fortune 1987] S. FORTUNE, A Sweepline Algorithm for Voronoi Diagrams, *Algorithmica*, volume 2, 1987, pp. 153-174.

[Foulds 1987] R.A. FOULDS, M. SOEDE, H. Van BALKOM, Statistical disambiguation of multi-character keys applied to reduce motor requirements for augmentative and alternative communication, In *Augmentative and alternative communication*, vol. 3, pages 192-195, 1987

[Furnas 1986] G. W. FURNAS, Generalized fisheye views, *Dans Actes de the SIGCHI conference on Human factors in computing systems*, pages 16-23, Boston, Massachusetts, United States, April 13-17, 1986

[Garay 1994] N. GARAY, J. ABASCAL, Using statistical and syntactic information in word prediction for input speed enhancement. *Information Systems Design and Hypermedia*. Cépaduès-Editions, Toulouse (France), pp 223–230, 1994.

[Garay-Victoria 1997] N. GARAY-VITORIA, J. GONZALEZ-ABASCAL, Intelligent Word-Prediction to Enhance Text Input Rate, (A Syntactic Analysis-Based Word-Prediction Aid for People with Severe Motor and Speech Disability), *IU1 97*, Orlando Florida USA, pages 241-244, 1997.

[Garay-Victoria 2006] N. GARAY-VITORIA et J. ABASCAL, Text prediction systems: a survey, dans *Univ Access Inf Soc 4:* 188–203, 2006.

[Gentner 1983] D.R. GENTNER, J.T. GRUDIN, S. LAROCHELLE, D.A. NORMAN and D.E. RUMELHART, Cognitive aspects of skilled typing, Chapitre A glossary of terms including a classification of typing errors, pages 39-44, New York, Springer-Verlag, 1983.

[Getshow 1986] C.O. GETSCHOW, M.J. ROSEN and C. GOODENOUGH-TREPAGNIER, A systematic approach to design of a minimum distance alphabetical keyboard, *In Proceedings of Rehabiliation Engineering Society of North America – RESNA 9th Annual*

Conference, pages 396-398, 1986.

[Gibler 1982] C.D. GIBIER, D.S. CHILDRESS, Language anticipation with a computer-based scanning aid. *Proc. IEEE Computer Society Workshop on Computing to Aid the Handicapped,* Charlottesville. VA. 11-16, 1982.

[Go 2007] K. GO, Y. ENDO, CATKey : Customizable and Adaptable Touchscreen Keyboard with Bubble Cursor-Like Visual Feedback, In *Proceedings of Interact 2007*, LNCS, pp. 493-496, 2007.

[Godard 2010] N. GODARD, J. BIER, B. MARTIN, M. RAYNAL , SlideKey : un clavier virtuel polyvalent, *Dans IHM'2010, Démonstration*, Luxembourg, LU Septembre 20-23, 2010.

[Gong 2006] J. GONG, P. TARASEWICH, A New Error Metric for Text Entry Method Evaluation, *Dans 18ème Conférence sur l'Interaction Homme-Machine (IHM 06)*, ACM, Montérial, Canada, 22-27 Avril 2006.

[Gong 2008] J. GONG, P. TARASEWICH, and I. S. MACKENZIE, Improved word list ordering for text entry on ambiguous keyboards. *Dans Actes de the Fifth Nordic Conference on Human-Computer Interaction* – (CHI 2008), pages 152-161, New York, 2008.

[Goodenough-Trepagnier 1988] C. GOODENOUG-TREPAGNIER, M.J. ROSEN, Predictive assessment for communication aid prescription: Motor-determined maximum communication rate, *The Vocally Impaired: Clinical Practice and Research*, Philadelphia, Grune and Stratton, pp. 167-185, 1988.

[Grange 2010] A. GRANGE, L'interface du clavier virtuel Chewing Word, *Dans IHM'2010, Démonstration*, Septembre 20-23, 2010, Luxembourg, LU.

[Greenberg 1995] S. GREENBERG, JJ. DARRAGH, D. MAULSBY, IH. WITTEN, Predictive interfaces: what will they think of next? *Extra-ordinary human-computer interaction. Interfaces for users with disabilities, Edwards ADN (Ed) Cambridge University Press*, London, pp 103–

140, 1995.

[Guenthner 1992] F. GUENTHNER, K. KRÜGER-THIELMANN, R. PASERO, P. SABATIER Communications Aids for ALS Patients, Proc. Third International Conference on Computers for Handicapped Persons, ICCHP'92, p. 303-307. 1992.

[Guiard 2004] Y. GUIARD and M. BEAUDOUIN-LAFON. Preface: Fitts' law 50 years later: Applications and contributions from human-computer interaction, *International Journal of Human-Computer Studies*, 61(6): 747-750, 2004.

[Harbusch 2003] K. HARBUSCH, S. HASAN, H. HOFFMANN, M. KUHN, B. SHULER, Domain specific Disambiguation for Typing with Ambiguous Keyboards. *In EACL Workshop on Language Modeling for Text Entry Methods*, Budapest, April 14, 2003

[Heckathorne 1983] C.W. HECKATHORNE, D.S. CHILDRESS, Applying anticipatory text selection in a writing aid for people with severe motor impairment, *IEEE Micro 3*(3):17–23, 1983.

[Heckthorne 1983] CW. Heckathorne, DS. Childress, Applying anticipatory text selection in a writing aid for people with severe motorimpairment. *IEEE Micro* 3(3):17–23, 1983.

[Heinisch 1993] B. HEINISCH, J. HECHT, Predictive word processors: a comparison of six programs. *Tam News 8*:4–9, 1993.

[Hick 1952] W.E. HICK, On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, vol 4, pages 11-36, 1952.

[Higginbotham 1992] D. HIGGINBOTHAM, Evaluation of keystroke savings across five assistive communication technologies. *Augmentative Alternative Commun 8*, pp.258–272, 1992.

[Holleis 2007] P. HOLLEIS, F. OTTO, H. HUSSMANN and A. SCHMIDT, Keystroke-level model for advanced mobile phone interaction. *In proceedings of: the SIGCHI conference on Human factors in*

computing systems, San Jose, California, USA, ACM Press. 1505-1514, 2007.

[Hornof 1997] A. J. HORNOF and D. E. KIERAS, Cognitive modeling reveals menu search in both random and systematic, *In proceedings of: the SIGCHI conference on Human factors in computing systems*, Atlanta, Georgia, United States, ACM Press. 107-114, 1997.

[Horstmann 1990] H.M. HORSTMANN, S.P. LEVINE, Modeling of user performance with computer access and augmentative communication systems for handicapped people. *AAC,* 6(4): 231-241, 1990.

[Horstmann 1991] H.M. HORSTMANN, S.P. LEVINE, The effectiveness of word prediction, *Proceeding* of *14th Annual RESNA Conference* Washington, D.C.: RESNA: 100-102, 1991.

[Horstmann 1992] H.M. HORSTMANN, S.P. LEVINE, Learning and performance of in scanning systems with and without word prediction - report on a pilot study, *Proc of 15th Annual RESNA Conf,* Washington, D.C., RESNA, 299-301, 1992.

[Hughes 2002] D. HUGHES, J. WARREN and O. BUYUKKOKTEN, Empirical Bi-Action Tables: A Tool for the Evaluation and Optimization of Text-Input Systems. Application I: Stylus Keyboards, *Human-Computer Interaction 17(Special Issue on Text Entry for Mobile Computing)*, pp. 131-169, 2002.

[Hunnicutt 1987] S. HUNNICUTT, Input and output alternatives in word prediction, *STL/QPRS 2–3/,* pp 15–29, 1987.

[Hunnicutt 2001] S. HUNNICUTT, J. CARLBERGER, Improving Word Prediction Using Markov Models and Heuristic Methods, *Augmentative and Alternative Communication*, 17, pp. 255-264, 2001.

[Hyman 1953] R. HYMAN, Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology*, vol 45, pages 188-196, 1953.

[ISO 9241-9] 9241-9:2000(E), R. N. I. Ergonomic requirements for office work with

visual display terminals (vdts)- part 9 - requirements for non-keyboard input devices, *International Organisation for Standardisation*,15:381–391, 2002.

[Jones 1998] P.E. JONES, Virtual keyboard with scanning and augmented by prediction, *In Proceeding 2nd European Conference. Disability, Virtual Reality & Assoc. Tech.*, Skövde, Sweden, pages 45-51, 1998.

[Kieras 1993] D. E. KIERAS, Using the Keystroke-Level Model to Estimate Execution Times, *Unpublished Report*, University of Michigan, 1993.

[Kieras 1995] D.E. KIERAS, S.D. WOOD, D.E. MEYER, Predictive engineering models using the EPIC architecture for a high-performance task. *In Proceedings of the SIGCHI conference on Human factors in computing systems,* ACM Press/Addison-Wesley Publishing Co., Denver, 1995.

[Kieras 1997] D.E. KIERAS, et D.E. MEYER, An overview of the EPIC architecture for cognition and performance with application to human-computer interaction, *Human-Computer Interaction*, 12, 391-438, 1997.

[Kieras 2005] D.E. KIERAS, Fidelity Issues In Cognitive Architectures For HCI Modelling: Be Careful What You Wish For. *In proceedings of: 11th International Conference On Human Computer Interaction (HCII 2005)*, Las Vegas, July. 2005

[King 1995] M.T. KING, C.A. KUSHLER, D.A. GROVER, JustType – Efficient Communication with Eight Keys. *In RESNA'95*, pages 94-96, 1995.

[Klund 2001] J. KLUND, M. NOVAK, If word prediction can help, which program do you choose? de http://www.tracecenter.org/docs/wordprediction2001/, 2001.

[Koester 1993] H. KOESTER, S.P. LEVINE, A model of performance cost versus benefit for augmentative communication systems, *In Proceedings of the 15th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 1303–1304, 1993.

[Koester 1994a] K.H. KOESTER, S.P. LEVINE, Learning and performance of able-bodied individuals using scanning systems with and without word prediction, Submitted for publication in *Assistive Technology 1994.*

[Koester 1994b] HH. Koester HH, SP. Levine, Modelling the speed of text entry with a word prediction interface. *IEEE Trans Rehabil Eng* 2(3):177–187, 1994.

[Koester 1996] HH. KOESTER, SP. LEVINE, Effect of a word prediction feature on user performance, *Augmentative Alternative Communication 12*:155–168, 1996.

[Koester 1998] HH. KOESTER, SP. LEVINE, Model simulations of user performance with word prediction, *Augmentative Alternative Communication 14*:25–35, 1998.

[Land 1981] R.I. LAND, Keyboard Entry – can it be simplified? *In Proceedings of the joint conference on Easier and more productive use of computer systems. (Part II): Human interface and the user interface*, ACM Press, New York, USA, pages 53-58, 1981.

[Landauer 1985] T. LANDAUER, D. NACHBAR, Selection from alphabetic and numeric menu trees using a touch screen: Breadth, depth, and width, In *Proc. CHI '85 Conference: Human Fators in Computer Systems*, pages 73-78. ACM Press, 1985

[Le Pévédic 1997] B. LE PEVEDIC, Prédiction morphosyntaxique évolutive dans un système d'aide à la saisie de textes pour les personnes handicapées physiques. *Rapport de thèse,* Université de Nantes, France, 1997.

[Le Pévédic 1998] B. LE PEVEDIC, Les niveaux syntaxiques dans le système HandiAS. *Actes TALN'1998*, pages 132-141, Paris, France, 1998.

[Lesher 1998] G. LESHER, B. MOULTON, J. HIGGINBOTHAM, Optimal Character Arrangements for Ambiguous Keyboards*, IEEE Transactions on Rehabilitation Engineering*, vol 6, pages 415-423, 1998

[Lesher 1999] G. LESHER, B. MOULTON, J. HIGGINBOTHAM, J. Effects of ngram order and training text size on word prediction. *In Proceedings of Rehabilitation Engineering Society of North America – RESNA'00 Annual Conference*, 1999.

[Lesher 2000] G. LESHER, B. MOULTON, A method for optimizing single-finger keyboards. *Dans Actes de Rehabilitation Engineering Society of North America – RESNA'99 Annual Conference*, pages 91-93, 2000.

[Levenshtein 1966] V.I. LEVENSHTEIN, Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklady*, 10:707-710, 1966

[Levine 1987] S.H. LEVINE, C. GOODENOUGHT-TREPAGNIER, C.O. Getschow, S.L Minneman, Multi-charater key text entry using computer disambiguation, *Dans Actes de Rehabilitation Engineering Society of North America - RESNA 10th Annual Conference*. pages 177-179, Washington, 1987

[Lewis 1999a] J. R. LEWIS, P. J. KENNEDY, and M. J. LALOMIA, Development of a digram-based typing key layout for single-finger/stylus input. *In Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting HFES'99*, pages 415-419, Santa Monica, CA, 1999.

[Lewis 1999b] J. R. LEWIS, M. J. LALOMIA, and P. J. KENNEDY, Evaluation of typing key layout for stylus input, *In Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting-HFES'99*, pages 420-424, Santa Monica, CA, 1999.

[Li 2005] J. LI and G. HIRST, Semantic knowledge in word completion. *Dans Actes de the ACM SIGACCESS Conference on Computers and accessibility, ASSETS'05*, pages 121-128, New York, 2005.

[Lin 1965] S. LIN, Computer solutions of the traveling salesman problem, *Bell System Technical Journal*, 44:2245-2269, 1965.

[Lin 1973] S. LIN, and B.W. KERNINGHAN, An effective heuristic algorithm for the traveling salesman problem, Operations Research, 21:498-516,

1973.

[Luo 2005] L. LUO and B.E. JOHN, Predicting task execution time on handheld devices using the keystroke-level model. *In proceedings of: CHI '05 extended abstracts on Human factors in computing systems*, Portland, OR, USA, ACM Press. 1605-1608, 2005.

[MacCaul 2004a] B. MCCAUL, A. Sutherland, Predictive Text Entry in Immersive Environments, *IEEE Virtual Reality,* pages 241-242, Chicago, IL USA, 2004.

[MacCaul 2004b] B. MCCAUL, A. Sutherland, Predictive Text Entry in Immersive Environments, *IEEE Virtual Reality,* pages 241-242, Chicago, IL USA, 2004.

[MacKenzie 1991] I.S. MACKENZIE, A. SELLEN and W. BUXTON. A comparison of input devices in element pointing and dragging tasks. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 161-166. ACM press, 1991.

[MacKenzie 1992a] I.S. MACKENZIE, Fitts' law as a research and design tool in human-computer interaction. Human-Computer Interaction, 7:91-139, 1992.

[MacKenzie 1992b] I. S. MACKENZIE and W. BUXTON, Extending Fitts' law to two-dimensional tasks. *In proceedings of: the SIGCHI conference on Human factors in computing systems*, Monterey, California, United States, ACM Press. 219-226, 1992.

[MacKenzie 1999a] I.S. MACKENZIE, X.S. ZHANG, R.W. SOUKOREFF, Text entry using soft keyboards, *Behavior & Information Technology*, 18, pages 235-244, 1999.

[MacKenzie 1999b] I.S. MACKENZIE, X.S. ZHANG, The design and evaluation of a high performance soft keyboard. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, ACM Press, New York, USA, pages 25-31, 1999.

[MacKenzie 2001a] I.S. MACKENZIE, H. KOBER, D. SMITH, T. JONES, E. SKEPNER,

LetterWise: prefix-based disambiguation for mobile text input, *In Proceedings of the 14th annual ACM symposium on User interface software and technology*, Orlando, Florida, 2001.

[Mackenzie 2001b] I. S. MACKENZIE and S. X.ZHANG, An empirical investigation of the novice experience with soft keyboards, *Behaviour & Information Technology 20(6)*: 411 – 418, 2001.

[MacKenzie 2002a] I.S. MACKENZIE and R.W. SOUKOREFF, A character-level error analysis technique for evaluation text entry methods. *In NordiCHI'02: Proceedings of the second Nordic conference on Human-Computer Interaction*, pages 243-246, ACM Press, 2002.

[MacKenzie 2002b] I.S. MACKENZIE, KSPC (keystrokes per character) as a characteristic of text entry techniques. *Proceedings of the Fourth International Symposium on Human-Computer Interaction with Mobile Devices*. Springer- Verlag pages 195-210, 2002.

[MacKenzie 2002c] I.S. MACKENZIE and R.W SOUKOREFF, Text entry for mobile computing: Models and methods, theory and practice. *Human-Computer Interaction*, 17:147-198, 2002.

[MacKenzie 2002d] I.S. MACKENZIE and R.W. SOUKOREFF, A model of two-thumb text entry. *In proceedings of: Graphics Interface 2002*, Toronto, Canada, Canadian Information Processing Society. 117-124, 2002.

[MacKenzie 2003] I.S. MACKENZIE, R.W. SOUKOREFF, Phrase sets for evaluating text entry techniques. *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems – CHI 2003*, pp. 754-755 New York: ACM, 2003.

[MacKenzie 2006] I.S. MACKENZIE, J. CHEN, A. ONISZCZAK, Unipad: Single Stroke Text Entry With Language-based Acceleration, *NordiCHI'06*, pages 14-18, Oslo, Norway, 2006.

[MacKenzie] I.S. MACKENZIE, Analysis of Variance explained and a tool to do it (API) http://www.yorku.ca/mack/RN-Anova.html

[Magnien 2003] L. MAGNIEN, J.L. BARAOUI, F. VELLA, Utilisation d'indices visuels pour l'aide à la saisie de texte sur PDA, *IHM 2003*, Caen, 25-28 Novembre 2003, pp. 252 à 255.

[Magnien 2004] J.L. MAGNIEN, L. BOURAOUI and N. VIGOUROUX, Mobile devices : soft keyboard text-entry enhanced by Visual Cues. *In proceeding of the 1st French-speaking conference on Mobility and ubiquity computing*, pages 158-165, New York, NY, USA, 2004. ACM Press.

[Marcou] G. MARCOU, C. LITZINGER, Programme développé en collaboration avec Handicap International - *Centre Icom'* – (Mode d'emploi).

[Masui 1999] T. MASUI, POBox: An Efficient Text Input Method for Handheld and Ubiquitous Computers. *Dans Actes de the International Symposium on Handheld and Ubiquitous Computing (HUC99)*, pages 289-300, September, 1999.

[Matias 1993] E. MATIAS, I.S MACKENZIE and W. BUXTON, Half-QWERTY: a one-handed keyboard facilitating skill transfer from QWERTY. *In Proceedings of the CHI 93*, pages 88-94. ACM, 1993.

[Matias 1996] E. MATIAS, I.S MACKENZIE and W. BUXTON, One-Handed Touch-Typing on a QWERTY Keyboard. *Human-Computer Interaction, 11:1-27*, 1996.

[Maurel 2001] D. MAUREL, N. ROSSI, R. THIBAULT, Handias : un système multilingue pour l'aide à la communication de personnes handicapées, *TALN'2001, vol. 2 (tutoriels et conférences associées)*, Tours, France, pp. 203-212, 2001.

[Ménier 2001] G. MENIER, F. POIRIER, Système adaptatif de prédiction de texte, *TALN 2001*, Tours, 2-5 juillet 2001.

[Merlin 2009] B. MERLIN, M. RAYNAL, SpreadKey: increasing software keyboard key by recycling needless ones. Dans : *European Conference for*

the Advancement of Assistive Technology in Europ (AAATE 2009), *Florence, Italie*, *31/08/2009-02/09/2009*, IOS Press, long paper, p. 138-143, septembre 2009.

[Merlin 2011] B. MERLIN, Méthodologie et instrumentalisation pour la conception et l'évaluation des claviers logiciels, Thèse de Doctorat, Université Paul Sabatier, 2011.

[Metropolis 1953] N. METROPOLIS, A. ROSENBLUTH, M. RPSENBLUTH, A. TELLER and E. TELLER, Equation of state calculations by fast computing machines, *Journal of Chemical Physics*, 21:1087-1090, 1953.

[Newell 1991] A.F. NEWELL, L. BOOTH, W. BEATTIE, Predictive text entry with PAL and children with learning difficulties. *Britich Journal of Educaction Technology,* 22(1), 23-40, 1991.

[Newell 1992] AF. NEWELL, JL. ARNOTT, L. BOOTH, W. BEATTIE, B. BROPHY, IW. RICKETTS, Effect of the ‘‘Pal’’ word prediction system on the quality and quantity of text generation, *Augmentative Alternative Communication 8*:304–311, 1992.

[Niemeijer 2005] D. NIEMEIJER, In memoriam of Christian Bérard: Striving for effort reduction through on-screen keyboard word prediction, *Assistive technology: from virtuality to reality - 8th European conference for the advancement of assistive technology in europe (AAATE 2005),* A Pruski, H Knops (Eds.), IOS Press, http://www.irit.fr/aaate2005wk/, Lille, France, 6-9 Septembre 2005.

[Norman 1982] D.A. NORMAN and D. FISHER, Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human factor*, 24: 509-519, 1982.

[Pavlovych 2005] A. PAVLOVYCH and W. STUERZLINGER. An analysis of Novice Text Entry Performance on Large Interactive Wall Surfaces. In *HCI International 2005*, pages CD-Rom. Lawrence Erlbaum, 2005.

[Poirier 2006] F. POIRIER, M. BELATAR, Glyph 2: un saisie de texte avec deux

appuis de touche par caractère - principes et comparaisons, *Dans Actes de the 18th international conference on Association Francophone d'Interaction Homme-Machine*, pages 159-162, Montreal, Canada, April 18-21, 2006.

[Potipiti 2001] T. POTIPITI, V. SORNLERTLAMVANICH, K. THANADKRAN, Towards an Intelligent Multilingual Keyboard System, *In Proceedings of the First International Conference on Human Language Technology Research*, pages 1070-1073, 2001.

[Pouech 2008] N. POUECH, Etude de présentation de listes de mots présentées par un système de prédiction, *Rapport de stage de 1ère et 2ème année d'école d'ingénieur spécialité cognitique*, avril-juillet 2008.

[Rabiner 1986] J. RABINER, An Introduction to Hidden Markov Models, *IEEE ASSP MAGAZINE*, vol 3, pages 4-16, 1986.

[Raynal 2004] M. RAYNAL, KeyGlasses : Des touches semitransparentes pour optimiser la saisie de texte, *Rencontre Jeunes Chercheurs en IHM*, pages 93-96, Lacanau, 2004.

[Raynal 2005a] M. RAYNAL, Systèmes de saisie de textes pour les personnes handicapées moteur: optimisation, interaction et mesure de l'utilisabilité, *thèse de doctorat*, IRIT, 2005.

[Raynal 2005b] M. RAYNAL, N. VIGOUROUX, Genetic Algorithm to Generate Optimized Soft Keyboard. *In Proceedings of the 1st International Conference for Human-Computer Interaction (CHI 2005)*, Portland, Oregon USA, pages CDRom, 2005.

[Raynal 2007] M. RAYNAL, M. SERRURIER, Loi de Fitts: Prédiction du temps de pointage sous forme d'ensemble flous, *Dans 19ème Conférence sur l'Interaction Homme-Machine (IHM 07),* Paris, France, 13-15 Novembre 2007.

[Sad 2008] H. SAD, F. POIRIER, *Evaluation and optimization of word disambiguation for text entry methods*. Dans International

[Sad 2009a] H. SAD, F. POIRIER, Modeling Word Selection in Predictive Text Entry, Human-Computer Interaction, Part II, HCII 2009, LNCS 5611, pp. 725–734, 2009.

[Sad 2009b] H. SAD, Interface de saisie de texte en mobilité : modélisation, conception et évaluation, *thèse de doctorat* soutenue à l'Université Européenne de Bretagne, le 02 octobre 2009.

[Schadle 2001] I. SCHADLE, B. Le PEVEDIC, J-Y. Antoine, F. Poirier, SibyLettre, Système de prédiction de lettre pour l'aide à la saisie de texte, *Actes TALN'2001, atelier thématique "Handicap et Ingénierie Linguistique"*, vol 2, pages 233-242, Tours, France, 2001.

[Schadle 2004] I. SCHADLE, J-Y. ANTOINE, B. Le PEVEDIC, F. Poirier, SibyMot: Modélisation stochastique du langage intégrant la notion de chunks, *Actes TALN'2004*, Session Poster, pages 19-21, Fès, avril 2004.

[Scull 1988] J. SCULL, L. HILL, A computerized communication message preparation program that learns the user vocabulary, *Augment Alternative Commun,* 4 (1): 40-42, 1988.

[Sears 2001] A. SEARS, J. JACKO, J. CHU, F. MORO, The role of visual search in the design of effective soft keyboards. Behavior and Info. Tech., 20, pp. 159-156, 2001.

[Shannon 1949] C.E. SHANNON and W. WEAVER, The mathematical theory of communication. University of Illinois Press, 1949.

[Shieber 2003] M. SHIEBER, E. BAKER, Abbreviated Text Input, *IUI'03*, pages 293-296, Miami, Florida, USA, 2003.

[Soede 1986] M. SOEDE, R.A. FOULDS, Dilemma of prediction in communication aids and mental load, *Proceeding of RESNA 9th Annual Conference*, Washington. DC., RESNA Press, pp. 357-359, 1986.

[Soukoreff 1995] R.W., SOUKOREFF and I.S, MACKENZIE, Theoretical Upper and Lower Bounds on Typing Speed Using a Stylus and Soft Keyboard, *In Behavior and Information Technology*, vol 14, pages 370-379, 1995.

[Soukoreff 2001] R.W. SOUKOREFF and I.S. MACKENZIE, Measuring errors in text entry tasks: an application of the Levenshtein string distance statistic, *In CHI'01: CHI'01 extended abstracts on Human factors in computing systems*, pages 319-320. ACM Press, 2001.

[Soukoreff 2003] R.W. SOUKOREFF and I.S. MACKENZIE, Metrics for text entry research: an evaluation of MSD and KSPC, and a new unified error metric. *In proceedings of: the SIGCHI conference on Human factors in computing systems, Ft. Lauderdale*, Florida, USA, ACM Press, pp113-120. 2003.

[SPB] http://www.spbsoftwarehouse.com/products/keyboard/?en

[Swiffin 1987] AL. SWIFFIN, JL. ARNOTT, JA. PICKERING, AF. NEWELL, Adaptive and predictive techniques in communication prosthesis, *Augmentative Alternative Communication 3*:181–191, 1987.

[T9] Reduced keyboard disambiguating system, Patent -US6307549, 1995.

[Treviranus 1987] J. TREVIRANUS, L. NORRIS., Predictive programs: Writing tools for severely physically disabled students, *Proceedings of RESNA 10th Annual Conference,* Washington. DC: RESNA Press, pp. 130- 132, 1987.

[Trnka 2006] K. TRNKA, D. YARRINGTON, K. MCCOY, C. PENNINGTON, Topic Modeling in Fringe Word Prediction for AAC, *IUI'06,* pages 276-278, Sydney, Australia, 2006.

[Tzimas] http://www.tzimasdimitrios.gr/onscreen_en.html

[Vanderheiden 1987] G.C. VANDERHEIDEN, D.P. KELSO, Comparative analysis OC

fixed-vocabulary communication acceleration techniques, *Augmentative and Alternative Communication, 3*. 196-206, 1987.

[Vella 2005] F. VELLA, N.VIGOUROUX and Ph. TRUILLET. *SOKEYTO : a design and simulation environment of software keyboards.* In A PRUSKI and H KNOPS, editors, Assistive Technology: from virtuality to reality - 8th European conference for the Advancement of Assistive Technology in Europe (AAATE 2005), Lille, France, pages 723-727, ISBN 1-58603-543-6. 6-9 Septembre 2005, IOS Press.

[Venkatagiri 1993] H.S. VENKATAGIRI, Efficiency of lexical prediction as a communication acceleration technique, *Augmentative Alternative Commun 9*:161–167, 1993.

[Venkatagiri 1999] H.S. VENKATAGIRI, Efficient keyboard layouts for sequential access in augmentative and alternative communication, *Augmentative Alternative Commun 15*, 126–134, 1999.

[Vigouroux 2004] N. VIGOUROUX, F. VELLA, Ph. TRUILLRT and M. RAYNAL. Evaluation of AAC for Text Input by two Groups of Subjects: Able-bodies Subjects and Disabled Motor Subjects. 2004.

[Wandmacher 2007] T. WANDMACHER, J-Y. ANTOINE, F. POIRIER, SIBYLLE: A System for Alternative Communication Adapting to the Context and Its User, *In Proceedings of The Ninth International ACM SIGACCESS Conference on Computers & Accessibility,* pages 203-210, Tempe, Arizona, USA, 2007

[Ward 2000] J.D. WARD, A.F. BLACKWELL, D.J.C. MACKAY, Dasher - a data entry interface using continuous gesture and language models. *In Proceedings of the 13th annual ACM symposium on User Interface Software and Technology (UIST 2000)*, pages 129-137, New York, NY, USA, 2000.

[Wester 2003] M. WESTER, User evaluation of a word prediction system, *Master's Thesis*, Department of Linguistics, Uppsala University, 2003.

[Wigdor 2004] D. WIGDOR, R. BALAKRISHNAN, A comparison of consecutive and

concurrent input text entry techniques for mobile phones, *Actes de the SIGCHI conference on Human factors in computing systems,* pages 81-88, Vienna, Austria, 2004.

[Wilson 2006] WILSON, M. AGRAWALA, Text Entry Using a Dual Joystick Game Controller, *Dans 18ème Conférence sur l'Interaction Homme-Machine (IHM 06)*, ACM, Montérial, Canada, 22-27 Avril 2006.

[Witten 1982] I.H. WITTEN, Principles of Computer Speech, *Academic Press*, London, 1982.

[Wobbrock 2003] J.O. WOBBROCK, B.A. MYERS, and S.E. HUDSON, Exploring edge-based input techniques for handheld text entry. *In Proceedings of the 23rd IEEE Conference on Distributed Computing Systems Workshops (ICDCSW '03). Providence, Rhode Island (May 19-22, 2003)*. Los Alamitos, California: IEEE Computer Society, pp. 280-282, 2003

[Wobbrock 2006] J.O. WOBBROCK, B.A. MYERS, From Letters to Words: Efficient Stroke-based Word Completion for Trackball Text Entry, *In Proceedings The Eighth International ACM SIGACCESS Conference on Computers & Accessibility*, pages 333-336, Portland, Oregon, USA, 2006.

[Woltosz 1990] W. WOLTOSZ, A Word about Word Prediction, *Paper presented at CAMA Workshop*, Toronto, ON., 1990.

[Wood 1996] MEJ WOOD, Syntactic pre-processing in single-word prediction for disabled people. *PhD Thesis of the Department Of Computer Science. University of Bristol, 1996.*

[Woodworth 1899] R.S. WOODWORTH, The accurarcy of voluntary movement, *Psychological review,* 3(2): 1-14, 1899.

[Yamada 80] H. YAMADA. *A historical study of typewriters and typing methods: from the position of planning Japanese parallels.* Journal of Information Processing, 2(4):175-202, 1980.

[Zadeh 1965] L. ZADEH, Fuzzy sets, Information Control, 8:338-353, 1965.

[Zhai 2000] S. ZHAI, M. HUNTER, B.A. SMITH, The Metropolis Keyboard - an exploration of quantitative techniques for virtual keyboard design, *Dans Actes de The 13th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 119-218, San Diego, California, 2000.

[Zhai 2002] S. ZHAI, M. HUNTER, B.A. SMITH, Performance Optimization of Virtual Keyboards, *Human - Computer Interaction*, Vol. 17, pages 229-269, 2002.

[Zhai 2003] S. ZHAI, P.O. KRISTENSSON, Shorthand writing on stylus keyboard. *In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '03)*. ACM, New York, NY, USA, 97-104, 2003

Résumé

Les claviers logiciels étaient conçus au départ pour permettre l'accessibilité des postes de travail aux personnes en situation d'handicap moteur. Ils se sont démocratisés pour rendre possible la saisie de textes en mobilité sur des dispositifs dépourvus de claviers physiques tels que les téléphones portables nouvelle génération. Cependant, ces claviers présentent plusieurs inconvénients comme la lenteur de la saisie et la fatigue engendrées pour les utilisateurs déficients moteurs. La solution intuitive était d'allier ces logiciels à des listes contenant les mots susceptibles de continuer la saisie d'un mot initié par l'utilisateur. Bien que ces listes, dites listes de prédiction, réduisent le nombre de clics et le nombre d'opérations, la vitesse de saisie de l'utilisateur a diminué. Cette diminution de la rapidité d'exécution est en partie due au temps nécessaire à l'utilisateur pour prendre connaissance des mots qui constituent la liste de prédiction.

Des modèles théoriques permettent de prédire les performances de tels systèmes de saisie de texte. Cependant, malgré le constat de réduction de la vitesse et l'hypothèse sur le temps de recherche visuelle, les modèles prédictifs actuels estiment que les performances de saisie sont en faveur des listes de prédiction alors que les expérimentations avec utilisateurs montrent l'inverse. Afin de remédier à cette incohérence des modèles théoriques, nous nous sommes intéressés au comportement (notamment oculaire) de l'utilisateur lors de la saisie sur clavier logiciel accompagné d'une liste de prédiction. Une expérimentation outillée d'un système de suivi du regard a ainsi permis de déterminer des « stratégies » de fonctionnement de l'utilisateur face à une liste de mots. Ces résultats ont ainsi permis d'affiner les modèles de prédiction de manière à réduire l'écart séparant les performances prédites des performances réellement enregistrées.

D'autre part, les modèles prédictifs étant fait pour prédire les performances d'un système dans une situation statique, il reste difficile de proposer des mesures de performances pour des systèmes évoluant dynamiquement en fonction de la saisie. Les performances sont

généralement calculées au moyen d'une simulation du fonctionnement du système. Ceci implique que le concepteur du système consacre du temps à implémenter la simulation, ce qui peut s'avérer parfois plus long que de faire une expérimentation avec des utilisateurs. C'est un problème car ces simulations doivent permettre aux concepteurs de détecter plus rapidement les problèmes majeurs de conception. Pour remédier à ce problème, nous proposons un outil de simulation de claviers logiciels. Cet outil permet de prototyper rapidement le clavier logiciel puis d'en simuler l'utilisation pour en obtenir des performances théoriques. Ainsi un concepteur peut aisément proposer un nouveau dispositif et valider son efficacité au moyen de cet outil de simulation avant d'implémenter et valider expérimentalement son système de saisie.

Enfin, à partir des constats effectués lors de la première expérimentation, nous proposons deux variantes de l'utilisation des listes de prédiction de mots. La première propose un nouveau moyen d'interagir avec la liste de mots et permet ainsi de maximiser l'utilisation de celle-ci. La seconde évalue un repositionnement de la liste de mots de manière à réduire le nombre de mouvements oculaires vers la liste. Ces deux évolutions, évaluées théoriquement puis au moyen d'une expérimentation utilisateur, permettent ainsi d'améliorer les performances de saisie par rapport à une liste de prédiction de mots classique.

Abstract

The software keyboard was originally designed to allow accessibility for people with disabilities. They are used to enable text input in mobility and for devices without physical keyboards, such as the new generation of mobile phones. However, these keyboards have several drawbacks such as slowness text entry and fatigue generated for motor impaired users. The solution was to combine software keyboard to lists containing the words likely to continue the word introduced by the user. While these lists, so-called prediction lists, reduce the number of clicks and the number of operations, the speed of user input has decreased. This decrease in speed is partly due to the time required for the user to read the words in the prediction list.

Theoretical models predict the performance of such text input systems. However, despite the a reduction in the speed caused by the visual search time, current predictive models estimate that the performance of text entry are due to the presence of prediction lists, while experiments with users show the opposite. To remedy to this inconsistency of theoretical models, we were interested to the behavior of the user (especially oculaire behavior) when he types on an on-screen keyboard with a prediction list. An experiment with an eye tracking system has identified the "strategies" of the user while using and searching a list of words. These results were helpful to refine the prediction models in order to reduce the gap between the performance predicted and the performance actually recorded.

In addition, predictive models are made to predict the performance of a system in a static situation; it remains difficult to propose measures of performance for dynamic systems that evolve according to the text entered. Performance is typically calculated using a simulation of the system. This implies that the system designer spends time to implement the simulation, which can sometimes be longer that to run a experiment with users. This is a problem because these simulations should enable designers to detect more quickly major problems in design. To solve this problem, we propose a simulation tool for on-screen keyboards. This tool allows us to rapidly prototype the software keyboard and to simulate the use of it to

obtain theoretical performance. Thus a designer can easily propose a new system and validate its effectiveness using this simulation tool before he implements and experimentally validates it.

Finally, based on observations made during the first experiment, we propose two variants of the use of word prediction list. The first proposes a new way to interact with the list of words and allows maximum use of it. The second evaluates a repositioning of the list of words in order to reduce the number of eye movements to the list. These two propositions were theoretically and experimentally evaluated by users. These sofwares can improve the input performances compared with a classic word prediction list.