UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

TECNOLOGIA ASSISTIVA: UM TECLADO VIRTUAL EVOLUTIVO PARA APLICAÇÃO EM SISTEMAS DE COMUNICAÇÃO ALTERNATIVA E AUMENTATIVA

LUIZ FERNANDO BATISTA LOJA

Orientadora: Prof[a] Dr[a] Edna Lúcia Flôres

Uberlândia

2015

LUIZ FERNANDO BATISTA LOJA

TECNOLOGIA ASSISTIVA: UM TECLADO VIRTUAL EVOLUTIVO PARA APLICAÇÃO EM SISTEMAS DE COMUNICAÇÃO ALTERNATIVA E AUMENTATIVA

Tese apresentada ao Programa de Pós-Graduação Strictu Sensu da Faculdade de Engenharia da Universidade Federal de Uberlândia, como recurso parcial para obtenção do título de ***Doutor em Ciência*.**

Área de Concentração: Engenharia Eletrônica
Linha de Pesquisa: Processamento Digital de Sinais

BANCA DE DEFESA:

Edna Lúcia Flôres, Dr[a](UFU) Orientadora

Gilberto Arantes Carrijo, Dr (UFU)

Eduardo Lázaro Martins Naves, Dr (UFU)

Carlos Galvão Pinheiro Junior, Dr (UFG)

Ricardo Antônio Gonçalves Teixeira, Dr (UFG)

Uberlândia

2015

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

L835t 2015

Loja, Luiz Fernando Batista, 1982-

Tecnologia assistiva: um teclado virtual evolutivo para aplicação em sistemas de comunicação alternativa e aumentativa / Luiz Fernando Batista Loja. - 2015.

175 f. : il.

Orientadora: Edna Lúcia Flôres.

Tese (doutorado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Elétrica.

Inclui bibliografia.

1. Engenharia biomédica - Teses. 2. Dispositivos de auto-ajuda para pessoas com deficiência - Teses. 3. Algoritmos genéticos - Teses. I. Flôres, Edna Lúcia, 1958- II. Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU: 62:61

A Deus, a toda espiritualidade e aos meus pais que me fizeram o homem que eu sou hoje.

AGRADECIMENTOS

A Deus, a Jesus, ao meu mentor espiritual e a toda espiritualidade por terem guiado os meus passos, pelos momentos de inspiração e pelo auxílio durante toda essa caminhada.

Ao meu pai, **Luiz Antonio Pinto Loja**, pela força, disponibilidade, otimismo, carinho, preocupação e orgulho. Por saber como me acalmar quando eu estava com pouca ou nenhuma esperança de concluir esta tese. Que eu continue sendo motivo de orgulho para ele e que eu consiga ser pelo menos a metade do homem que ele foi para nossa família.

A minha mãe, **Helena de Oliveira Batista**, que sempre me guiou no caminho do bem e da sabedoria. Por sempre estar disponível para me ensinar e para fazer tudo por mim.

A minha família, **Bianca de Oliveira Batista Loja**, **Luiz Antonio Fonseca Loja**, **Amélia Fernandes** e minha madrinha **Wilma Alves Tolentino** que me apoiaram durante todo esse trabalho com muita paciência e carinho.

A minha orientadora e amiga, **Edna Lúcia Flôres**, pela amizade, confiança, orientação, persistência, paciência e principalmente pela disponibilidade.

Ao meu amigo, **Renato de Sousa Gomide**, que foi peça fundamental para conclusão desta tese, por acreditar nas minhas ideias, pelas inspirações, por nunca ter deixado eu caminhar sozinho, por sempre estar disponível e por gostar e achar bom.

A minha amiga, **Fabiana Freitas Mendes**, pelo carinho, cuidado, disponibilidade, orientação e por ter acreditado em mim e me apoiado em todos os meus projetos, por mais malucos que eles fossem.

Ao meu amigo, **Rodrigo Pinto Lemos**, pela paciência, humildade, inspiração e pelas diversas horas de orientação dispendidas neste trabalho.

Ao meu amigo, **Sirlon Diniz Carvalho**, por me apresentar a Universidade Federal de Uberlândia, por me ensinar a ser melhor como pessoa e por me apoiar e auxiliar durante todo o doutorado.

Ao meu amigo, **Ricardo Antônio Gonçalves Teixeira**, por ter me apresentado a área de conhecimento abordada neste trabalho e pelas orientações.

Ao meu amigo, **Francisco Ramos de Melo**, pelo apoio, cuidado e orientação.

Aos meus amigos, **Leandro dos Santos Pereira** e **Daniel Vitor Lucena**, pois sem nem mesmo saber, foram a inspiração para o ponto de tese apresentado neste trabalho.

A professora **Elise Mendes**, por me acolher em seu projeto de pesquisa facilitando minha adaptação a Universidade Federal de Uberlândia.

Ao meu amigo e aluno, **Robson Barbosa**, por ter me auxiliado na construção estrutural deste documento e pelo apoio.

Ao meu amigo, **Valdemar Vicente Graciano Neto**, pelas orientações e por ter acreditado em mim todo o tempo.

Ao meu amigo, **Márcio Rodrigues Arantes**, pelo apoio e por ouvir sobre os meus problemas constantemente.

A **Kesi Line**, pelo carinho, amor, compaixão e principalmente pelo apoio durante essa longa jornada.

A minha amiga, **Christiane Borges Santos**, pelo apoio, carinho e por estar sempre presente nas horas mais difíceis.

Aos meus amigos, **Sofia Larissa Costa** e **Marcelo Quinta**, pelos artigos que foram essenciais para as publicações dos trabalhos desenvolvidos nesta tese.

Ao meu amigo, **Leandro Theodoro**, pelas disciplinas cursadas em conjunto e pelo apoio acadêmico.

Ao meu amigo, **Fábio Henrique Monteiro Oliveira**, por ter me acolhido no laboratório de processamento de imagens e pelo auxílio nos estudos.

Agradeço todo o corpo docente do Programa de Doutorado em Engenharia Elétrica da Universidade Federal de Uberlândia por me auxiliar na construção do saber.

Ao Instituto Federal de Goiás Campus Luziânia na figura dos professores, **Daniel Rosa Canedo**, **Aldo Lúcio de Freitas Mundim**, **Audir da Costa Oliveira Filho** e **José Carlos Barros Silva**, por terem cuidado dos meus horários de aula e pela motivação.

Aos meus alunos do Instituto Federal de Goiás, pelo apoio, carinho e compreensão.

"Não há fé inabalável senão aquela que pode encarar a razão face a face, em todas as épocas da Humanidade."

(Evangelho Segundo o Espiritismo)

RESUMO

Pessoas com restrições motoras e de fala simultâneas têm a comunicação verbal e a linguagem corporal prejudicadas. Nos casos mais extremos o paciente é privado de todos os seus movimentos e da capacidade de fala. Essa situação é caracterizada como a Síndrome do Encarceramento (SE). As tecnologias de comunicação aumentativa e alternativa são responsáveis por proporcionarem métodos e *softwares* que possibilitam a comunicação dos pacientes com o ambiente e as pessoas que os cercam. Entre os vários métodos e programas de comunicação existentes pode-se destacar o teclado virtual. Porém, a entrada de dados utilizando teclados virtuais é consideravelmente mais lenta e cansativa para pessoas com SE. O objetivo deste trabalho é construir um teclado virtual assistivo para auxiliar pacientes com restrições motoras graves e de fala a se comunicarem. Para atingir esse objetivo foram realizadas duas revisões sistemáticas e uma revisão de literatura. A partir do conhecimento adquirido dessas revisões foi modelado e desenvolvido um teclado virtual assistivo. Além disso, foi elaborada e implementada uma metodologia evolutiva que permite o teclado se adaptar ao vocabulário e o modo de escrita do usuário. Finalmente, foi realizado um experimento que compara o método tradicional de otimização de teclados com a metodologia proposta nesta tese.

Palavras-chave: Síndrome do encarceramento ; Tecnologia assistiva; Comunicação alternativa e aumentativa; Teclado virtual; Metodologia evolutiva; Algoritmos genéticos

ABSTRACT

People with mobility and speech restrictions simultaneous have verbal communication and body language impaired. In the most extreme cases the patients are deprived of all their movements and speech capabilities. This situation is characterized as the Locked-in Syndrome (LIS). The augmentative and alternative communication technologies provide methods and softwares that allow these patients to communicate with the external environment and people around them. Among the various methods and existing communication programs we can highlight the virtual keyboard. However, data input using this kind of keyboard is considerably slower and more tiresome for people with LIS. This work aims to build an assistive virtual keyboard to assist patients with severe mobility restrictions and speech to communicate. To achieve this goal it was conducted two systematic reviews and a literature review. From the knowledge gained from these researchs we modeled and developed an assistive virtual keyboard. Moreover, we designed and implemented an evolutionary methodology that allows the keyboard to adapt itself to user's vocabulary and writing mode. Finally, an experiment that compares the traditional method keyboards optimization with the methodology proposed in this thesis was carried out.

Key-words: Locked-in Syndrome ; Assistive Technology; Augmentative and alternative communication; Virtual Keyboard; Evolutionary Methodology; Genetic Algor

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADA	*American with Disabilities Act*
CAA	Comunicação Alternativa e Aumentativa
GPC	Gestos por Caractere
IHC	Interface Homem Computador
KSPC	*Keystrokes per Chacratere*
MSD	*Minimum String Distance*
PPM	Palavras por minuto
SE	Sindrome do Encarceramento
T9	Texto em 9 teclas
TA	Tecnologia Assistiva
TNK	*Text in N Keys*
WPM	*Words per minute*

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

1.1 Introdução

Até o ano 2000 mais de 10% de toda população mundial possuía algum tipo de deficiência (SIK-LÁNYI; MOLNÁR-LÁNYI, 2000). Dados do Censo 2010 revelaram que quase 24% da população brasileira – 45,6 milhões de pessoas – tem algum tipo de deficiência. Dessas pessoas 7% tinham alguma restrição motora (INSTITUTO BRASILéIRO DE GEOGRAFIA E ESTATíSTICA, 2010). Essas restrições podem ser causadas por diversos fatores como Esclerose Lateral Amiotrófica, Paralisia Cerebral, Acidente Vascular Cerebral ou acidentes que lesionam o sistema nervoso.

As deficiências motoras podem limitar a movimentação do indivíduo. Essa limitação pode impedir o indivíduo de realizar ações básicas como escrever um texto, andar ou realizar um movimento que necessite de uma destreza mais fina. O estado de restrição mais grave é chamado de Síndrome do Encarceramento (SE). Os pacientes que possuem essa síndrome ficam impossibilitados de movimentar os membros superiores e inferiores, além de possuir sua habilidade de fala prejudicada.

Apesar, de toda a limitação proporcionada pela SE, a capacidade cognitiva e intelectual do paciente geralmente se mantém intacta. Nesse estado, os pacientes com SE ficam presos ao próprio corpo sem a possibilidade de comunicação com o ambiente externo.

Restaurar a capacidade básica de comunicação desses indivíduos pode melhorar sua qualidade de vida significativamente (MAK; WOLPAW, 2009b). Além de aumentar sua independência, reduzir seu isolamento social e minimizar os gastos com seus cuidados.

Com o objetivo de aumentar a qualidade de vida dos portadores de SE, várias áreas trabalham em conjunto. A engenharia desenvolve equipamentos que permitem a comunicação desse indivíduo por meio dos olhos ou sinais captados por via cerebral. Os *softwares* para comunicar com esses equipamentos são implementados pela computação. A área de enfermagem auxilia na identificação dos métodos de adaptação do equipamento às necessidades do paciente. Esse conjunto de esforços de diversas áreas com objetivo de auxiliar pessoas com deficiência constitui a área de conhecimento chamada Tecnologia Assistiva (TA).

A Tecnologia Assistiva proporciona formas, meios, recursos e métodos que visam que melhorar a qualidade de vida das pessoas com deficiência ou mobilidade reduzida. Uma subárea da TA é a Comunicação Aumentativa e Alternativa (CAA). Essa área reúne os métodos e tecnologias desenhadas para auxiliar ou substituir a comunicação oral de pessoas com limitação de fala (WILKINSON; HENNIG, 2007). Os sistemas computacionais implementados para auxiliar a comunicação desse tipo de paciente são chamados de sistemas de CAA. Esses sistemas podem ser segmentados em dois componentes distintos: dispositivo de entrada e *software* de comunicação.

O dispositivo de entrada é o equipamento utilizado para capturar qualquer tipo de intenção voluntária do paciente. Como por exemplo, equipamentos que identificam uma piscada voluntária ou impulsos cerebráis do paciente. O *software* de comunicação é o programa desenvolvido para analisar os dados capturados pelos dispositivos de entrada e transformá-los em informação. Esses programas são diversificados e abrangem desde teclados virtuais (DOVAL; CARBALLO; JEREMIAS, 2010; FU; HO, 2009; ORHAN et al., 2012) até complexas pranchas de comunicação (BISWAS; SAMANTA, 2008; MASON; CHINN, 2010).

As pranchas de comunicação possuem comandos predefinidos que proporcionam ao usuário a capacidade de se expressar de maneira rápida. Esses comandos podem ser selecionados diretamente pelos usuários e os permitem expressarem seus sentimentos, desejos ou necessidades rapidamente.

Apesar da eficiência das pranchas de comunicação, os usuários dos sistemas de CAA necessitam de meios para se comunicarem de forma detalhada, formal e específica. Nesse caso, os teclados virtuais ou *soft keyboards* podem ser considerados como uma alternativa para essa necessidade.

O teclado virtual é uma das alternativas mais primitivas de mecanismos de entrada de texto. Esse tipo de *software* substitui o teclado físico por uma representação gráfica do teclado na tela do computador (GHOSH, 2011). Entretanto, em comparação ao seu correlato físico sabe-se que esse tipo de solução possui uma baixa performance de digitação (KIM et al., 2014; KWON; LEE; CHUNG, 2009).

Essa performance é em geral mais baixa quando os teclados virtuais são utilizados por pessoas com SE. Pois, esses usuários estão aptos a transmitirem apenas um único tipo de estímulo. Essa restrição faz com que essa comunicação seja realizada de maneira objetiva (ativado ou desativado). Para alternar entre as opções do teclado, os pacientes com SE utilizam teclados que implementam técnicas de varredura.

O método de varredura consiste em percorrer o teclado virtual destacando suas teclas em sequência. Assim que a tecla desejada receber destaque o usuário transmite o estímulo. Esse estímulo é identificado pelo *software* e o programa seleciona a tecla ou a opção em destaque

(EMILIANI et al., 2009). Esse tipo de método de interação limita a performance de escrita do usuário e aumenta o esforço necessário para se digitar um texto (MIRó-BORRáS et al., 2009; EMILIANI et al., 2009). Para maximizar a quantidade de palavras escritas por minuto e diminuir o esforço de digitação é necessário otimizar o teclado virtual utilizado por pessoas com SE.

A proposta deste trabalho é desenvolver um teclado virtual assistivo, compacto e otimizado fundamentado na literatura. Esse teclado deve diminuir o esforço de digitação do usuário, aumentar a performance de entrada de dados e se adaptar ao modo de escrita e ao vocabulário do paciente.

Este capítulo apresenta a motivação, a justificativa, os objetivos e a estrutura deste trabalho. Finalmente, são realizadas as considerações finais deste capítulo.

1.2 Motivação

As pessoas com SE ficam totalmente restritas e em fases mais avançadas não são capazes de se comunicarem ou realizar qualquer tipo de atividade motora. Desenvolver um sistema de Comunicação Alternativa e Aumentativa para pacientes com SE pode proporcionar um aumento da qualidade de vida restaurando a sua comunicação com familiares, equipe de saúde e com o mundo externo.

Um dos dispositivos que possibilita reestabelecer essa comunicação é o teclado virtual assistivo. Entretanto, esse *software* possui um elevado esforço de digitação e uma baixa performance de entrada de dados. O esforço de digitação está relacionado a quantidade de interações necessárias entre o paciente e o software para inserir um texto. A performance de entrada de dados determina qual a velocidade que o paciente consegue digitar um determinado texto utilizando o teclado virtual.

Enquanto os teclados físicos podem produzir mais de 30 palavras por minuto, se operados por digitadores experientes (SIMATHAMANAND; PIROMSOPA, 2011; VARCHOLIK; LAVIOLA; HUGHES, 2012; HOSTE; SIGNER, 2013). Os teclados virtuais assistivos inserem uma média de quatro a sete palavras por minuto quando utilizados por uma pessoa com deficiência motora (MIRó-BORRáS et al., 2009). Assim, o teclado físico supera o teclado virtual assistivo em aproximadamente 400%, no melhor caso.

Assim, restaurar a comunicação dos pacientes com SE auxiliando-os a digitarem por mais tempo e mais rápido utilizando um sistema que se adapte as suas necessidades é a principal motivação desta tese.

1.3 Justificativa

No Brasil os estudos e as análises dos processos de pesquisa e desenvolvimento na área da TA ainda são escassos (GARCÍA; FILHO, 2012). Realizar um projeto nessa área pode proporcionar às pessoas com deficiência uma qualidade de vida melhor. Além disso, não foram encontradas pesquisas que indicam qual a melhor estratégia para se construir um teclado virtual assistivo para pacientes com SE.

Finalmente, o problema de distribuição de letras entre as teclas de um teclado virtual com o objetivo de diminuir o esforço de digitação e aumentar a performance de entrada de dados não pode ser resolvido de maneira trivial. A solução parcial desse problema requer uso de metaheurísticas avançadas. Novos estudos podem ser realizados com a finalidade de melhorar as soluções existentes na literatura.

1.4 Objetivos

O objetivo geral deste trabalho é desenvolver um teclado virtual assistivo, compacto, otimizado e evolutivo para pacientes com SE. Esse teclado poderá minimizar o esforço de digitação, aumentar a performance de entrada de dados e se adaptar ao usuário de acordo com o uso. Os objetivos específicos desta tese são:

- Investigar sobre Tecnologia Assistiva e Comunicação Alternativa e Aumentativa;
- Pesquisar o estado da arte dos teclados virtuais assistivos;
- Identificar as características, os métodos de otimização e as métricas aplicadas a teclados virtuais;
- Modelar e desenvolver um teclado virtual assistivo para pessoas com Síndrome do Encarceramento fundamentado na literatura;
- Elaborar uma metodologia de evolução de teclados virtuais assistivos de acordo com cada usuário; e
- Comparar a metodologia de evolução dos teclados virtuais com o método de otimização de teclados virtuais assistivos proposto na literatura.

1.5 Problema e Hipóteses

Os teclados assistivos possuem um esforço de digitação alto e uma performance de entrada de dados baixa (KIM et al., 2014; KWON; LEE; CHUNG, 2009). Esse problema ainda não foi resolvido. Porém, na literatura são encontradas diversas abordagens de teclados assistivos com o objetivo de solucionar essa questão. Todas as pesquisas nesse assunto possuem diversos avanços específicos que podem ser agrupados para construir um teclado assistivo otimizado. Gerar um teclado assistivo, compacto e otimizado, considerando diversas pesquisas realizadas nesta área pode proporcionar um ganho de qualidade de vida para pessoas com SE.

Porém, é importante ressaltar que cada tipo de usuário possui um vocabulário e um modo de escrita diferente. Os teclados assistivos propostos até o momento na literatura são otimizados com base em um apanhado de textos e não consideram as palavras que o usuário conhece e seu modo de escrita.

A hipótese deste trabalho é que otimizar o *layout* de teclados virtuais assitivos de acordo com o conhecimento do usuário resulta em melhores soluções do que otimizar os teclados virtuais utilizando o corpus de uma língua.

1.6 Metodologia de Pesquisa

A pesquisa realizada nesta tese possui um caracter exploratório, pois foi necessário esclarecer alguns detalhes do problema abordado. Esses detalhes foram esclarecidos de acordo com a literatura atual. Além disso, este trabalho pode ser classificado como bibliográfico, pois foram realizadas pesquisas em bases de conhecimento disponíveis digitalmente.

Assim, inicialmente foram realizadas duas revisões sistemáticas e uma revisão de literatura. O objetivo da primeira foi determinar o estado da arte da área de Tecnologia Assistiva e Comunicação Alternativa e Aumentativa. A finalidade da segunda revisão foi identificar as características do teclado virtual, as maneiras de otimizar o esforço e a performance de digitação. Finalmente, a terceira pesquisou sobre os métodos de otimização aplicados ao problema da distribuição de letras entre as teclas do teclado virtual.

A análise dessas revisões originaram um protótipo de teclado virtual assistivo com abordagens mais indicadas para pessoas com SE. Além disso, ao final das revisões foi elaborada a hipótese deste trabalho.

Com o objetivo de provar essa hipótese foi construída uma metodologia de otimização de teclados virtuais assistivos diferente do método de otimização tradicional. Em seguida, foi realizado um experimento para verificar a eficiência do método proposto. Finalmente, foi efe-

tuado outro experimento para determinar a maneira mais indicada de utilizar a metodologia proposta nesta tese.

1.7 Contribuições

As contribuições deste trabalho são:

- Contribuições diretas
  - Elaboração e implementação parcial de um teclado virtual assistivo para comunicação de pacientes com SE;
  - Construção e desenvolvimento de uma metodologia de evolução continua de teclados assistivos; e
  - Elaboração e implementação de um método de desambiguidade parcial.
- Publicações de Artigos
  - Publicação de um artigo de revisão sistemática na Revista Brasileira de Engenharia Biomédica intitulado "*A Concept-Environment for Computer-Based Augmentative and Alternative Communication Founded on a Systematic review*";
  - Publicação de um capítulo no livro "Formação e Políticas Públicas na Educação: Tecnologias, Aprendizagem, Diversidade e Inclusão" com o título "Tecnologia Assistiva: Mapeando a Área";
  - Publicação de um artigo com uma parte do resultado de uma revisão sistemática na revista IEEE Latina America intitulado "*A Systematic Review on Methods and Techniques for Optimizing Assistive Virtual Keyboards*" ;
  - Publicação de um artigo de revisão sistemática na Revista Brasileira de Engenharia Biomédica com o título "*A New Concept of Assistive Virtual Keyboard Based on a Systematic Review of Text Entry Optimization Techniques*";
  - Publicação de um artigo de revisão literária na XIII Conferência de Estudos em Engenharia Elétrica intitulado "Revisão sobre otimizações de layouts de teclados virtuais voltados a tecnologia assistiva".; e
  - Publicação de um artigo na XIII Conferência de Estudos em Engenharia Elétrica intitulado "Um estudo preliminar do protocolo de navegação de interfaces gráficas baseado na codificação de Huffman".
- Projetos de Pesquisa

- Projeto aprovado no Programa de Apoio à Produtividade em Pesquisa (PROAPP) com o título "Um Layout de Teclado Virtual Otimizado para Pessoas com Afasia e Deficiência Motora Grave". Nesse projeto está sendo orientado o trabalho de conclusão de curso de um aluno de graduação em Técnico de Análise de Desenvolvimento de Sistemas; e
- Projeto aprovado no Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação (PIBIT) intitulado "Um *software* de interação homem-computador utilizando o piscar de olhos". Nesse projeto estão sendo orientadas três alunas de iniciação científica.

1.8 Estrutura deste Trabalho

Este trabalho está estruturado em sete capítulos:

Este capítulo apresenta a motivação, os objetivos e a estrutura deste trabalho. Finalmente, são realizadas as considerações finais deste capítulo.

O Capítulo 2 mostra uma introdução a área de conhecimento de Tecnologia Assistiva e Comunicação Alternativa e Aumentativa e uma parte da revisão sistemática realizada para pesquisar o estado da arte dessas áreas. No final desse capítulo descreve-se o conceito de sistema de CAA e a proposta geral de um novo ambiente de Comunicação Alternativa e Aumentativa.

O Capítulo 3 apresenta parte da revisão sistemática realizada com o objetivo de pesquisar as principais características do teclado virtual, os métodos de otimização e as métricas utilizadas para medir a performance e o esforço desse tipo de teclado. Finalmente, são descritos os trabalhos correlatos relacionados a teclados virtuais.

O Capítulo 4 mostra uma revisão de literatura realizada com o objetivo de identificar os trabalhos relacionados à otimização da distribuição das letras em relação as teclas do teclado virtual. Nesse capítulo, é apresentado o problema de distribuição de letras em relação as teclas, os métodos aplicados para resolver parcialmente esse problema e uma análise dos trabalhos correlatos. Finalmente, descreve-se uma fundamentação teórica sobre algoritmos genéticos e suas principais características.

O Capítulo 5 apresenta o teclado assistivo, compacto, otimizado e evolutivo proposto neste trabalho e define todas as características, os métodos e as métricas utilizadas para desenvolver esse teclado. Todas as funcionalidades adotadas para construir o teclado virtual são justificadas a partir da literatura pesquisada nesta tese. Finalmente, é apresentada a metodologia evolutiva de teclados virtuais assistivos.

O Capítulo 6 apresenta um experimento com a finalidade de verificar a eficiência da

metodologia evolutiva de teclados assistivos comparada ao método de otimização de teclados utilizado na literatura. Além disso, esse capítulo efetua um segundo experimento para determinar a maneira mais indicada de utilizar a metodologia proposta nesta tese.

O Capítulo 7 apresenta as conclusões e os trabalhos futuros que poderão ser realizados a partir desta tese.

1.9 Considerações Finais

Este capítulo apresentou a motivação, a justificativa, os objetivos, o problema, a hipótese, a metodologia de pesquisa, as contribuições desta tese e a estrutura deste trabalho.

O próximo capítulo mostra uma introdução à área de conhecimento de Tecnologia Assistiva e Comunicação Alternativa e Aumentativa e uma parte da revisão sistemática realizada para pesquisar o estado da arte dessas áreas. No final desse capítulo descreve-se o conceito de sistema de CAA e a proposta geral de um novo ambiente de Comunicação Alternativa e Aumentativa.

CAPÍTULO 2

TECNOLOGIA ASSISTIVA E COMUNICAÇÃO AUMENTATIVA E ALTERNATIVA

2.1 Introdução

Atualmente a sociedade vem pesquisando tecnologias e técnicas com a finalidade de auxiliar a inclusão social de indivíduos com deficiência (GARCÍA; FILHO, 2012). Essa tendência originou uma nova área de conhecimento denominada Tecnologia Assistiva (TA). Cook e Hussey (2001) definem TA mencionando o conceito elaborado por (COLKER, 1999):

> "Uma ampla gama de equipamentos, serviços, estratégias e práticas concebidas e aplicadas para minorar os problemas funcionais encontrados por indivíduos com deficiências."

Uma sub-área da Tecnologia Assistiva é a Comunicação Aumentativa e Alternativa (CAA). Essa área reúne os métodos e as tecnologias desenhadas para auxiliar ou substituir a comunicação oral de pessoas com limitação de fala (WILKINSON; HENNIG, 2007). As pessoas que possuem restrições motoras e de fala simultâneas apresentam a comunicação verbal e a linguagem corporal prejudicadas. Nos casos mais extremos o paciente é privado de todos os seus movimentos e da capacidade de fala. Essa situação extrema é caracterizada como a síndrome do encarceramento (SE).

O objetivo dos ambientes de comunicação de CAA é permitir a comunicação das pessoas portadores de SE com os seus semelhantes e com o mundo externo. Esses ambientes são compostos por métodos e *softwares* que possibilitam essa comunicação.

Este capítulo descreve os conceitos de Tecnologia Assistiva e de Ambientes de Comunicação Aumentativa e Alternativa. Posteriormente, é apresentada uma proposta para um novo ambiente de CAA. E finalmente, são realizadas as considerações finais deste capítulo.

2.2 Tecnologia Assistiva

A Tecnologia Assistiva é uma expressão nova e está em processo de construção, porém a utilização dos recursos dessa tecnologia remonta aos primórdios da história da humanidade. Por exemplo, qualquer pedaço de pau utilizado como uma bengala improvisada, caracteriza o uso de um recurso de TA (GARCÍA; FILHO, 2012). Assim, essa linha de pesquisa permeia várias áreas de conhecimento conduzindo a multidisciplinaridade.

Considerando pacientes que possuem algum tipo de amputação dos membros inferiores, esse tipo de deficiência é suprimida com o uso de uma simples muleta. Esta solução compreende duas áreas de conhecimento. A área de saúde responsável por definir as especificações do equipamento e a área de engenharia que projeta e constroi o dispositivo.

Infelizmente, existem deficiências mais severas como: a Síndrome do Encarceramento e a Esclerose Lateral Amiotrófica (ELA). Essas deficiências podem manter a capacidade cognitiva e intelectual do paciente intacta. Porém, elas podem causam tetraplegia e restringem a capacidade de fala de seus portadores.

Nesses casos, várias áreas trabalham em conjunto com o objetivo de aumentar a qualidade de vida da pessoa com deficiência. A engenharia desenvolve tecnologias que permitem a comunicação dessa pessoa pelos olhos ou pelo cérebro. Os *softwares* para comunicar com esses equipamentos são implementados pela computação. A área de enfermagem auxilia na identificação dos métodos de adaptação do equipamento às necessidades do paciente. O sistema de comunicação permite que os portadores de deficiência tenham uma vida mais digna e sejam capazes de exprimir suas vontades.

A Tecnologia Assistiva originou da expressão *assitive technology*. Essa expressão foi elaborada em 1988 e registrada no contexto da legislação que regula os direitos dos cidadãos estadunidenses com deficiência, conhecida desde de 1998 como *American with Disabilities Act* (ADA). A tradução de *assistive technology* para a expressão tecnologia assistiva foi sugerida por Sassaki. Essa tradução serviu para substituir diferentes denominações adotadas até então. Além disso, ela unificou a terminologia referente aos mais diversos suportes que as pessoas com deficiência necessitam. Sassaki, definiu tecnologia assistiva como:

> "A tecnologia destinada a fornecer suporte (mecânico, elétrico, eletrônico, computadorizado etc.) a pessoas com deficiência física, visual, auditiva, mental ou múltipla. Esses suportes, então, podem ser uma cadeira de rodas de todos os tipos, uma prótese, uma órtese, uma série infindável de adaptações, aparelhos e equipamentos nas mais diversas áreas de necessidade pessoal (comunicação, alimentação, mobilidade, transporte, educação, lazer, esporte, trabalho e outras)" (SASSAKI, 2007).

É com este significado que a expressão passou a ser incorporada ao discurso oficial e à legislação brasileira, ampliando o entendimento e a abrangência do que antes era considerado, na literatura especializada e na legislação, sob a denominação de expressões, como "tecnologias de apoio", "tecnologias adaptativas", "tecnologias de reabilitação", "adaptações" e "ajudas técnicas", que se referiam apenas a produtos.

No Brasil, o Comitê de Ajudas Técnicas (CAT), vinculado à Secretaria Nacional de Promoção das Pessoas com Deficiência (SNPD), órgão da Secretaria de Direitos Humanos da Presidência da República, formulou o seguinte conceito dessa tecnologia:

> "Tecnologia Assistiva é uma área do conhecimento, de característica interdisciplinar que compreende produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e a participação de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social" (TéCNICAS, 2007).

Os recursos de TA são organizados de acordo com seus diferentes objetivos funcionais. Uma classificação utilizada em trabalhos Internacionais é proposta pela ISO 9999:2002, substancialmente voltada para os produtos. A ISO classifica TA em:

a) Tratamento médico pessoal;

b) Treinamento de habilidades;

c) Órteses e próteses;

d) Proteção e cuidados pessoais;

e) Mobilidade pessoal;

f) Cuidados com o lar;

g) Mobiliário e adaptações para residencias e outras edificações;

h) Comunicação e informação;

i) Manuseio de objetos e equipamentos;

j) Melhorias ambientais, ferramentas e máquinas; e

k) Lazer

Essa categorização limita TA quanto aos serviços, pois define ajudas técnicas como:

Qualquer produto, instrumento, equipamento ou sistema tecnológico, de produção especializada ou comumente à venda, utilizado por pessoa com deficiência para prevenir, compensar, atenuar ou eliminar uma deficiência, uma incapacidade ou uma desvantagem.

Uma outra definição para as sub-áreas de TA é realizada pela Horizontal European Activities in Rehabilitation Technology (HEART). A HEART divide TA em 6 categorias. O objetivo dessas subdivisões é atender os usuários de TA, com a finalidade de auxiliá-los a realizar a escolha adequada de produtos que os ajudem nas tarefas do cotidiano. Diferente da ISO, a HEART organiza TA em componentes, técnicos, humanos e sociais (GARCÍA; FILHO, 2012). A Tabela 2.1 mostra essa classificação.

Tabela 2.1 – Classifição HEART

Fonte: (GARCÍA; FILHO, 2012)

Componentes técnicos	Comunicação Mobilidade Manipulação Orientação
Componentes Humanos	Tópicos sobre a deficiência Aceitação de TA Seleção de TA Aconselhamento em TA Atendimento pessoal
Componentes socioeconômicos	Noções básicas de TA Noções básicas de desenho universal Emprego Prestação de serviços Normalização/Qualidade Legislação / Economia Recursos de informação

Diferente das classificações apresentadas, José Tonolli e Rita Bersch descreveram uma outra classificação para a área de TA (BERSCH, 2008). Esses pesquisadores fundamentaram-se na existência de recursos e serviços para cada sub-área definida. Essa classificação foi desenvolvida com base em outras classificações, dentre elas, na Classificação Nacional de Tecnologia Assistiva do Departamento de Educação dos Estados Unidos.

A categorização proposta por José Tonolli e Rita Bersch consiste no conhecimento adquirido no Programa de Certificação em Aplicações da Tecnologia Assistiva – ATACP da California State University Northridge, College of Extended Learning and Center on Disabilities (BERSCH, 2008). A finalidade dessa classificação é organizar a TA de forma acadêmica. A tabela 2.2 apresenta as sub-áreas de TA, de acordo com a Classificação Nacional de Tecnologia Assistiva do Departamento de Educação dos Estados Unidos.

Devido a interdisciplinaridade e a incipiência da TA foram definidas várias definições sobre essa área de conhecimento. Todos os conceitos sobre essa linha de pesquisa são fundamentados no esforço da sociedade em promover a inclusão social das pessoas portadoras de deficiência.

Neste trabalho é utilizada a classificação proposta por José Tonolli e Rita Bersh (BERSCH,

2008). Pois essa definição é voltada a área acadêmica e abrange o conceito de Comunicação Aumentativa e Alternativa (CAA) que é o objetivo principal desta tese.

Tabela 2.2 – Sub-áreas da TA.

Fonte: (GARCÍA; FILHO, 2012)

Sub-Área	**Objetivo**
Auxílios para a vida diária e a vida prática	Materiais ou produtos que facilitam as funções básicas do cotidiano, para o uso ou o auxílio.
Comunicação Aumentativa e Alternativa	Reúne os métodos e as tecnologias desenhadas para auxiliar pessoas sem fala ou escrita funcional ou em defasagem.
Recursos de acessibilidade ao computador	Todo hardware e *software* modificado para tornar acessível o computador.
Sistemas de controle de ambiente	Agrupa todas as funções para ajustar os aparelhos eletroeletrônicos em um controle remoto para ser utilizado por aqueles com limitações motoras.
Projetos arquitetônicos para acessibilidade	Adaptações estruturais e reformas na casa ou ambiente de trabalho que garantem o acesso e a mobilidade.
Órteses e próteses	Uma prótese substitui um membro ou uma parte do organismo. Órteses são apoios ou dispositivos externos aplicados ao corpo para modificar os aspectos funcionais.
Adequação Postural	Recursos que auxiliam e estabilizam a postura deitada e de pé
Auxílios de mobilidade	A mobilidade pode ser auxiliada por vários instrumentos. Desde bengalas até cadeiras de rodas.
Auxílios para a qualificação da habilidade visual e os recursos que ampliam a informação a pessoas com baixa visão ou cegas.	Reúne equipamentos como: auxílios ópticos, lentes, lupas manuais e lupas eletrônicas.
Auxílios a pessoas com surdez ou com déficit auditivo	Conjunto de equipamentos e serviços que possibilitam as pessoas com déficit auditivo se comunicarem. Livros, textos e dicionários digitais em língua de sinais. Sistema de legendas (*close-caption/subtitles*).
Mobilidade em veículos	Adaptações em veículos automotores que garantem a mobilidade de pessoas com deficiência física.
Esporte e Lazer	Recursos que possibilitam a prática de esporte e participação em atividades de lazer.

2.3 Sistemas de Comunicação Aumentativa e Alternativa

A Comunicação Aumentativa e Alternativa reúne os métodos e as tecnologias projetadas para auxiliar ou substituir a comunicação oral de pessoas com limitação de fala (WILKINSON; HENNIG, 2007). Park et al. concluíram que devido à crescente popularização dos

computadores e à expansão da internet, estão sendo desenvolvidos vários trabalhos para auxiliar a comunicação de pessoas com afasia e tetraplegia (PARK et al., 2012).

Esses pacientes possuem restrições motoras e de fala, ou seja, a comunicação verbal e a linguagem corporal são prejudicadas. O caso mais extremo é a síndrome do encarceramento (SE). Na SE a pessoa perde completamente suas funções motoras dos membros superiores e inferiores, mantendo apenas estímulos motores limitados como, por exemplo, o movimento dos olhos (KEEGAN; BURKE; CONDRON, 2009).

Existem três tipos de SE: incompleta, clássica e completa (SORGER et al., 2009; MAK; WOLPAW, 2009a). Na SE incompleta o indivíduo possui movimentos voluntários limitados, como o movimento de um dedo ou parte do rosto. Pessoas que possuem a SE clássica conseguem apenas movimentar os olhos e piscá-los. Finalmente, na SE completa o paciente fica impossibilitado de realizar movimentos voluntários em qualquer parte do corpo. Essa síndrome pode ser causada por doenças neurodegenerativas como a esclerose lateral amiotrófica (ELA), por paralisia cerebral e até mesmo por tetraplegia (CARDWELL et al., 2012).

A palavra neurodegenerativa é composta do prefixo "neuro", que significa células nervosas, e "degenerativa" que neste caso tem o sentido de perda da função ou desestruturação de um tecido ou órgão. Portanto, doenças neurodegenerativas correspondem a qualquer condição patológica que acomete os neurônios. Essas doenças representam um grande grupo de doenças neurológicas com expressões heterogêneas clínicas e patológicas que afetam um subconjunto de neurônios que possuem funções específicas do sistema anatômico. Essa degeneração surge por razões desconhecidas e progride inevitavelmente (PRZEDBORSKI; VILA; JACKSON-LEWIS, 2003).

Esse tipo de doença causa a morte ou o mau funcionamento das células do sistema nervoso de forma excessiva e progressiva de determinadas áreas do cérebro. Portanto, pacientes portadores de doenças neurodegenerativas podem perder sua capacidade cognitiva e as funções motoras. No caso da ELA, a parte cognitiva em geral permanece intacta enquanto a parte motora é totalmente danificada. Segundo Krug e Amaral (2002), ELA é um distúrbio neurodegenerativo de origem desconhecida, progressivo e associado à morte do paciente em um tempo médio de três a quatro anos. Porém ha casos de sobrevida longa. Sua incidência é estimada em 1 a 2,5 indivíduos portadores a cada 100.000 habitantes por ano, com uma prevalência de 2,5 a 8,5 por 100.000 habitantes.

Paralisia Cerebral é uma desordem do movimento e da postura, persistente, porém variável, surgida nos primeiros anos de vida pela interferência no desenvolvimento do sistema nervoso central, causada por uma desordem cerebral não progressiva (SCHWARTZMAN, 1993). Esse tipo de deficiência pode causar danos irreversíveis ao sistema motor, impedindo que o paciente realize ações que necessitam de destreza fina e até mesmo impossibilitando o movimento

dos membros superiores e/ou inferiores. Segundo (HERRERO; MONTEIRO, 2008), essa deficiência pode ocorrer devido a fatores hereditários, eventos ocorridos durante a gravidez, parto, período neonatal ou durante os primeiros dois anos de vida.

A tetraplegia pode ser causada pelas doenças citadas nos parágrafos acima, ou por algum dano causado na espinha dorsal do paciente. Nessa situação, as capacidades cognitivas dos pacientes permanecem intactas. Em casos extremos nos quais a capacidade de fala também é prejudicada, essas pessoas ficam impossibilitadas de estabelecer comunicação com seus semelhantes e com o meio em que vivem (CALTENCO et al., 2012).

Por não terem poder de comunicação oral nem gestual, esses pacientes necessitam de dispositivos que forneçam meios alternativos de comunicação. Assim, uma maneira de viabilizar a comunicação desses indivíduos é realizada por meio dos sistemas de CAA. Fundamentado em Hill (2010) este trabalho propõe uma segmentação dos ambientes de comunicação aumentativa e alternativa em três componentes: primário, secundário e terciário.

Os **componentes primários** indicam a maneira como o ambiente de comunicação representa a linguagem "natural". A interface com o usuário, os métodos de controle e seleção e a saída do lsoftware são **componentes secundários**. Os **componentes terciários** mostram a relação entre a adequação e a expectativa do usuário ao utilizar o sistema de comunicação.

2.3.1 Componentes primários

A linguagem utilizada pelos sistemas de CAA pode ser representada por figuras com um único significado, técnicas que utilizam o alfabeto ou a compactação semântica. Esses conceitos são independentes da tecnologia e são utilizados para definir as estratégias de comunicação dos dispositivos de CAA (HILL, 2010).

A representação da linguagem "natural" por figuras usa símbolos gráficos para representarem uma palavra ou uma mensagem. Esses símbolos podem ser fotos, desenhos, gráficos ou animações. De acordo com (HILL, 2010), esse tipo de representação da linguagem necessita de um grande banco de dados de imagens. Porém, os sistemas que utilizam essa representação facilitam a comunicação, além de permitir que usuários não alfabetizados se comuniquem. A Figura 2.1 mostra esse modo de representação da linguagem.

A representação por alfabeto utiliza a ortografia e as técnicas de otimização de entrada de informação. Os sistemas que usam essa representação utilizam teclados virtuais, ou um vetor com alfabeto ou até mesmo uma lista de letras e palavras para gerar uma mensagem. Para usar esse tipo de sistema é necessário que o usuário saiba ler e escrever (HILL, 2010). A Figura 2.2 ilustra esse tipo de comunicação representada por um teclado virtual comum.

A representação por compactação semântica ou ícones de multi significado foi proposta

Figura 2.1 – Modo de representação da linguagem usando apenas figuras.

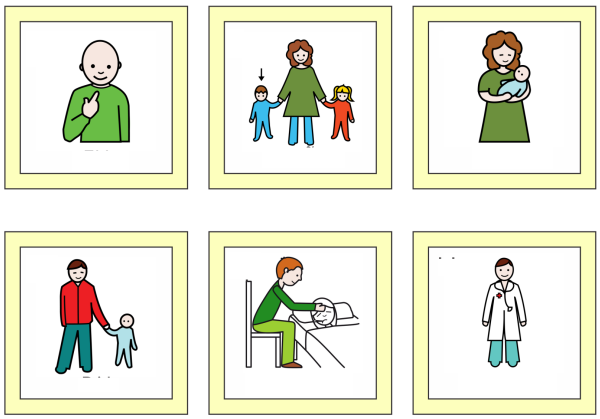


Figura 2.2 – Modo de representação da linguagem utilizando apenas o alfabeto, números e caracteres especiais.

e patenteada por (BAKER, 1986). Esse método utiliza um conjunto de ícones para representar uma palavra e/ou uma frase. De acordo com o autor Hill (2010), os usuários que usam os sistemas com este tipo de representação não precisam ser alfabetizados.

Nos trabalhos de (USAKLI et al., 2009; ARBOLEDA et al., 2009), os autores adotaram uma estratégia híbrida utilizando tanto imagens quanto caracteres para substituir a linguagem "natural". As Figuras 2.3 e 2.4 mostram as propostas de comunicação de (USAKLI et al., 2009) e (ARBOLEDA et al., 2009), respectivamente.

Arboleda et al. (2009) apresentaram uma matriz de quatro linhas por três colunas. Nesta matriz o usuário pode escrever uma mensagem ou selecionar as atividades diárias, como por exemplo, pedir água ou indicar se está bem ou mal. O sistema apresenta automaticamente as imagens e as letras e o paciente envia um sinal escolhendo a opção desejada. Essa estratégia permite o paciente se comunicar utilizando as letras e formando um texto ou expressar uma ação rápida como desejo de ir para cama.

Na planilha de comunicação desenvolvida por Usakli e Gurkan (2009), o usuário pode escolher as ações entre várias imagens e escrever textos mais complexos utilizando um teclado virtual. Essa abordagem também permite controlar o *mouse* por meio de setas que representam a direção do movimento. Prabhu e Prasad (2011), Doval, Carballo e Jeremias (2010), Park et al.

Figura 2.3 – Modo híbrido de representar a linguagem proposto por Usakli et al. (2009) usando simbolos, números e letras. As setas podem ser utilizadas para mover o mouse. As figuras permitem ao usuário uma comunicação mais rápida, equanto as letras proporcionam uma comunicação mais detalhada.

Fonte: (USAKLI et al., 2009)

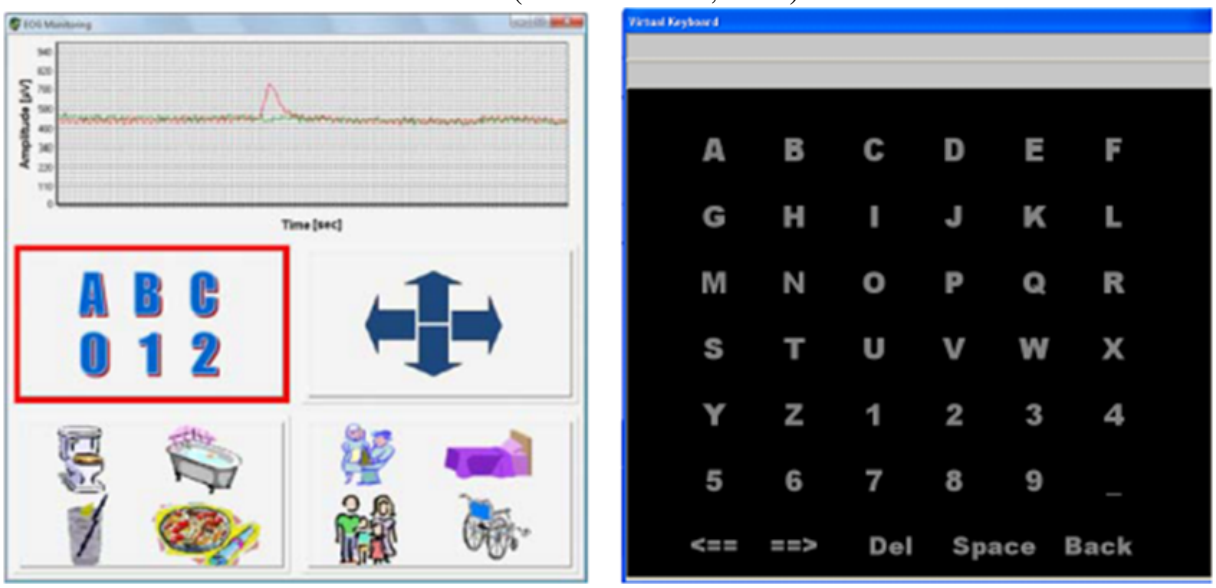


Figura 2.4 – Interface proposta por Arboleda et al. (2009) para representar a linguagem com simbolos e letras.

Fonte: (ARBOLEDA et al., 2009)

(2012) usaram somente o alfabeto para representar a linguagem “natural”.

Em (SILVA; PEREIRA, 2011) a linguagem “natural” foi representada por compactação semântica. Estas imagens são organizadas em uma matriz de símbolos e descrições. Os símbolos representam alguma informação ou comando que o usuário pode realizar. O usuário seleciona a informação que deseja transmitir para o receptor selecionando um elemento dessa matriz e compondo a frase e/ou as palavras. A Figura 2.5 mostra a interface proposta por Silva.

Figura 2.5 – Interface utilizando compactação semântica proposta por Silva e Pereira (2011). Esse modo de representação permite construir sentenças usando apenas figuras.

Fonte: (SILVA; PEREIRA, 2011)

2.3.2 Componentes secundários

Os componentes secundários são dependentes da tecnologia utilizada no desenvolvimento do ambiente de CAA. Esta tese segmenta esses componentes em duas partes distintas: dispositivo de entrada e *software* de comunicação. Essa segmentação acontece devido a distinção entre esses dois componentes do sistema. Assim, cada componente pode ser desenvolvido independentemente.

O dispositivo de entrada é o equipamento usando para capturar qualquer tipo de intenção voluntária do paciente. As pessoas portadoras de SE para se comunicarem por meio desses dispositivos estão aptas a emitirem 5 tipos de sinais biológicos que podem ser convertidos em comandos computacionais. Esses sinais são adquiridos a partir do movimento dos olhos, das atividades cerebrais, por meio do sopro, da capacidade de fala parcial ou das expressões faciais.

O dispositivo de entrada transforma os sinais biológicos em informações de entrada decodificadas pelo computador. Por exemplo, o sinal do movimento dos olhos pode ser adquirido por meio de câmeras de vídeo (PARK et al., 2012) ou pela leitura do sinal dos músculos oculares pelo Eletrooculograma (EOG) (USAKLI; GURKAN, 2009). O dispositivo de entrada é definido conforme o estímulo que o paciente consegue executar.

Park et al. (2012) e Cipresso et al. (2011) apresentaram pesquisas usando como entrada

de dados o movimento dos olhos. Os trabalhos de Al-Abdullatif et al. (2013), Blain, Mihailidis e Chau (2008), Schalk et al. (2008) e Usakli e Gurkan (2009) utilizaram as atividades cerebrais do paciente na comunicação. Poláček, Míkovec e Slavík (2012) apresentaram um dispositivo de comunicação que utiliza o sopro como entrada. Ann e Theng (2011) e Sorger et al. (2009) analisaram os movimentos faciais como informação de entrada e finalmente Hanson et al. (2010) implementaram um interpretador vocal para o reconhecimento da fala degradada.

Poláček, Míkovec e Slavík (2012) construíram um teclado virtual que interage com o paciente por meio do sopro. Um pequeno aparelho fica posicionado na boca do paciente que emite um sinal assoprando o dispositivo. Esse dispositivo captura a interação do paciente por meio do sopro e a transforma em um comando de entrada.

As pessoas que possuem a fala parcialmente íntegra podem utilizar sistemas de predição de palavras e processadores de voz para se comunicarem. Hanson et al. (2010) mostram um protótipo de um dispositivo de comunicação que interpreta uma fala degradada por um processador de voz e gera como saída uma expressão sintetizada.

Ann e Theng (2011) desenvolveram um sistema para pessoas com paralisia cerebral. Neste sistema, as mensagens são definidas em um banco de dados e o usuário não possui a flexibilidade de elaborar novas mensagens. Essas mensagens são vinculadas a determinadas expressões faciais do paciente. A interação entre o usuário e o sistema é realizada pelo reconhecimento dessas expressões faciais adquiridas pela webcam. Assim que uma expressão é identificada a mensagem relacionada a ela é emitida.

Para uma pessoa com SE a fala e o movimento dos músculos faciais são ações restritas (SORGER et al., 2009). A preservação dos movimentos oculares pode facilitar a comunicação não verbal (SMITH, 2004). O movimento dos olhos é um estímulo utilizado como interface homem-computador. Este estímulo pode ser adquirido pelo processamento da imagem do olho (PARK et al., 2012) ou pelo sinal da atividade dos músculos oculares (KEEGAN; BURKE; CONDRON, 2009).

Park et al. (2012) desenvolveram um trabalho com o objetivo de pesquisar CAA na Coreia do Sul e apresentaram um novo dispositivo de entrada. Esse dispositivo utiliza uma mini câmera acoplada a um óculos para capturar o piscar dos olhos dos pacientes portadores de SE. Para interagir com a interface proposta por Park é necessário uma calibração do *software* de reconhecimento de piscadas. A Figura 2.6 ilustra o dispositivo usado no trabalho de (PARK et al., 2012).

O trabalho de (PANWAR; SARCAR; SAMANTA, 2012) tentou solucionar os problemas relacionados a movimentação do *mouse* realizada pelos movimentos oculares. Nesse trabalho, o *software ITU Gazetracker* é utilizado para monitorar os movimentos dos olhos. Esses movimentos são convertidos em coordenadas relacionadas a tela do computador que possibili-

Figura 2.6 – Dispositivo de entrada adaptado a um óculos proposto por Park et al. (2012). O usuário pode interagir com o sistema por meio da piscada.

Fonte: (PARK et al., 2012)

tam o *mouse* se movimentar pelo olhar. A Figura 2.7 ilustra o uso do dispositivo de entrada.

Figura 2.7 – Dispositivo de entrada proposto por Panwar, Sarcar e Samanta (2012) utilizando o *software ITU Gazetracker*. O usuário interage com o sistema usando os olhos.

Fonte: (PANWAR; SARCAR; SAMANTA, 2012)

Usakli et al. (2009) propuseram uma abordagem híbrida que utiliza o eletrooculograma (EOG) e o eletroencefalograma (EEG) para facilitar comunicação de pacientes com SE. A finalidade do EOG é capturar os potencias entre a córnea e a retina. Esse potencial é utilizado como estimulo de entrada. O EEG é usado para capturar os impulsos cerebrais e validar a leitura realizada pelo EOG.

Na SE completa o paciente pode ficar totalmente restrito a impulsos cerebrais. Nesses casos, a única solução até o presente momento é usar o *Brain Computer Interface* (BCI). Os sistemas de BCI utilizam as características das atividades cerebrais e as codificam em forma de sinais de controle para diversos dispositivos (SCHALK et al., 2008).

Os sinais das atividades cerebrais podem ser adquiridos de diversas formas. A natureza desses sinais é o fator mais significativo para a escolha das técnicas de BCI. O sinal pode ser eletrofisiológico ou hemodinâmico (SORGER et al., 2009). A escolha da técnica de BCI a ser utilizada depende de fatores financeiros, espaço físico, locomoção do dispositivo, tempo de resposta, precisão, adequação do usuário e complexidade da instalação (MAK; WOLPAW, 2009a).

A operação do BCI é organizada nas seguintes atividades: aquisição do sinal, extração da característica, codificação da característica, saída do dispositivo e protocolo de operação

(MAK; WOLPAW, 2009a). Durante a pesquisa desta tese foram encontrados trabalhos relacionados às atividades de operação do BCI (SCHALK et al., 2008; CHIN; LEE; LEE, 2010; OKASAKA; HOSHINO, 2012; BESIO; KAY; LIU, 2009; SUN; HU; WU, 2010; THOMAS et al., 2008; DEEPA; THANGARAJ; CHITRA, 2010) e dos sistemas de CAA completos (ARBOLEDA et al., 2009; PRABHU; PRASAD, 2011).

Também existe a possibilidade de desenvolver sistemas que utilizam mais de uma técnica de Interface Homem-Computador. Nos trabalhos de (CIPRESSO et al., 2011; USAKLI; GURKAN, 2009) foram usados o movimento dos olhos e o BCI. Todos esses dispositivos transmitem sinais que devem ser decodificados e posteriormente transformados em informções. Os *softwares* de comunicação são responsáveis por realizarem essa transformação.

Os *softwares* de comunicação são os programas desenvolvidos para analisar os dados capturados pelos dispositivos de entrada e transformá-los em informação. Esses programas são diversificados e incluem desde teclados virtuais (FU; HO, 2009; DOVAL; CARBALLO; JEREMIAS, 2010; ORHAN et al., 2012) até planilhas complexas de comunicação (BISWAS; SAMANTA, 2008; MASON; CHINN, 2010).

Prabhu e Prasad (2011) desenvolveram um teclado circular dividido em setores e um módulo de predição de palavras. Nesse teclado as letras são agrupadas nos setores. Ao selecionar um setor, o teclado apresenta uma lista com as suas respectivas letras. Na predição de palavras, o usuário escreve apenas algumas letras e o restante da palavra é completado pela lista de opções apresentadas pelo teclado.

Doval, Carballo e Jeremias (2010) desenvolveram um teclado virtual sem as teclas de funções e agruparam as consoantes, vogais, números e pontuações em teclas específicas. As consoantes, as vogais e os números são diferenciados pelas cores das teclas.

2.3.3 Componentes terciários

Os principais fatores que fazem com que o usuário abandone os sistemas de CAA são: o suporte inadequado, o treinamento, a manutenção e os ajustes dos fornecedores do sistema de CAA (KRASKOWSKY; FINLAYSON, 2001; JOHNSON et al., 2006).

O usuário deve estar sempre seguro do que o sistema pode lhe oferecer em termos de comunicação. Além disso, é importante que os usuários de CAA saibam usar todo o sistema. Assim, o treinamento dos usuários na utilização de sistemas de CAA deve ser realizado periodicamente (HILL, 2010).

O suporte por telefone proporciona a oportunidade de otimizar o treinamento dos pacientes em relação ao sistema de CAA (HILL, 2010). Se realizado adequadamente, esse tipo de suporte pode potencializar a utilização do sistema e melhorar a qualidade de vida do paciente.

A aplicação dos métodos de interface Homem-Computador necessita de esforço na instalação, calibração e operação (KUBLER et al., 2006). Para realizar essas etapas é necessário um bom suporte para os sistemas de CAA. Porém, os custos de manutenção e suporte técnico são altos e podem desestimular a continuidade do uso dessa tecnologia (SIMPSON, 2008).

Outro fator importante a ser considerado é a complexidade da interface. Interfaces muito complexas podem gerar insatisfação no usuário diante das dificuldades de se adaptar a tecnologia (MCCOY et al., 2013). Além disso, o que é complexo ou simples para um usuário pode não ser para um outro. Assim, os sistemas de CAA devem possuir interfaces simples, intuitivas e que possam se adaptar a cada usuário.

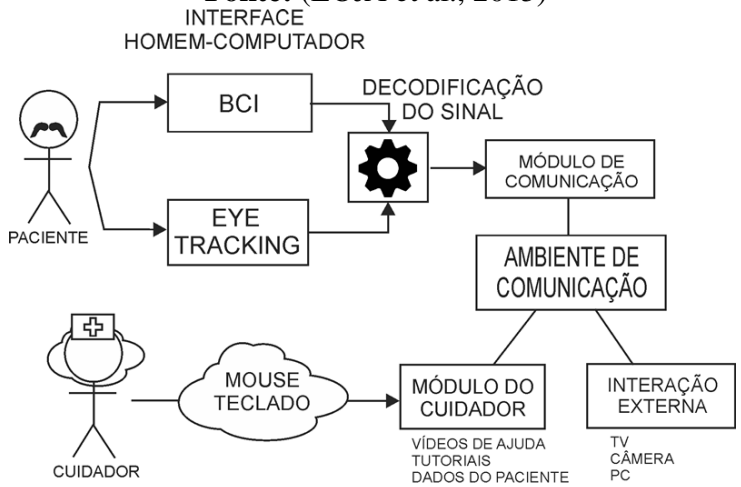
2.4 Proposta de Ambiente para a Comunicação Aumentativa e Alternativa

Os trabalhos pesquisados nesta tese apresentaram uma ou mais contribuições na área de comunicação aumentativa e alternativa. Alguns mostraram melhorias nos componentes primários, outros trabalhos apresentaram aperfeiçoamentos nos componentes secundários e poucos se destacaram nos componentes terciários.

Um ambiente de comunicação adaptável, robusto e de baixo custo é foi apresentado por Loja et al. (2015). Esse ambiente contém quatro módulos: entrada de dados, comunicação, auxílio ao cuidador e interação externa. O programa é composto por componentes primários, secundários e terciários para possibilitar a comunicação dos usuários. A Figura 2.8 ilustra a interação do usuário com o ambiente de comunicação proposto e as suas funcionalidades.

Figura 2.8 – Arquitetura do sistema de CAA proposto por Loja et al. (2015)

Fonte: (LOJA et al., 2015)



Com o objetivo de aumentar a velocidade de comunicação do paciente com o meio ex-

terno, o **módulo de comunicação** mostrado na Figura 2.8 possui uma prancha de comunicação com ícones e palavras que representam as ações mais frequentes dos pacientes. Além disso, esse módulo deve possuir um teclado virtual, otimizado, compacto, flexível e adaptável. O objetivo desta tese é analisar, modelar, codificar e validar esse teclado que será acoplado ao módulo de comunicação desse ambiente.

Nesse teclado serão implementados algoritmos de predição de texto com a finalidade de minimizar o número de interações necessárias para escrever uma palavra e aumentar a performance de digitação. O teclado virtual será compacto, pois utilizará mais de uma letra por tecla. Assim, para escrever uma determinada palavra será minimizado o número de interações entre o sistema e o paciente. Finalmente, o *layout* do teclado será adaptado conforme a utilização e o vocabulário do usuário, permitindo que o *software* se adapte ao modo de escrita do paciente.

2.5 Considerações Finais

Este capítulo descreveu os conceitos de Tecnologia Assistiva e de Ambientes de Comunicação Aumentativa e Alternativa. No final deste capítulo foi mostrado o conceito de sistema de CAA e a proposta geral de um novo ambiente de Comunicação Alternativa e Aumentativa.

O próximo capítulo apresenta parte da revisão sistemática realizada com o objetivo de compreender as principais características do teclado virtual, os métodos de otimização e as métricas utilizadas para medir a performance e o esforço desse tipo de teclado. Finalmente, são descritos os trabalhos correlatos relacionados a teclados virtuais.

CAPÍTULO 3

TECLADOS VIRTUAIS

3.1 Introdução

Com a finalidade de construir um teclado virtual assistivo, otimizado, compacto e adaptativo foi realizada uma revisão sistemática para determinar: 1) quais são as principais características de um teclado virtual; 2) quais são os métodos e as técnicas utilizados para otimizar a performance de digitação desse teclado; e 3) quais são as métricas utilizadas para avaliar a performance de entrada de dados dos teclados virtuais. A resposta a essas três perguntas são essenciais para a modelagem e a construção do teclado virtual proposto neste trabalho, bem como a verificação de características de inovação e novas contribuições.

A primeira pergunta determina quais são as características essenciais para o funcionamento do teclado e como esses atributos podem variar. Os teclados virtuais assistivos apresentam um esforço de digitação alto e uma baixa performance de entrada de dados. Portanto, o objetivo da segunda pergunta é identificar quais os métodos e as técnicas de otimização são utilizados para reduzir o esforço de digitação e aumentar a performance de entrada de dados. Finalmente, a finalidade da terceira pergunta é identificar quais são as métricas que avaliam a performance de comunicação dos teclados.

Além de responder as três perguntas que fundamentam a base deste trabalho, este capítulo descreve os trabalhos correlatos relacionados a teclados virtuais.

Este capítulo descreve as principais características de um teclado virtual. São apresentados os métodos e as técnicas de otimização da performance de digitação e as métricas utilizadas para avaliar essa performance. Em seguida, são descritos os trabalhos correlatos. E finalmente, são realizadas as considerações finais deste capítulo.

3.2 Características do Teclado Virtual

Antes de implementar um *software* de teclado é necessário entender quais são os requisitos desse programa. Portanto, para desenvolver um teclado virtual é necessário identificar

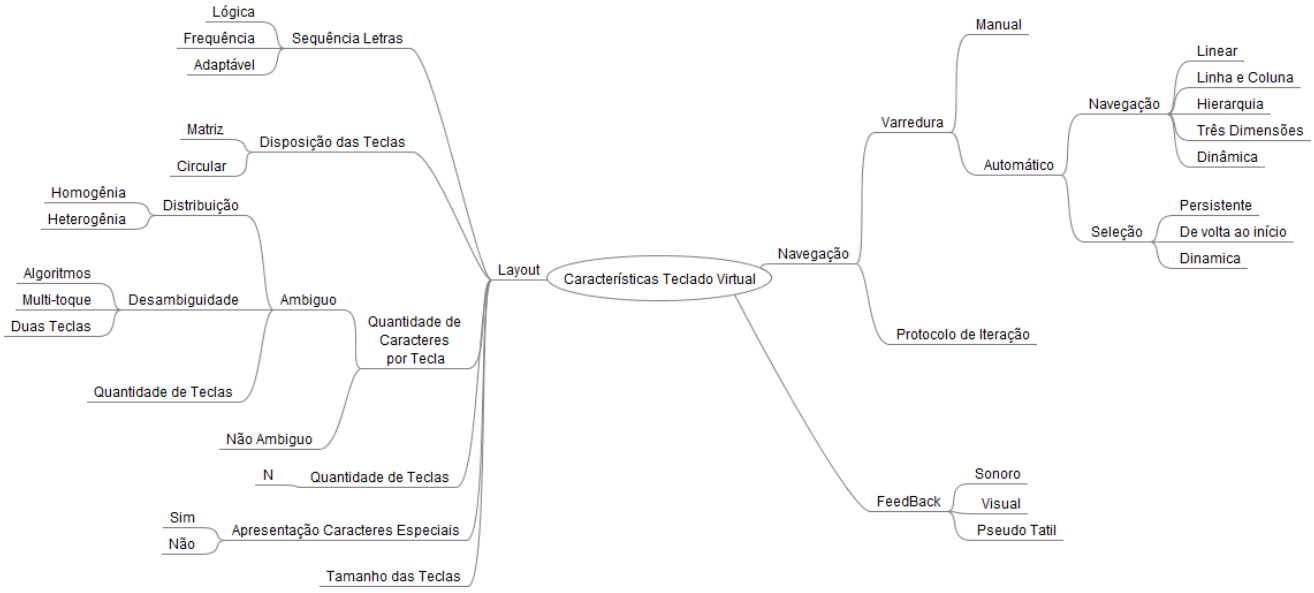
quais as características essenciais desse tipo de *software* e quais valores esses atributos podem assumir.

Com o objetivo de identificar essas características definiu-se o que é classificado como uma característica do teclado virtual. Assim, determinou-se o conceito de **característica do teclado virtual** como: qualquer atributo pertencente a um teclado virtual, essencial para a sua utilização e visualização que possa variar entre as abordagens de construção de cada teclado.

Partindo desse conceito foram identificadas oito características inerentes a todos os teclados virtuais. Essas características são: posicionamento e tamanho das teclas, quantidade de teclas, sequência das letras distribuídas entre as teclas, apresentação dos caracteres especiais, quantidade de caracteres apresentados em cada tecla, feedback do teclado e tipo de navegação.

A Figura 3.1 mostra as características pesquisadas e quais valores cada característica pode assumir.

Figura 3.1 – Mapa mental das características do teclado virtual identificadas durante a revisão sistemática



3.2.1 Posicionamento das teclas

O **posicionamento das teclas** é a característica que define como as teclas serão posicionadas visualmente no teclado. Existem duas maneiras de organizar as teclas: em forma matricial ou circular. A maioria dos trabalhos relacionados aos teclados virtuais posicionam as teclas em forma de matriz. Poucas pesquisas, posicionam as teclas de forma circular (PRABHU;

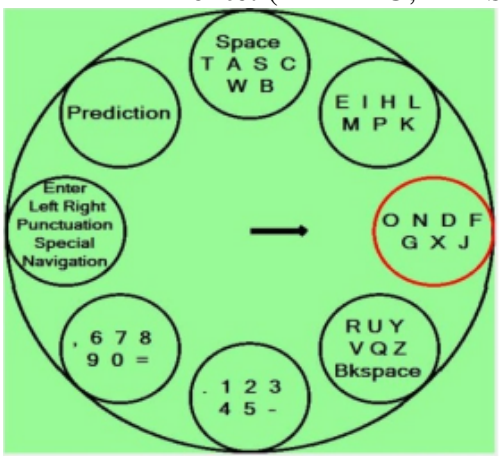
PRASAD, 2011; TOPAL; BENLIGIRAY, 2012). As Figuras 3.2 e 3.3 ilustram essas duas disposições de teclas, respectivamente.

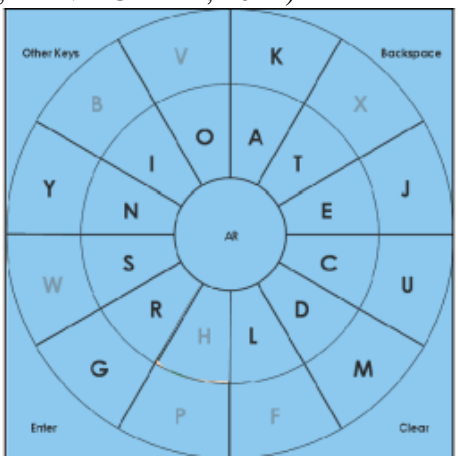
Figura 3.2 – Teclado virtual em forma de matriz. As teclas são dispostas em linhas e colunas.

A forma de distribuição matricial se assemelha mais a disposição de teclas de um teclado físico. A curva de aprendizado dos teclados virtuais que apresentam a distribuição de teclas matricial é menor devido a essa semelhança com os teclados físicos.

Figura 3.3 – Teclados virtuais em forma circular. As teclas ficam dispostas em forma circular.

Fonte: (PRABHU; PRASAD, 2011; TOPAL; BENLIGIRAY, 2012)





A forma circular é totalmente diferente da forma de apresentação das teclas dos teclados convencionais. Pesquisadores como (TOPAL; BENLIGIRAY, 2012), defendem que o posicionamento circular possibilita que as letras posicionem-se mais próximas umas das outras, aumentando a performance de digitação.

3.2.2 O tamanho das teclas

Além da geometria de apresentação do teclado virtual, vários trabalhos consideram relevante o tamanho das teclas (FARAJ; MOJAHID; VIGOUROUX, 2009b; FARAJ; VIGOUROUX, 2009; GELORMINI; BISHOP, 2013; GHOSH et al., 2010; NICOLAU et al., 2013;

VARCHOLIK; LAVIOLA; HUGHES, 2012; WU; HUANG; WU, 2014; GHOSH et al., 2010). O teclado virtual pode possuir diversas teclas de mesmo tamanho ou tamanhos variados.

Trabalhos como os de (FARAJ; MOJAHID; VIGOUROUX, 2009b; FARAJ; MOJAHID; VIGOUROUX, 2009a) variam o tamanho das teclas para diminuir a quantidade de erros inseridos pelos usuários e aumentar a performance de digitação. A metodologia utilizada por esses pesquisadores é expandir o tamanho das próximas teclas mais prováveis de serem selecionadas. Assim, diminui a possibilidade do usuário de pressionar uma tecla indesejada. Essa estratégia também reduz a distância entre as teclas e o tempo de procura pela letra desejada, aumentando a performance de digitação.

3.2.3 Distribuição das letras

A **sequência das letras** determina a ordem em que as letras são apresentadas nas teclas do teclado virtual. Essa distribuição ocorre de duas formas: por frequência ou de maneira lógica (JOSHI et al., 2011). Entretanto, o trabalho de (BHATTACHARYA; LAHA, 2012) adicionou mais uma forma de distribuição chamada de adaptativa.

A **distribuição das letras** fundamentada na frequência posiciona as letras tomando como base a frequência com que elas aparecem em um determinado corpus do idioma adotado pelo teclado virtual (JOSHI et al., 2011). O corpus de uma linguagem é uma coleção representativa de textos de diferentes tamanhos e estilos (BHATTACHARYA; LAHA, 2012). O objetivo desse apanhado de textos é representar a linguagem a qual pertence. A distribuição por frequência torna mais fácil o acesso a caracteres mais frequentes (BHATTACHARYA; LAHA, 2012).

Os arranjos das letras de maneira lógica fundamentam-se na lógica do idioma utilizado no teclado (JOSHI et al., 2011). A distribuição em ordem alfabética é um exemplo de distribuição lógica. Os teclados que adotam a distribuição das letras obedecendo a ordem alfabética apresentam as letras dispostas de “a” a “z” sequencialmente entre as teclas.

A distribuição adaptativa das letras altera a sequência das letras de acordo com o comportamento do usuário, portanto, a ordem das letras é dinâmica (BHATTACHARYA; LAHA, 2012). O objetivo é desenvolver um teclado que se adapte ao modo de escrita do usuário.

3.2.4 Quantidade de caracteres por tecla

A **quantidade de caracteres por tecla** é uma característica relevante na construção dos teclados virtuais. Teclados que possuem uma letra por tecla são classificados como teclados não ambíguos (MOLINA; RIVERA; GóMEZ, 2009). Os teclados que possuem mais de um

caractere por tecla são categorizados como teclados ambíguos (MOLINA; RIVERA; GóMEZ, 2009). Analisando os trabalhos pesquisados para esta tese verificou-se que a distribuição dos caracteres por tecla pode ser homogênea ou heterogênea. Distribuições homogêneas possuem a mesma quantidade de caracteres por tecla, enquanto os arranjos heterogêneos apresentam uma quantidade diferenciada de caracteres por tecla.

Outra característica dos teclados virtuais ambíguos é a variação na **quantidade de teclas**. Em teclados não ambíguos a quantidade de teclas é igual a quantidade de caracteres. Porém em teclados ambíguos o número de teclas pode variar. Miró e Bernabeu (2008) usaram apenas duas teclas para a inserção de texto. Outros trabalhos como os de (MIRó-BORRáS et al., 2009; TANAKA-ISHII; INUTSUKA; TAKEICHI, 2002a) utilizaram quatro teclas. O número de teclas pode variar, contanto que esse número seja menor ou igual ao número de caracteres contidos no alfabeto utilizado pelo teclado.

3.2.5 *Feedback* do teclado

O ***feedback* do teclado** é o modo como o teclado informa ao usuário que a tecla foi selecionada. Para usuários que utilizam o teclado físico, o *feedback* é tátil, ou seja, o usuário pode sentir as teclas sendo pressionadas por seus dedos. Entretanto, teclados virtuais não são capazes de fornecer este tipo de *feedback* de maneira natural. Para esses teclados é necessário desenvolver *feedbacks* alternativos. Algumas dessas alternativas de feedback são: sonoro, visual e pseudo tátil.

Os teclados virtuais fornecem *feedbacks* sonoros emitindo algum som, assim que uma tecla é selecionada. Os *feedbacks* visuais são realizados destacando a tecla selecionada das demais teclas do teclado virtual, expandindo a tecla escolhida ou marcando-a com uma cor diferente das demais. Finalmente, os *feedbacks* pseudo tátil podem ser realizados se o usuário estiver utilizando o dispositivo móvel. Esse *feedback* é fornecido quando o dispositivo vibra assim que uma tecla é pressionada.

3.2.6 Navegação

Para pessoas com necessidades especiais qualquer diminuição no esforço para escrever um texto é de grande ajuda. Além disso, há pacientes que são capazes de emitir apenas um tipo de sinal: ativo ou inativo. Para que esse tipo de usuário utilize o teclado virtual é essencial que o teclado possua um sistema de varredura ou um protocolo de interação (POLácEK; MíKOVEC; SLAVíK, 2012). Por esse motivo, essas duas técnicas são consideradas essenciais para a utilização do teclado virtual assistivo.

3.2.6.1 Protocolo de interação

O protocolo de interação permite desenvolver comandos específicos utilizando um determinado tipo de estímulo. Por exemplo, em um sistema que utiliza o piscar dos olhos do usuário como estímulo de entrada, o sistema poderia ser codificado para classificar determinados padrões de piscada. Assim, se o usuário piscasse brevemente os olhos o sistema poderia realizar uma ação específica. Porém, se o paciente fechasse os olhos durante um tempo maior o sistema poderia realizar outra ação. Essa abordagem utiliza como suporte o reconhecimento de padrões, no qual cada padrão corresponde a um comando específico.

A vantagem de utilizar o reconhecimento de padrões é permitir que um sinal adquirido possa gerar *ene*s comandos. Com uma maior variedade desses comandos, pode-se aumentar a performance de uso do mecanismo de entrada de dados (SUN; HU; WU, 2010). Porém, a concentração do usuário em realizar um estímulo específico é uma atividade cansativa, além disso o sistema computacional pode falhar na interpretação do sinal e codificá-lo como outro comando (DREWES, 2010).

Considerando os sistemas que utilizam o piscar dos olhos como entrada, a piscada pode ser classificada de acordo com a sua duração: *curta* ou *longa*. A codificação de Huffman foi utilizada para organizar os comandos conforme uma sequência de piscadas (KNUTH, 1985). Essa codificação utiliza uma combinação binária de no máximo n elementos para obter uma representação. Um exemplo é a representação alfabética do código de Morse. Essa representação usa um código de até cinco elementos para representar uma letra, um número ou um caractere especial.

Utilizando a codificação de Huffman é possivel estruturar os comandos de um sistema conforme uma sequência de piscadas. Uma sequência de piscadas curtas e longas podem ampliar o número de comandos do usuário sem o auxílio de dispositivos de entrada convencionais. Esses dispositivos podem ser o teclado e o mouse. As Figuras 3.4 a 3.6 ilustram as possíveis opções de árvores conforme o tamanho do código a ser interpretado. Nessas figuras e letra P significa uma piscada curta, enquanto a letra L representa uma piscada longa.

Figura 3.4 – Possíveis opções para as palavras de tamanho um utilizando a codificação de Huffman.

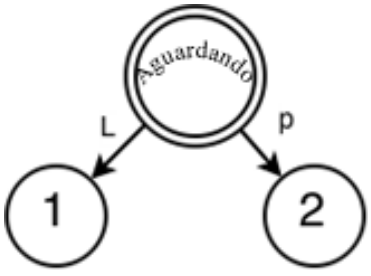
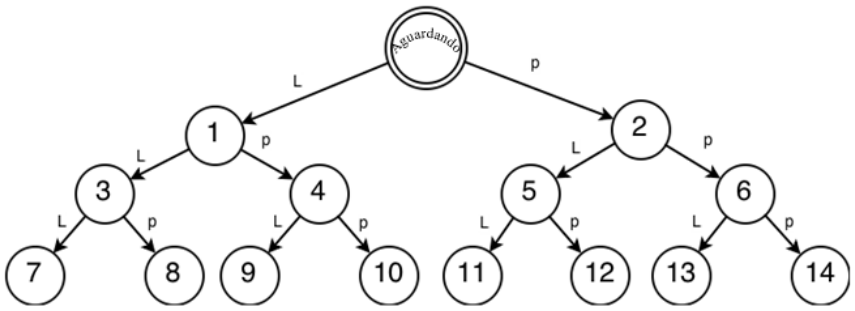
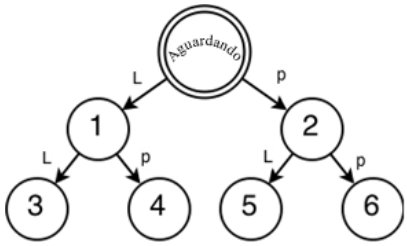


Figura 3.6 – Possíveis opções para as palavras de tamanho três utilizando a codificação de Huffman.



Pode-se observar nas Figuras 3.4 a 3.6 que a possibilidade de opções aumenta exponencialmente em relação ao tamanho da código. Além disso, quanto maior é o número de elementos do código, mais tempo é necessário para acionar um comando. Para acionar o comando de número oito ilustrado na Figura 3.6 é necessário duas piscadas longas e uma curta mais o tempo de inatividade. Uma piscada curta leva de 200 a 500 milissegundos (ms) e uma piscada longa pode demorar de 500 a 1000 ms (KROLAK; STRUMILLO, 2008). Assim, o tempo total para o acionamento desse comando é de no mínimo 1.700 ms.

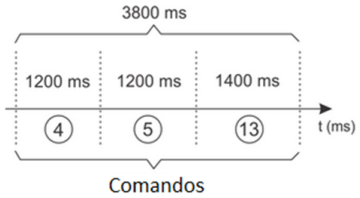
Figura 3.5 – Possíveis opções para as palavras de tamanho dois utilizando a codificação de Huffman



A Figura 3.7 mostra o tempo de execução dos comandos 4, 5 e 13 da Figura 3.6. O tempo total para a execução dessa sequência de comandos é de no mínimo 3.800 ms.

A subseção 3.2.6.2 apresenta os métodos de varredura, que podem ser utilizados como uma alternativa para o mecanismo de navegação. Diferente do protocolo de interação, a navegação por varredura é realizada de forma automática. Portanto, o usuário precisa apenas confirmar a seleção dos elementos da tela no momento certo.

Figura 3.7 – Tempo de execução dos comandos 4, 5 e 13 da Figura 3.6.

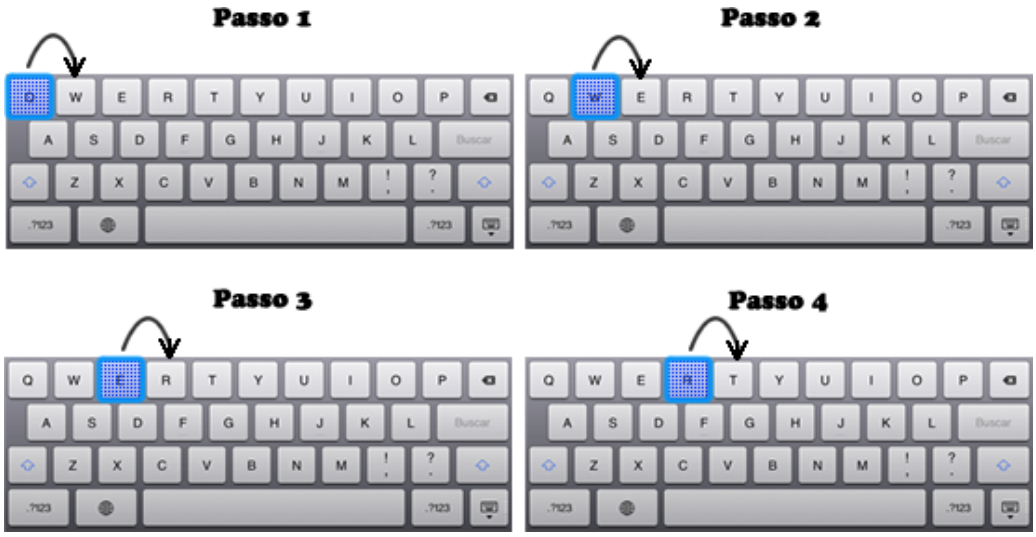


3.2.6.2 Métodos de varredura

Foram identificados quatro **métodos de varredura** (POLácEK; MíKOVEC; SLAVíK, 2012): linear, linha e coluna, três dimensões, hierarquia e dinâmico. O *software* de varredura linear move o foco do cursor entre os elementos do teclado item a item, sucessivamente. A cada alternância do foco, o elemento em que o cursor está fica destacado por um determinado período de tempo. O usuário seleciona o elemento emitindo um sinal, assim que a tecla desejada ficar destacada das demais teclas (SCHADLE, 2004).

A Figura 3.8 ilustra a varredura linear. Nessa figura o foco do cursor move-se entre as teclas “q”, “w”, “e” e “r” sequencialmente. Se o usuário não selecionasse a letra “r” o cursor deslocaria para letra “t” e assim por diante.

Figura 3.8 – Método de varredura linear em quatro passos. O método de varredura passa pelas teclas sequencialmente até que seja selecionada uma delas.



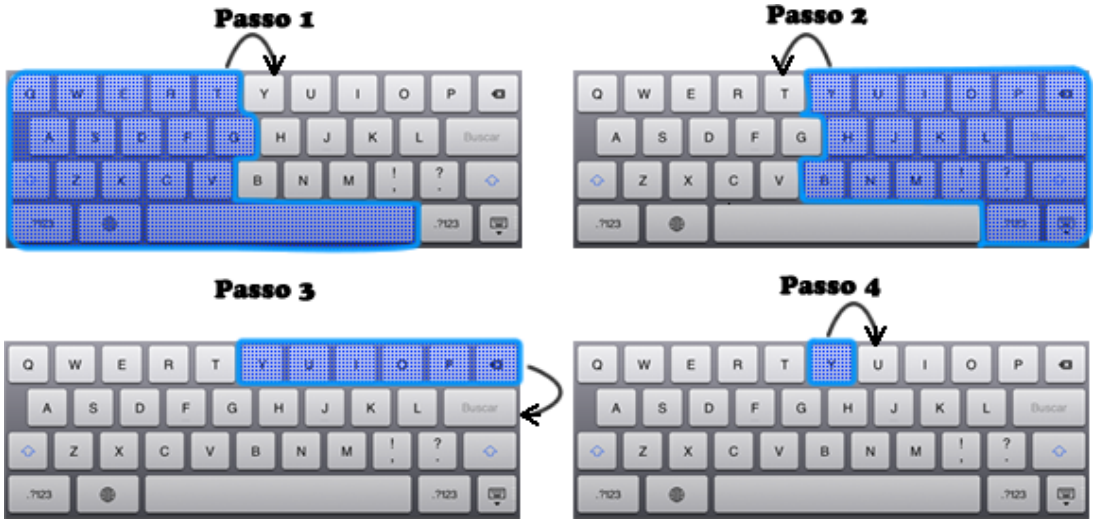
O *software* de varredura linha e coluna move o foco do cursor entre os grupos alinhados de elementos do teclado virtual. A seleção da tecla desejada é realizada em dois passos. O primeiro consiste em selecionar o grupo de elementos desejado e o segundo passo é realizado pela varredura linear (SCHADLE, 2004). A Figura 3.9 mostra a varredura linha e coluna.

Nos passos um e dois o foco do cursor está se movendo entre as linhas do teclado. No passo dois o usuário emite o sinal para selecionar a linha. No passo três inicia o processo de varredura linear começando na letra “a”.

A varredura em três dimensões ou por bloco é realizada dividindo o teclado em *n* seções distintas (FELZER; RINDERKNECHT, 2009). O *software* de varredura destaca cada quadrante do teclado, sucessivamente. Assim que o usuário emite um sinal, o quadrante em destaque é selecionado. Após a seleção é utilizado o método de varredura linha e coluna. A Figura 3.10 mostra a varredura em três dimensões.

Figura 3.9 – Método de varredura linha e coluna em 4 passos. Esse método destaca os grupos de linhas até que o usuário escolha a linha na qual a letra está. Após a primeira seleção é executado o método de seleção linear.

Figura 3.10 – Método de varredura três dimensões em 4 passos. Esse método destaca blocos de caracteres. Assim que o usuário seleciona um bloco, o sistema executa o método de varredura linha e coluna.



Nesse exemplo, o teclado é dividido em dois blocos. Nos passos um e dois, os blocos recebem o foco do cursor alternadamente. No passo dois, o usuário seleciona o segundo bloco para iniciar a varredura de linha. No passo três, o usuário escolhe a primeira linha para iniciar a varredura de linha e coluna. Finalmente, no passo quatro. o usuário pode escolher entre as teclas individualmente.

O método *containment hierarchy* agrupa as teclas em um grafo em forma de árvore (BALJKO; TAM, 2006). Cada nó do grafo é associado a um grupo de teclas e os nós folhas representam apenas uma tecla. O método de varredura destaca as teclas que estão no nó superior. A cada seleção, o usuário realiza a operação de *drill-down* no grafo de árvore, até chegar a uma folha e escolher uma letra. A árvore de letras pode ser estruturada de várias formas, uma delas é

utilizando unigram. As Figuras 3.11 e 3.12 ilustram o grafo de navegação balanceado utilizando o unigram e a varredura em hierarquia usando esse grafo, respectivamente.

Figura 3.11 – Grafo do método de varredura *containment hierarchy* em forma de árvore.

Fonte: (BALJKO; TAM, 2006)

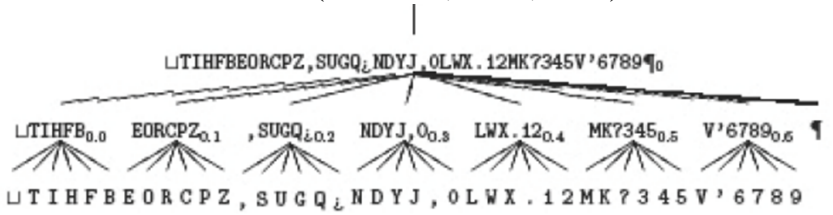
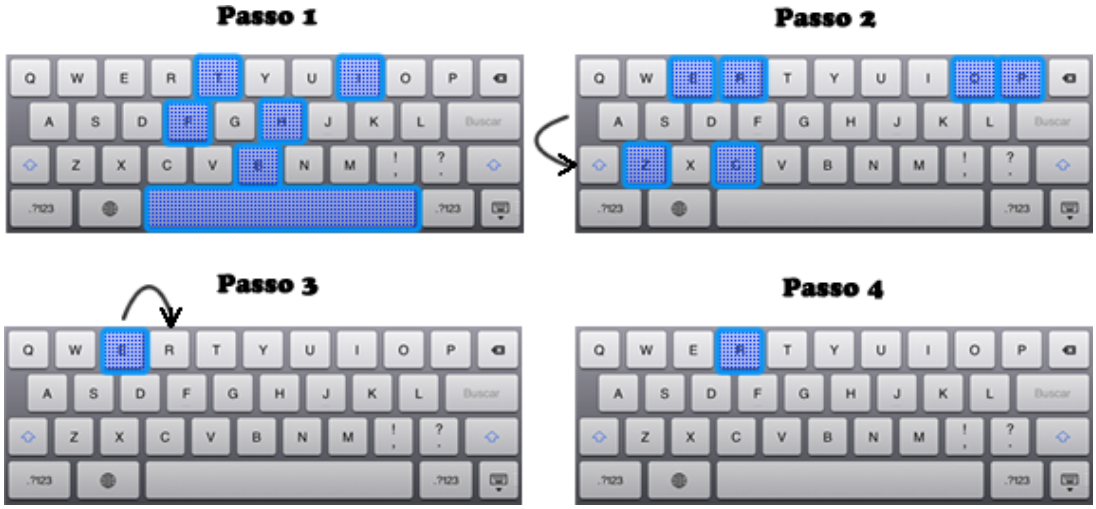


Figura 3.12 – Método de varredura *containment hierarchy* em 4 passos. Esse método é iniciado destacando o grupo de caracteres. Assim que um grupo é selecionado o sistema inicia o método de varredura linear.



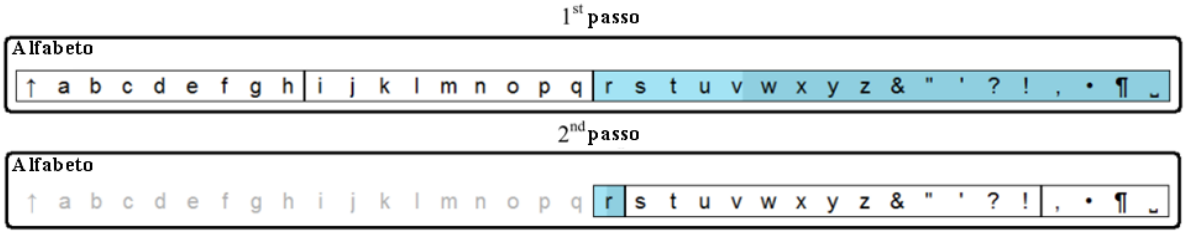
Além desses quatro métodos de varredura, Poláček, Míkovec e Slavík (2012) propuseram um método de varredura dinâmico. O processo de varredura no teclado virtual, proposto por eles utiliza uma generalização do algoritmo de busca binária denominado *N-ary search*.

Para utilizar esse algoritmo Polacek et al. ajustaram os caracteres em ordem alfabética e posicionaram as teclas em uma matriz de uma única linha. O processo de varredura é iniciado dividindo os caracteres em N grupos. Assim que um grupo é selecionado, esse grupo de símbolos é dividido em N grupos novamente. Essa divisão é realizada até que um único caractere seja selecionado. A Figura 3.13 mostra um exemplo de varredura do algoritmo de Polacek.

O método de varredura também pode ser caracterizado como automático ou manual (MOLINA et al., 2009). No método automático, a varredura é realizada pelo próprio sistema, o usuário deve selecionar as teclas, enquanto o *software* realiza a navegação. No método manual, o usuário pode interagir com o método de varredura mudando sua direção, interrompendo seu andamento ou retrocedendo sua seleção.

Figura 3.13 – Método de varredura proposto por Poláček, Míkovec e Slavík (2012). É realizada uma varredura binária entre os caracteres. Assim, o sistema destaca o bloco desejado, o usuário seleciona o bloco no qual está o caractere, até que o sistema destaque o caractere desejado.

Fonte: (POLácEK; MíKOVEC; SLAVíK, 2012)



Outra característica variável do método de varredura é a ação após a seleção de uma determinada tecla. Esse atributo pode ser definido como *snapt-to-home* ou persistente (MILLET; ASFOUR; LEWIS, 2009). Métodos de varredura *snapt-to-home* retrocedem a seleção das teclas à primeira tecla do teclado, assim que é realizada uma seleção. Métodos persistentes continuam a varredura a partir da tecla selecionada.

Outra característica importante do método de varredura é o tempo de parada em cada elemento. Esse tempo é conhecido como *scanning delay* e está vinculado diretamente a performance de digitação dos teclados que utilizam o método de varredura (MIRó-BORRáS et al., 2009). Se a varredura é realizada rapidamente, ou seja, se o *scanning delay* é baixo, naturalmente o número de palavras digitadas pode ser maximizado (MOLINA; RIVERA; GóMEZ, 2009).

3.2.7 Caracteres especiais

Com o objetivo de aumentar a performance de digitação, os teclados virtuais podem ou não conter caracteres especiais. Alguns trabalhos retiram caracteres especiais como pontuação, navegação e formatação permitindo que o tempo de varredura diminua.

Ocultar os caracteres especiais é uma técnica válida para aumentar a performance de digitação. Porém, em situações em que o usuário deseja escrever um texto acadêmico ou um email formal, esse método pode prejudicar a comunicação.

3.3 Técnicas de Otimização

Vários métodos foram pesquisados para otimizar a performance de entrada de dados e minimizar as interações do usuário com o teclado virtual. Os principais métodos identificados

foram: algoritmos de predição de letras e palavras, otimização da disposição das letras e teclas, métodos de desambiguidade e adaptação do teclado ao usuário.

3.3.1 Predição

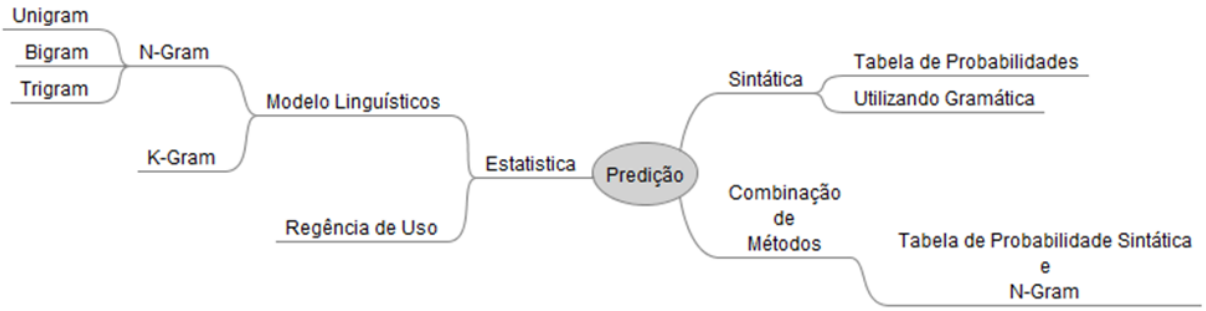
Os métodos de predição podem ser divididos em dois tipos: predições estatísticas e sintáticas (SHARMA et al., 2010; SHARMA et al., 2012). A predição estatística é fundamentada nos modelos de linguagem como n-gram. Nesse tipo de predição uma lista de sugestões de termos é gerada baseada na frequência de um termo e/ou na palavra utilizada mais recentemente (SHARMA et al., 2012).

A finalidade da predição sintática é garantir que o sistema não apresente uma palavra gramaticalmente inapropriada para o usuário (SHARMA et al., 2010). Esse tipo de predição utiliza a sintaxe do texto para prever quais serão os próximos termos a serem digitados (GARAY-VITORIA; ABASCAL, 2005).

A predição sintática apresenta duas vantagens (SHARMA et al., 2010). Primeiro, como o sistema considera a sintaxe da sentença para sugerir as palavras, os termos sugeridos são sintaticamente corretos. Outra vantagem é que o número de combinações dos termos sintáticos é menor do que o número de combinações entre as palavras. Portanto, a memória computacional para guardar a sequência das categorias sintáticas é menor (GARAY-VITORIA; ABASCAL, 2005).

Os métodos de predição de letras utilizam apenas a predição estatística, enquanto que o método de predição de palavras usa os dois tipos de predição. A Figura 3.14 ilustra os tipos de predição encontrados na literatura.

Figura 3.14 – Tipos de predição identificados durante a revisão sistemática. Eles se dividem em dois grupos: predição estatística e predição sintática. Além disso, esses grupos podem ser combinados.



3.3.1.1 Predição de letras

Os **algoritmos de predição de letras** auxiliam o usuário sugerindo os caracteres mais prováveis de ocorrerem após a escolha de uma determinada sequência de letras. Foram encontrados na literatura dois modelos de linguagem que constituem a base dos algoritmos de predição estatísticos. Esses modelos são: n-grams e k-grams.

3.3.1.1.1 Modelo linguístico n-gram

A probabilidade de um termo especifico aparecer após uma sequência de termos está diretamente relacionada aos termos anteriores dessa sequência (SHANNON, 1951). O n-gram busca sugerir o termo seguinte de uma determinada sequência de termos baseando-se nos n termos digitados anteriormente.

O número de termos usados como base de cálculo para a sugestão pode variar. Os algoritmos que utilizam apenas um termo para predição são denominados unigram $(n = 1)$. O objetivo dessa predição é completar a palavra que está sendo digitada. Os algoritmos que utilizam mais de um termo como o bigram $(n = 2)$ e o trigram $(n = 3)$ são usadas para sugerir um termo antes mesmo do início da digitação. Apesar de ser tradicionalmente limitado a quatro, o número de termos pode variar indefinidamente (LESHER; MOULTON; HIGGINBOTHAM, 1998). O aumento do número de termos torna a predição mais exata.

O modelo linguístico n-gram é amplamente utilizado em processamento de linguagem (JANPINIJRUT; NATTEE; KAYASITH, 2011) e pode ser usado tanto para predição de palavras ou letras (GOODMAN et al., 2002). A Equação (3.1) mostra o cálculo realizado no n-gram.

$$P(c_i) = P(c_i|c_{i-n}, \ldots, c_{i-1}) \tag{3.1}$$

em que: $P(c_i)$ é a probabilidade de ocorrência de um caractere c, i representa a posição do caractere no texto e c_i é o caractere na posição i.

A probabilidade de um termo aparecer após o n-gram é calculada a partir da análise do corpus da linguagem desejada. Esse cálculo constrói uma tabela de probabilidade, similar a Tabela 3.1. Essa tabela é utilizada pelo algoritmo de predição que analisa os termos anteriores e apresenta o termo que possui a maior probabilidade de ocorrer após uma determinada sequência de termos.

Considerando a Tabela 3.1 se o usuário está inserindo a sequência de letras “lui”, a probabilidade da letra “z” ser a próxima letra da sequência é 40%. Enquanto a probabilidade das próximas letras serem “s” ou “g” é de 40% e 30%, respectivamente.

Tabela 3.1 – Probabilidades do método n-gram

Probabilidade	**Próximo caractere**	**Caracteres**
40%	z	lui
40%	s	lui
30%	g	lui

3.3.1.1.2 Modelo linguístico k-gram

Outro modelo linguístico utilizado para predição de texto é o k-gram. Segundo (MIRÓ-BORRáS et al., 2009) n-grams e k-grams são modelos similares. A diferença entre eles é que, a predição de caracteres que utiliza o modelo k-gram sugere os próximos termos fundamentado nos caracteres da palavra que está sendo inserida. Para realizar a sugestão, o modelo n-gram considera toda a sentença anterior. Portanto, no n-gram os caracteres utilizados para a predição podem começar em qualquer parte da palavra ou até mesmo pertencer a outra palavra.

Além da previsão do caractere, o k-gram pode ser usado para a previsão das palavras (LESHER; MOULTON; HIGGINBOTHAM, 1998). Como esse modelo linguístico considera somente a palavra que está sendo inserida, esse método pode sugerir o final de uma palavra a partir dos $k-1$ caracteres digitados. A probabilidade da palavra prevista por esse método é obtida pela frequência de ocorrência dessas palavras no corpus fonte (LESHER; MOULTON; HIGGINBOTHAM, 1998).

3.3.1.2 Predição de palavras

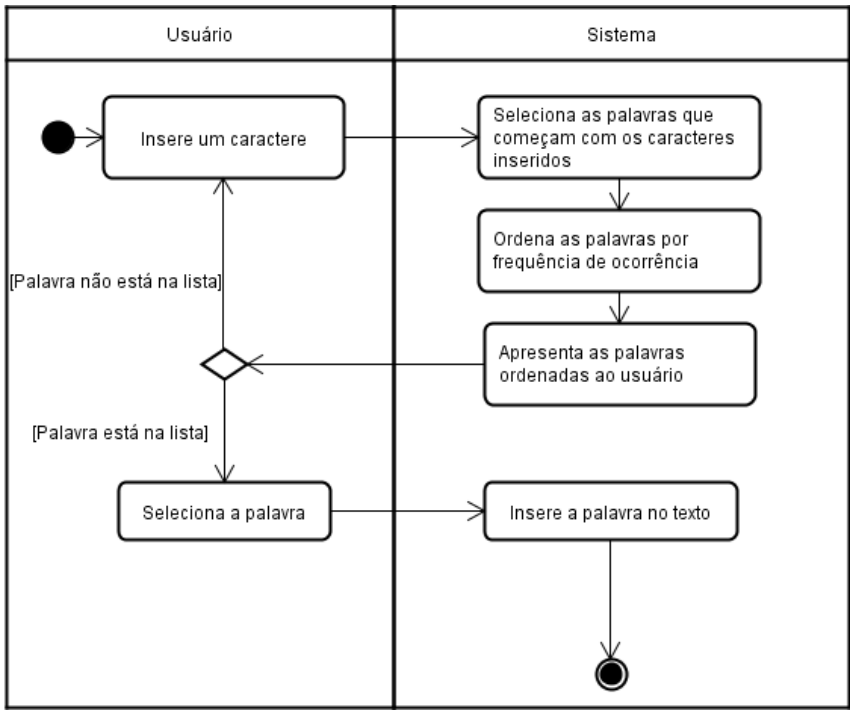
A predição de palavras pode ser utilizada para completar palavras (TRUONG, 2013) ou para sugerir uma palavra durante a escrita/digitação. O método de predição é um sistema que busca antecipar o próximo bloco de caracteres, letras, sílabas, palavras, sentenças, que o usuário deseja expressar (GARAY-VITORIA; ABASCAL, 2005).

Os métodos de predição sugerem um conjunto de termos com alta probabilidade de ocorrência a partir da sequência inicial de palavras ou caracteres digitados. Os métodos de predição são aplicados comumente a *softwares* de comunicação com o objetivo de aumentar a performance de digitação (GARAY-VITORIA; ABASCAL, 2005). Nas pesquisas realizadas neste trabalho foram encontrados cinco métodos de predição de palavras, sendo eles: predição baseada na frequência de ocorrência, regência de uso, tabela de probabilidade de palavras, n-gram e tabela de probabilidade sintática.

3.3.1.2.1 Frequência de ocorrência de uma única palavra

O método de freqüência de uma única palavra calcula o número de vezes que uma palavra aparece no corpus da linguagem (GARAY-VITORIA; ABASCAL, 2005). Portanto, cada palavra é vinculada a sua frequência de ocorrência. A partir dessa lista de palavras e frequência é construída uma lista de sugestões de palavras mais prováveis. O diagrama da Figura 3.15 ilustra o processo de sugestão de palavras utilizando a frequência única.

Figura 3.15 – Processo de predição de uma única palavra.



Assim que o usuário inicia o processo de digitação, o sistema seleciona quais as palavras que começam com os caracteres inseridos que possuem maior frequência de ocorrência. Logo, o sistema de predição sugere as palavras selecionadas ao usuário. O usuário pode aceitar a sugestão ou continuar digitando. Se ele continuar digitando o sistema propõe novas palavras e as apresenta como sugestão. Esse processo continua até que o usuário selecione uma palavra ou termine de digitar.

3.3.1.2.2 Regência de uso

O objetivo principal do método de regência de uso é que as palavras que ocorreram recentemente possuem maior probabilidade de serem digitadas novamente (WANDMACHER et al., 2008). Esse método pode ser utilizado em conjunto com a predição de frequência de uma única palavra (GARAY-VITORIA; ABASCAL, 2005). Portanto, o método de regência de uso especifica o quão recentemente uma palavra foi usada. Se um determinado termo foi utilizado

recentemente, essa palavra tem prioridade sobre as outras (SHARMA et al., 2010). Esse tipo de predição prioriza as palavras que estão sendo usadas pelo usuário (SHARMA et al., 2010).

Para que esse método funcione é necessário adicionar ao sistema uma variável para quantificar a regência de uso. Assim que o usuário digita uma palavra completa ou seleciona uma palavra a partir da lista de sugestões, a variável relativa a regência de uso da palavra inserida é acrescida de um. Dependendo da posição do elemento no texto, essa variável é ponderada por uma fator de deterioração (CLARKSON; ROBINSON, 1997).

O conceito do fator de deterioração determina que a probabilidade de recorrência de uma palavra depende da distância entre a palavra utilizada recentemente e a palavra a ser predita (WANDMACHER et al., 2008). A maior probabilidade de reutilização da palavra ocorre de 15 a 20 palavras depois do termo escrito. Após esse número de palavras a probabilidade de reutilização tende a cair.

3.3.1.2.3 Tabelas de probabilidades

A tabela de probabilidades de palavras é uma lista de duas ou mais palavras e da probabilidade desses termos ocorrerem em sequência (GARAY-VITORIA; ABASCAL, 2005). Para n maior do que um, esse método pode ser calculado como um n-gram. Assim que o usuário insere uma palavra o sistema busca quais os termos que possuem a maior probabilidade de ocorrência em sequência com as n palavras anteriores e constrói uma lista de sugestões com esses termos.

Essa estratégia de predição oferece ao usuário uma lista de sugestões de palavras antes mesmo do usuário inserir o primeiro caractere da palavra (GARAY-VITORIA; ABASCAL, 2005). Os bigrams consideram uma palavra anteriormente escrita para predizer a próxima palavra. A Equação (3.2) calcula a probabilidade do bigram.

$$P(w_i) = P(w_i|w_{i-1}) \tag{3.2}$$

em que: w_i é a palavra predita e w_{i-1} representa a palavra anterior.

Os trigrams calculam as duas palavras anteriores para predizer a próxima. A Equação (3.3) mostra o cálculo do trigram.

$$P(w_i) = P(w_i|w_{i-1}, w_{i-2}) \tag{3.3}$$

em que: w_i é a palavra a ser predita e, w_{i-1} e w_{i-2} são as palavras anteriores.

O trigram é um dos métodos de predição estatística mais utilizados (SHARMA et al., 2010). Wandmacher et al. (2008) afirmaram que quanto maior é o número de termos conside-

rados anteriormente a palavra que está sendo escrita maior é a exatidão do método de predição. Porém, Sharma et al. (2010), Garay-Vitoria e Abascal (2005) ressaltaram que o aumento do número de palavras pode ocasionar em um acréscimo considerável no uso de processamento e memória computacional.

3.3.1.2.4 Predição sintática utilizando a tabela de probabilidades

A predição sintática utilizando a tabela de probabilidades é realizada usando uma tabela de probabilidades que possui dois tipos de dados estatísticos. O primeiro elemento é a probabilidade de ocorrência de cada palavra. O segundo termo da tabela é a probabilidade relativa de ocorrência de todas as categorias sintáticas após cada categoria sintática (GARAY-VITORIA; ABASCAL, 2005). Quando o usuário insere uma palavra, o sistema calcula o próximo termo utilizando a probabilidade de ocorrência da próxima palavra relacionada a sua categoria sintática.

O trabalho de Sharma et al. (2010) ressaltou que o método de predição sintática usando a tabela de probabilidades pode ser realizado com mais de uma categoria sintática. Assim, nessa tabela podem existir duas ou mais categorias sintáticas e as palavras mais usadas após essas categorias. Fazly (2002) apresentou três maneiras de realizar a predição sintática. Essas maneiras são: utilizando somente as categorias sintáticas, combinando com a probabilidade da última palavra e as categorias sintáticas e realizando a combinação linear dos modelos de predição.

Com o objetivo de prever a próxima palavra usando somente as categorias sintáticas, Fazly (2002) armazenou duas categorias e a probabilidade da próxima palavra após essas duas categorias. Assim, o sistema analisa as categorias sintáticas das duas últimas palavras e sugere o próximo termo. A Equação (3.4) calcula a probabilidade de prever a próxima palavra baseada nas duas últimas categorias sintáticas.

$$P(w_i) = P(w_{ii}|t_{i-1}, t_{i-2}) \tag{3.4}$$

em que: w_i é a palavra a ser predita e, t_{i-1} e t_{i-2} são as categorias sintáticas das palavras anteriores.

A segunda maneira proposta por Fazly (2002) foi predizer a próxima palavra a partir da probabilidade de ocorrência dessa palavra precedida do termo anterior e da probabilidade das duas categorias sintáticas das palavras anteriores. A Equação (3.5) apresenta o cálculo dessa probabilidade.

$$P(w_i) = P(w_i|w_{i-1}, t_{i-1}, t_{i-2}) \tag{3.5}$$

em que: w_i é a palavra a ser predita, w_{i-1} representa a palavra anterior e, t_{i-1} e t_{i-2} são as categorias sintáticas dos termos anteriores.

A terceira maneira de realizar a predição sintática é pela combinação linear dos modelos (FAZLY, 2002). Primeiramente é calculada a probabilidade da categoria sintática da próxima palavra de acordo com as duas categorias anteriores. A partir desse cálculo são propostas as palavras que pertencem a categoria sintática selecionada. O resultado desse cálculo é combinado com a probabilidade de ocorrência da próxima palavra com base nas outras categorias. Neste caso, Fazly (2002) ultilizou o modelo de linguagem bigramo. Fazly usou essas duas predições balanceadas por meio de uma variável α que define o peso de cada probabilidade. A Equação (3.6) mostra esse cálculo.

$$P(w_i) = \alpha \times P(w_i|w_{i-1}) + (1-\alpha) \times [P(w_i|t_{cw}) \times P(t_{cw}|t_{pw}, t_{ppw})] \tag{3.6}$$

em que: w_i é a palavra a ser predita e w_{i-1} representa a palavra anterior.

3.3.1.2.5 Predição sintática utilizando gramática

De acordo com Garay-Vitoria e Abascal (2005), a predição sintática que utiliza gramática analisa as sentenças pelas regras gramaticais e pelo processamento da linguagem natural. A partir dessa análise são sugeridas as palavras que mais se adequam a sentença que está sendo digitada. Esse tipo de predição tem uma complexidade maior do que a predição sintática que usa as tabelas. Isso acontece porque o sistema de predição precisa analisar a sentença inteira antes de propor a próxima palavra (GARAY-VITORIA; ABASCAL, 2005).

3.3.1.3 Dicionário ou léxico

O dicionário ou léxico é uma unidade de armazenamento que é gerada depois do processamento do corpus da linguagem (SHARMA et al., 2010). Esse componente do teclado virtual é responsável pelo armazenamento das palavras, das categorias sintáticas, da frequência das palavras e das suas categorias, do domínio da conversa e de outros dados relativos as palavras. Algumas características desse componente são: organização, tamanho, número de dicionários e adaptação do dicionário.

3.3.1.3.1 Organização

A estrutura do dicionário é a forma como ele armazena as informações. O dicionário pode ser organizado de forma simples como uma lista de itens, palavras e suas informações ou

de forma mais complexa como a de uma árvore (SHARMA et al., 2010; GARAY-VITORIA; ABASCAL, 2005).

A forma de armazenamento de lista pode ser ordenada de maneira ascendente ou descendente de acordo com a ordem alfabética ou a frequência de ocorrência (GARAY-VITORIA; ABASCAL, 2005). A vantagem da estrutura de lista é a facilidade da alteração dos dados e a desvantagem é a demora no processamento computacional.

A estrutura de armazenamento em árvore torna a consulta aos dados menos onerosa em relação ao processamento computacional. Porém, esse tipo de estrutura torna mais difícil a atualização do dicionário. A inserção de palavras ou a alteração delas pode ocasionar restruturações inteiras no dicionário (GARAY-VITORIA; ABASCAL, 2005).

3.3.1.3.2 Tamanho do dicionário

O tamanho do dicionário é outra característica que deve ser considerada ao construir esse componente. Essa características mede a quantidade de informações armazenadas em um léxico. Dicionários que realizam a predição de palavras utilizando mais de um termo precedente possuem tamanhos maiores (GARAY-VITORIA; ABASCAL, 2005). Dependendo do tipo de predição o léxico deve armazenar mais ou menos informações.

Por exemplo, se o dicionário utilizar a predição sintática, o léxico deve armazenar as categorias sintáticas das palavras e as suas probabilidades de ocorrência. Por outro lado, se o sistema de predição usar as probabilidades estatísticas, então é necessário guardar as probabilidades de ocorrência calculadas com os n-grams e os k-grams.

Uma maneira de reduzir o tamanho dos dicionários é armazenar somente as raízes das palavras (SHARMA et al., 2010). A raiz pode ser combinada com diversos prefixos e sufixos para formar outros termos. A combinação de termos morfológicos é uma abordagem adequada quando a predição de palavras é aplicada a linguagens que são flexionadas como, por exemplo, a língua portuguesa (GARAY-VITORIA; ABASCAL, 2005).

3.3.1.3.3 Número de dicionários

O número de dicionários define quantos léxicos são utilizados no sistema de predição. Pode ser utilizado apenas um único dicionário ou a informação pode ser distribuída entre vários léxicos. A distribuição das informações entre os vários dicionários torna a consulta mais rápida, porém aumenta a complexidade do sistema (GARAY-VITORIA; ABASCAL, 2005).

Utilizar pequenos dicionários baseados no domínio do assunto que está sendo escrito possibilita melhores resultados em comparação o uso de dicionários grandes (SHARMA et

al., 2010). Porém, em termos de predição sintática, os dicionários maiores apresentam melhor relação custo benefício.

3.3.1.3.4 Adaptação do dicionário

A característica de adaptação determina como o dicionário se ajusta ao modo como o usuário escreve. Inicialmente o léxico pode apresentar um conjunto de dados com diversas palavras e suas frequências de uso. Esses dados podem ser extraídos do corpus de uma linguagem ou de vários textos escritos pelo usuário. A medida que o usuário utiliza o sistema o dicionário deve se ajustar ao seu modo de escrita.

A adaptação do dicionário é realizada pela entrada de novas palavras que antes não existiam no léxico. Além disso, o sistema deve alterar a frequência de uso das palavras de acordo com o vocabulário do usuário. Isso pode ser realizado aumentando a frequência das palavras que são geralmente usadas pelo usuário e diminuindo a frequência dos termos que não fazem parte do seu vocabulário. A satisfação do usuário com o sistema de predição tende a aumentar quando são sugeridas as palavras que fazem parte do seu vocabulário (GARAY-VITORIA; ABASCAL, 2005).

3.3.2 Organização e dimensionamento das teclas

Em um teclado virtual eficiente, as teclas e as letras devem ser organizadas de maneira a maximizar a performance de digitação e minimizar o movimento (GHOSH, 2011). Com a finalidade de alcançar esse objetivo as letras e as teclas devem ser reorganizadas de maneira ótima. Para desenvolver um *layout* otimizado foram encontrados seis métodos: lei de Fitts (FITTS, 1954), *Fitts digraph* (MACKENZIE; ZHANG, 1999), frequência de um único caractere, frequência de dígrafos, utilização de n-gram e algoritmos evolucionários.

3.3.2.1 Lei de Fitts

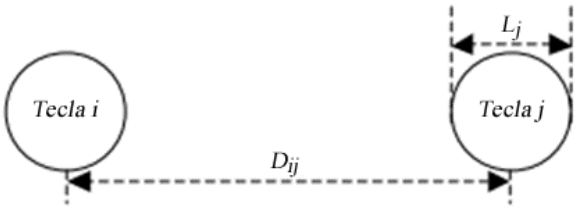
A lei de Fitts é um modelo de movimentação humana que determina o tempo necessário para o deslocamento de algo a partir de uma posição inicial até um objeto final em função da distância e do tamanho desse objeto. Esse modelo pode ser usado para otimizar a interação homem computador (IHC). Esse modelo afere o tempo de movimentação considerando a distância e o tamanho das teclas. A Equação (3.7) representa a lei de Fitts.

$$TM = a + b\log_2\left(\frac{D_{ij}}{L_j + 1}\right) \tag{3.7}$$

em que: D_{ij} é a distância Euclidiana entre as teclas i e j, L_j é a largura da tecla j, e a e b são coeficientes empíricos.

A Figura 3.16 ilustra as variáveis D_{ij} e L_j.

Figura 3.16 – Iustração da Lei de Fitts na qual a distância e o tamanho das teclas são calculados para determinar o tempo de movimentação.



Com a finalidade de otimizar a performance de digitação, alguns trabalhos utilizam o modelo de Fitts para calcular o tamanho e a distância entre as teclas. O trabalho de Faraj, Mojahid e Vigouroux (2009b) usa a lei de Fitts para otimizar o teclado de dispositivos móveis. Nessa pesquisa o teclado proposto pelos autores expande as próximas teclas mais prováveis. Pois, de acordo com a lei de Fitts, quanto maior é o tamanho da tecla a ser selecionada, menor é o tempo necessário para pressioná-la (FARAJ; MOJAHID; VIGOUROUX, 2009b). A Figura 3.17 mostra o teclado proposto por eles.

Figura 3.17 – Teclado proposto por Faraj, Mojahid e Vigouroux (2009b) no qual as teclas são aumentadas de tamanho para facilitar sua seleção.

Fonte: (FARAJ; MOJAHID; VIGOUROUX, 2009b)

Faraj, Mojahid e Vigouroux (2009a) propuseram outro *layout* de teclado virtual adicionando uma terceira dimensão a ele. Novamente, utilizando a lei de Fitts, Faraj, Mojahid e Vigouroux (2009a) apresentaram um *layout* que expande e contrai. Dois fatores realizam a expansão e a contração desse *layout*. Esses fatores são: o movimento de seleção do usuário e a

predição dos próximos caracteres mais prováveis. A Figura 3.18 ilustra o *layout* proposto por em seus estudos.

Figura 3.18 – Teclado proposto por Faraj, Mojahid e Vigouroux (2009a) na qual o teclado possui a forma de uma sanfona. Assim que o usuário inicia a seleção, as letras e as teclas se expandem.

Fonte: (FARAJ; MOJAHID; VIGOUROUX, 2009a)

A lei de Fitts pode ser utilizada em conjunto com as técnicas de frequência de um único caractere e a frequência de dígrafos.

3.3.2.2 Frequência de um único caractere

O método de frequência de um único caractere calcula a probabilidade de ocorrência de caractere em um corpus de uma linguagem. Esse cálculo resulta em uma tabela de probabilidades contendo a letra e a sua frequência de ocorrência. A partir dessa tabela de frequência, as teclas e as letras são reposicionadas e/ou redimensionadas de maneira que esses caracteres fiquem em posições e tamanhos que facilitam o acesso.

O trabalho de Merino et al. (2012) distribuiu as letras de maior frequência na língua espanhola entre as primeiras teclas do teclado. Prabhu e Prasad (2011) consideraram a frequência de uso da letra e a probabilidade dela ser a primeira letra da palavra para distribuir as letras entre as teclas.

Topal e Benligiray (2012) propuseram um teclado circular que aproxima as letras que possuem alta frequência de ocorrência. O trabalho de Gelormini e Bishop (2013) redimensiona as teclas do teclado virtual conforme a sua frequência na língua Inglesa. Assim, a letra “e”, por exemplo, tem um tamanho maior do que as outras letras que são menos frequentes no idioma Inglês.

Panwar, Sarcar e Samanta (2012) utilizaram a técnica de captura do movimento dos olhos para interagir com o teclado virtual. O *layout* do teclado proposto por eles posiciona a tecla de espaço no centro do teclado e as demais em volta dessa tecla. Assim, foram definidos dois níveis de teclas. O primeiro nível contém as letras com maior frequência de utilização e as posiciona mais próximas do espaço. O segundo nível possui as letras com menor frequência de

ocorrência. Esse *layout* mostrado na Figura 3.19 foi construído para minimizar o movimento dos olhos.

Figura 3.19 – Teclado proposto por Panwar, Sarcar e Samanta (2012). Esse teclado é similar ao teclado de Fitally, no qual as letras mais utilizadas ficam mais próximas da tecla de espaço equanto as demais teclas ficam na parte mais exterior do teclado.

Fonte: (PANWAR; SARCAR; SAMANTA, 2012)

3.3.2.3 Frequência de dígrafos

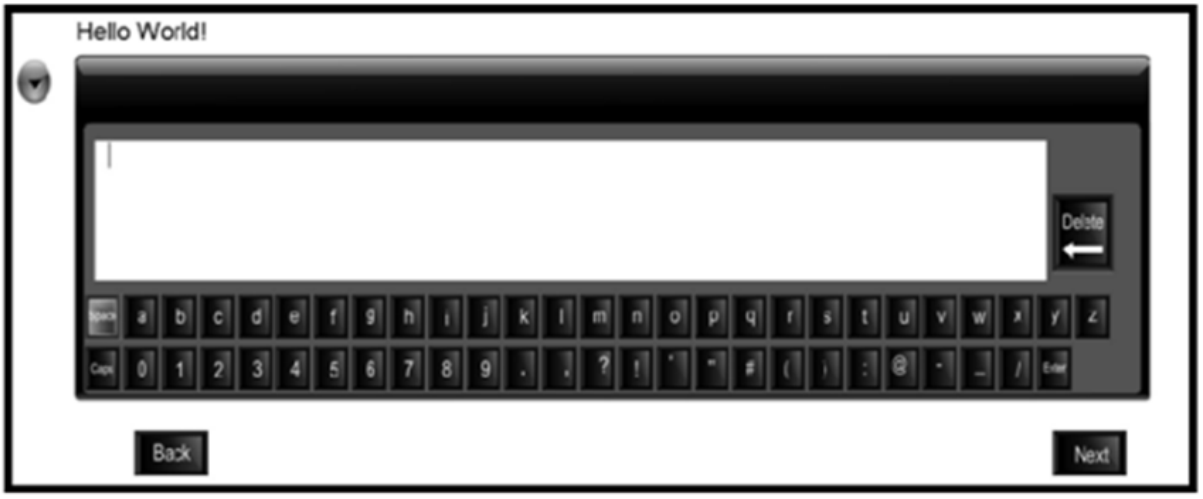
O método de otimização de *layouts* que utiliza dígrafos calcula a probabilidade de ocorrência de um dígrafo a partir de um corpus de uma linguagem. Com base nesse cálculo é construída uma tabela contendo o dígrafo e a sua frequência de ocorrência. Esse método utiliza a tabela gerada a partir desse cálculo para reposicionar e/ou redimensionar as teclas e as letras que compõem os dígrafos de forma ótima. O método de frequência de dígrafos é similar ao método de frequência de um único caractere (GELORMINI; BISHOP, 2013).

Gelormini e Bishop (2013) propuseram um teclado virtual que redimensiona as teclas de acordo com os dígrafos presentes na língua inglesa. As letras “th” formam um dígrafo em inglês. Portanto, conforme a proposta deles se a letra “t” é pressionada o tamanho da tecla que contém a letra “h” é aumentado com o objetivo de facilitar a sua seleção.

O trabalho de (MILLET; ASFOUR; LEWIS, 2009) apresentou uma ferramenta de construção e testes de teclados virtuais. Nessa ferramenta é possível reajustar o *layout* de acordo com a frequência de dígrafos da linguagem utilizada no teclado. A Figura 3.20 mostra um exemplo de teclado virtual construído pela ferramenta apresentada por Millet, Asfour e Lewis (2009).

Figura 3.20 – *Software* proposto por Millet, Asfour e Lewis (2009) para construção de teclados virtuais com o objetivo de realizar testes de performance e usabilidade.

Fonte: (MILLET; ASFOUR; LEWIS, 2009)



3.3.2.4 Fitts's Digraph

O modelo *Fitts's Digraph* foi construido a partir da lei de Fitts e da frequência de ocorrência dos dígrafos de uma linguagem. Esse modelo, proposto por MacKenzie e Zhang (1999) prevê a performance de digitação utilizando essa lei para calcular o tempo de movimentação entre todos os dígrafos do teclado. O resultado de cada tempo de movimentação entre cada dígrafo é ponderado pela frequência de ocorrência do dígrafo na língua inglesa. Considerando a média de tamanho da palavra igual a cinco, a média desses tempos são convertidas em palavras por minuto.

MacKenzie e Zhang (1999) desenvolveram um *layout* de teclado para dispositivos móveis denominado OPTI. Eles utilizaram o modelo *Fitts Digraph* para a construir esse *layout*. Após quatro horas de treino um usuário que usa o teclado OPTI atinge uma melhor performance de digitação do que quando ele utiliza o *layout* QWERTY (MACKENZIE; ZHANG, 1999).

O trabalho de (GHOSH et al., 2010) apresentou um novo teclado virtual para a língua Bengali utilizando *Fitts's Digraph*, bigram e trigram. Esse novo *layout* é dividido em três partes distintas: caracteres, flexões e pontuações.

3.3.2.5 N-gram

O modelo linguístico n-gram também pode ser usado para otimizar *layouts* de teclado. A partir da tabela de probabilidades calculada por bigramos, trigramos ou n-grams é proposto um *layout* que aproxima as letras que possuem maior frequência de ocorrência em sequência.

Sarcar et al. (2010) pesquisaram o corpus da Wikipedia na linguagem Bengali e analisaram ambas as frequências unigramas e bigramas. A partir dessa análise, eles propuseram um novo *layout* de teclado virtual para essa linguagem. Esse *layout* é mostrado na Figura 3.21.

Figura 3.21 – Teclado proposto por Sarcar et al. (2010) para a linguagem Bengali.

Fonte: (SARCAR et al., 2010)

3.3.2.6 Algoritmos de otimização

Ainda não existe uma solução definitiva que resolve o problema da baixa performance de escrita em teclados virtuais utilizados por pessoas com deficiência (POLácEK; MíKOVEC; SLAVíK, 2012). Portanto, para otimizar os *layouts* dos teclados virtuais alguns pesquisadores utilizam algoritmos de otimização. A finalidade desses algoritmos é identificar uma solução ótima para um problema minimizando uma função objetivo. O objetivo dos trabalhos que aplicam o algoritmo de otimização é encontrar uma combinação de teclas parcialmente ótima.

Para solucionar parcialmente o problema da distribuição de letras entre as teclas de um teclado de modo a otimizar o esforço e a performance de digitação é necessário primeiro definir a estrutura do teclado virtual. Em seguida, é essencial estabelecer quais as características que devem ser otimizadas como por exemplo, ergonomia, performance de digitação, redução da movimentação das mãos, esforço de digitação ou outra qualquer. Além disso, é preciso determinar qual o método será usada para otimizar o arranjo das letras.

Como o processo de otimização de *layouts* utilizando algoritmos é complexo esta tese apresenta um capítulo inteiro para abordar esse problema. Assim, a otimização de *layouts* dos teclados virtuais utilizando algoritmos de otimização é detalhada no Capítulo 4 deste trabalho.

3.3.3 Métodos de desambiguidade

Os teclados virtuais podem ser ambíguos ou não ambíguos. Para que os teclados ambíguos funcionem é necessário resolver a ambiguidade do teclado virtual. Existem duas maneiras de resolver essa ambiguidade, pelas letras ou pelas palavras. Os métodos que solucionam a

ambiguidade pelas letras realizam o processo de desambiguidade especificando cada letra selecionada. O método de palavras realiza a desambiguidade após a escolha de todos os conjuntos de letras que formam a palavra.

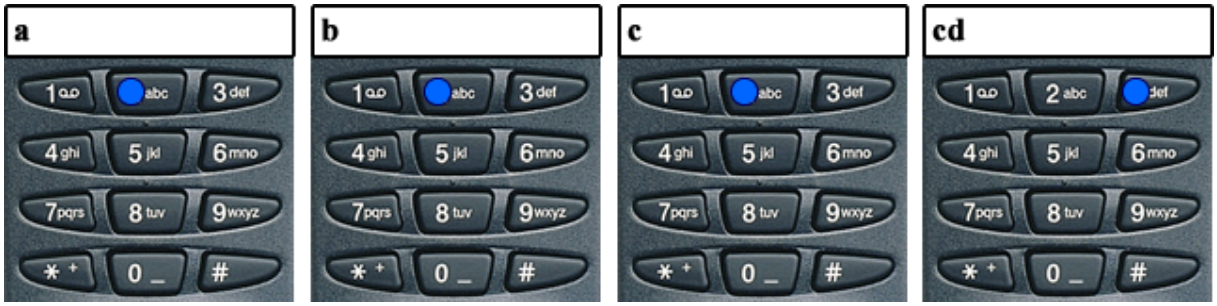
Os métodos de resolução encontrados na literatura foram multi-toque, duas teclas e algoritmos de desambiguidade.

3.3.3.1 Multi-toque

No método multi-toque o usuário escolhe o caractere selecionando uma ou mais vezes a mesma tecla até que a letra desejada seja escolhida (KWON; LEE; CHUNG, 2009). Por exemplo, se a tecla selecionada tiver as letras “a”, “b” e “c” e o usuário desejar escolher o caractere “b” ele terá que selecionar essa tecla duas vezes. Se o usuário desejar inserir a mesma letra duas vezes ou uma letra que esteja no grupo de teclas selecionado, ele deverá aguardar um tempo limite para pressionar a tecla novamente. Alguns sistemas disponibilizam uma tecla especial para eliminar esse tempo de espera.

Esse método de desambiguidade é utilizado frequentemente em teclados de dispositivos móveis. A Figura 3.22 ilustra o método de desambiguidade multi-toque. Nessa figura o usuário digita a palavra “cd”. O circulo em cima da tecla significa que ela foi pressionada.

Figura 3.22 – Desambiguidade utlizando o método multitoque. O processo apresentado pela figura mostra a escrita da palavra “cd”.



3.3.3.2 Desambiguidade com duas teclas

Os métodos de desambiguidade com duas teclas são realizados durante a digitação das teclas. Esses métodos realizam a desambiguidade das teclas pressionando duas teclas (SILFVERBERG; MACKENZIE; KORHONEN, 2000). A primeira tecla selecionada define o conjunto de letras que podem ser inseridas no texto. A segunda tecla pressionada realiza a desambiguidade e define qual das letras serão digitadas.

Por exemplo, se o usuário desejar selecionar a letra “e”. Considerando o teclado mostrado na Figura 3.22, ele deve pressionar a tecla que possui o conjunto de letras “d, e, f”. Logo depois de selecionar essa tecla ele precisa pressionar a tecla 2 vezes, escolhendo a letra “e” que é a segunda letra da sequência “d, e, f”.

Não existe limite de tempo para realizar a desambiguidade. As teclas de espaço e qualquer letra são inseridas com apenas um toque e dois toques, respectivamente (SILFVERBERG; MACKENZIE; KORHONEN, 2000).

3.3.3.3 Algoritmos de desambiguidade

Os algoritmos de desambiguidade são mais complexos. Essa abordagem utiliza um dicionário de termos para encontrar as palavras candidatas que correspondem a sequência de combinação dos grupos de letras (MOLINA; RIVERA; GóMEZ, 2009). Cada tecla possui um grupo de letras e os usuários selecionam apenas uma vez a tecla que contém a letra desejada. Quando a palavra é formada, a partir da combinação das letras escolhidas, o algoritmo disponibiliza uma lista com as possíveis palavras.

Por exemplo, considerando um teclado ambíguo com 9 teclas que possuia a sequência de letras em ordem alfabética, como mostrado na Figura 3.23. Se o usuário deseja escrever a palavra “lua”. Ele precisa pressionar a tecla que contém o conjunto de letras “j, k, l”, em seguida o usuário seleciona o conjunto “t, u, v” e logo depois, “a, b, c”. O sistema apresenta a ele uma lista de palavras formadas pela combinação dessas letras. Então, entre essas palavras o usuário seleciona a palavra “lua”.

Figura 3.23 – Teclado ambíguo com nove teclas. As letras de “a” a “z” são distribuidas entre as nove teclas seguindo a ordem alfabética.

3.3.3.3.1 LetterWise

Makenzie apresentou um novo método denominado *LetterWise* para realizar o processo de desambiguidade (MACKENZIE et al., 2001). Esse método utiliza uma tabela de prefixos para executar esse processo. O prefixo é composto pelas letras que precedem a próxima tecla a ser pressionada.

O método proposto por Mackenzie armazena uma tabela de prefixos e suas probabilidades de ocorrência. Quando o usuário introduz a primeira letra de uma palavra, não existem prefixos definidos. Para a segunda letra o prefixo tem tamanho um, e assim por diante até o tamanho máximo dos prefixos armazenados. A probabilidade da letra correta ser sugerida aumenta acentuadamente com a posição dentro da uma palavra. Por exemplo, considerando o idioma inglês, se o usuário pressiona o grupo das letras “d, e, f” após a digitação das letras “th” a próxima letra sugerida é “e”. Isso acontece porque em inglês a probabilidade de “th” ser sucedido pela letra “e” é maior do que a sucessão por “d” ou “f”.

Os prefixos não devem ultrapassar o limite da palavra (MACKENZIE et al., 2001). Portanto, as probabilidades relativas aos prefixos podem ser calculadas usando o modelo de linguagem k-gram e o corpus da linguagem que se deseja aplicar o método.

3.3.3.3.2 Algoritmo T9

O método T9 (texto em 9 teclas) de desambiguidade (GROVER; KING; KUSHLER, 1998) é usado para resolver a ambiguidade das palavras nos telefones com 9 teclas. Esse método de desambiguidade possui uma melhor performance de digitação se comparado ao método multi-toque (MOLINA; RIVERA; GóMEZ, 2009). Esse algoritmo foi proposto por Martin King e Kushler exatamente com o propósito de auxiliar as pessoas com deficiência. Em 1998, ele foi registrado como patente (GROVER; KING; KUSHLER, 1998). O sucesso desse produto originou a empresa Tegic Communications.

O objetivo do T9, em princípio, era auxiliar as pessoas com deficiência diminuindo o esforço necessário para entrar um texto por um teclado ambíguo. Além de diminuir esse esforço, o método T9 aumenta a performance de digitação minimizando a quantidade de teclas necessárias para inserir um texto. No final da década de 1990, vários aparelhos celulares começaram a utilizar esse método de entrada de texto (SILFVERBERG; MACKENZIE; KORHONEN, 2000).

Além de resolver o problema de ambiguidade das teclas, o T9 se adapta a maneira de escrita do usuário. A lista de palavras formadas pelos vários conjuntos de letras selecionados

pelo usuário é ordenada pela frequência de uso. Assim, esse método se ajusta a maneira de escrever de cada usuário posicionando as palavras do vocabulário do usuário no início da lista.

Outra característica do método T9 é a atualização do dicionário de dados. Se uma palavra não está no léxico do sistema o algoritmo permite inserir o novo termo. Assim, a próxima vez que o usuário entrar com essa palavra o algoritmo a apresentará na lista de sugestões.

No ano de 2007, a empresa Nuance Communications incorporou-se a Tegic Communications e adquiriu os direitos sobre o T9 (NUANCE.COM, 2015a). Logo, foram adicionadas novas funcionalidades a ele, como a predição de texto, a correção de erros por região, a correção da pronuncia, completar as palavras, os termos de atalho e a adição de pontuação automática (NUANCE.COM, 2015b).

O método de predição de texto consiste em prever as frases e as palavras que são utilizadas com mais frequência pelo usuário. Como o n-gram, esse método permite sugerir palavras antes mesmo de ter digitado o primeiro caractere.

O objetivo da correção de erros por região é ajustar as palavras que foram inseridas com erros de digitação. Esse tipo de correção avalia os grupos de letras próximas das teclas que foram pressionadas. Esses grupos são combinados com as teclas que efetivamente foram selecionadas. O resultado dessas combinações forma uma lista de termos que são apresentados ao usuário juntamente com a lista de sugestões de palavras.

O método de completar as palavras sugere um conjunto de flexões possíveis de serem adicionadas a raiz da palavra. Portanto, quando o usuário insere a raiz de uma palavra, o sistema sugere alguns termos que podem completar a raiz digitada.

As palavras de atalho podem ser codificadas para aumentar a velocidade de comunicação do sistema. Assim, um comando do tipo “obcvv” pode ser traduzido automaticamente para “Oi, bom dia! Como vai você?”. Além disso, as palavras comuns podem ser abreviadas como por exemplo, “vc” pode ser alterado para “você” automaticamente.

3.3.3.3.3 Algoritmo *text in n keys* (TNK)

O algoritmo *text in n keys* (TNK) é uma generalização do método T9 (MOLINA; RIVERA; GóMEZ, 2009). A diferença entre esses dois métodos está no número de teclas. O T9 utiliza 9 teclas e o TNK pode utilizar qualquer quantidade de teclas. Assim como o T9, o método TNK necessita de um dicionário de palavras. A cada sequência de teclas pressionadas, o algoritmo busca no dicionário as combinações dos grupos de letras que formam as palavras.

Molina, Rivera e Gómez (2009) utilizaram o método TNK com teclados de 4, 6, 9, 12 e 16 teclas. Eles concluíram que os teclados ambíguos com quatro teclas diminuem a interação

entre o usuário e o teclado, minimizando o número de vezes que o usuário interage com o sistema.

3.3.4 Adaptação do teclado virtual

A adaptação do *software* de comunicação ao usuário é essencial para que ele continue utilizando a ferramenta de comunicação. Um programa complexo pode causar insatisfação e frustração em seus usuários e dificultar sua adaptação a tecnologia (MCCOY et al., 2013). Consequentemente, esses usuários podem parar de utilizar esse software. Assim, o teclado virtual deve se adaptar ao usuário ou fornecer opções de personalização.

Foram encontrados na literatura duas maneiras de adaptação do teclado ao usuário. A primeira é realizada antes do usuário utilizar o teclado virtual. A segunda maneira é dinâmica e é realizada durante o processo de entrada do texto. Para essa última maneira foram encontrados trabalhos que adaptam a sequência das letras ou o tamanho das teclas de acordo com o texto inserido pelo usuário.

3.3.4.1 Personalização do teclado virtual

A personalização do teclado virtual é realizada pelo usuário configurando o *software* e modificando seus atributos manualmente. Algumas das características que podem ser alteradas são: a quantidade de teclas do teclado, a sequência dos símbolos, o tipo de predição e o tempo de espera entre as teclas durante o processo de varredura. A personalização desse teclado é realizada explicitamente, ou seja, o usuário configura o *software* da maneira que ele desejar.

Ao alterar a quantidade de teclas do teclado virtual, o usuário pode escolher entre um teclado ambíguo e não ambíguo. Por exemplo, se a quantidade de letras é igual a 26 e o usuário escolhe um teclado com cinco teclas, o número de letras por tecla deve aumentar tornando o teclado ambíguo.

O usuário pode desejar distribuir as letras de modo que o satisfaça. Assim, mesmo que as letras tenham sido organizadas utilizando alguma técnica como n-gram, o usuário pode preferir que as letras fiquem na sequência alfabética. Esse tipo de adaptação pode auxiliar o usuário a localizar as teclas mais facilmente.

As alterações no tipo de predição também podem ser consideradas no teclado virtual. O tempo de processamento da predição é um fator essencial para o usuário desse teclado (GARAY-VITORIA; ABASCAL, 2005). Se esse tempo é muito longo, pode frustrar as expectativas do usuário e deixá-lo insatisfeito. Assim, usuários que possuem computadores mais antigos podem desabilitar um ou outro método de predição com o objetivo de melhorar a performance de

processamento do teclado virtual.

Durante o processo de varredura, o tempo de espera entre as teclas é um fator determinante para a performance de digitação. Quanto menor é esse tempo maior é a velocidade de digitação. Porém, se o tempo de espera é muito curto pode aumentar a incidência de erros de digitação (FRANCIS; JOHNSON, 2011). Portanto, é necessário que o usuário possa ajustar o tempo de espera entre as teclas.

3.3.4.2 Adaptação dinâmica

Diferente da personalização, a adaptação dinâmica não é realizada manualmente e ocorre durante a entrada do texto. As modificações dinâmicas geralmente consistem em alterar a sequência das letras, modificar o tamanho das teclas e alterar o tempo de espera entre as teclas. Essas modificações utilizam um método ou uma técnica de predição para determinar a mudança que será realizada no teclado.

Os teclados que variam a posição das letras de acordo com a escrita do usuário podem realizar essa modificação utilizando o método de predição n-gram ou k-gram. Assim, as letras que são mais prováveis de serem digitadas são reposicionadas no início do teclado.

Os métodos n-gram e k-gram podem ser utilizados para expandir as teclas que possuem a maior probabilidade de serem digitadas após determinada sequência de símbolos. Além de destacar as teclas mais prováveis diminuindo o tempo de procura, esse método minimiza a probabilidade do usuário encostar acidentalmente em uma tecla vizinha à tecla desejada.

O tempo de espera entre as teclas também pode ser alterado dinamicamente. Para isso, o *software* deve considerar a velocidade de escolha das teclas pelo usuário. Usuários que conseguem selecionar as teclas rapidamente, provavelmente preferem um tempo de espera menor do que os usuários que demoram para pressioná-las.

O trabalho de Bhattacharya e Laha (2012) propôs um teclado virtual em Bengali que alterna o *layout* das teclas entre dois conjuntos de símbolos. Bengali é um dialeto indiano que possui 63 símbolos divididos em 11 vogais, 39 consoantes e 13 matras. Devido a grande quantidade de símbolos, esses pesquisadores dividiram o *layout* em dois conjuntos de símbolos. O objetivo da alternância do *layout* é apresentar os caracteres mais prováveis de serem digitados a maior parte do tempo. Essa alternância entre os conjuntos de símbolos é realizada considerando o método de paginação Menos Recentemente Utilizado. Esse método é usado para gerenciar a memória do computador.

3.4 Métricas utilizadas para Avaliar a Performance do Teclado Virtual

Assim que um novo método de entrada de texto é proposto e implementado, a primeira questão analisada está relacionada a sua velocidade de entrada de dados (WOBBROCK, 2007). Essa é uma questão crucial para novos métodos, pois se um método é mais lento do que o anterior ele provavelmente não será utilizado.

A precisão é outra característica importante para analisar a relação dos novos métodos de entrada. Quanto mais preciso é um método menor é a incidência de erros durante a sua utilização. Porém, de acordo com Wobbrock (2007) a velocidade de entrada de texto e a precisão do método possuem uma relação direta. Assim, não é vantagem um método ter uma performance de entrada de texto alta e apresentar uma quantidade muito grande de erros.

Para aferir a performance de entrada de dados, o esforço de digitação e a taxa de erros dos teclados virtuais foram pesquisadas seis métricas. Essas métricas são: palavras por minuto (PPM ou WPM), caracteres por minuto (CPM), gestos por caractere (GPC), teclas por caractere (KSPC), taxa de erro total (TER) e *minimum string distance* (MSD).

3.4.1 Palavras por Minuto

Talvez o método mais utilizado para aferir a performance de digitação é o número de palavras por minuto (WOBBROCK, 2007). 21 trabalhos pesquisados nesta tese apresentaram a performance de digitação utilizando esta medida.

O WPM ou PPM é calculado dividindo o total de símbolos transcritos pelo tempo necessário para escrevê-los em segundos, multiplicado por 60, que é a quantidade de segundos em um minuto, dividido pelo tamanho médio da *string* (NICOLAU et al., 2013). Esse tamanho é considerado cinco caracteres (YAMADA, 1980). Essa média considera o tempo entre a entrada do primeiro caractere até o último caractere da sentença. A Equação (3.8) mostra o cálculo da WPM.

$$WPM = \frac{|T|-1}{S} \times 60 \times \frac{1}{5} \tag{3.8}$$

em que: T é o número de caracteres transcritos e S é o tempo em segundos decorridos entre a entrada do primeiro caractere até a inserção do último termo do texto.

Na Equação (3.8) o literal transcrito pode conter letras, números, pontuação, espaços e qualquer tipo de caractere que possa ser impresso (WOBBROCK, 2007). Porém, esse texto não

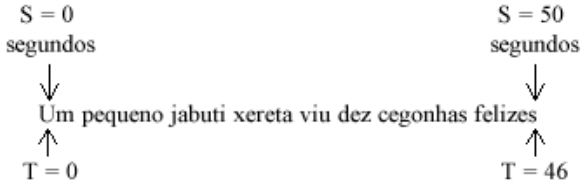
deve possuir *backspace* ou *delete*, que são caracteres não imprimíveis. O "-1" no numerador diminui o tamanho do texto em 1 caractere, pois a contagem do tempo começa a partir da entrada do primeiro caractere. Um exemplo de medição utilizando o WPM é descrito abaixo.

A aferição da performance de digitação pode ser realizada de duas maneiras distintas. A primeira é pela reescrita de um texto previamente definido e a segunda maneira é pela escrita de um texto idealizado pelo usuário. A primeira maneira é mais confiável do que a segunda (WOBBROCK, 2007). Isso acontece porque a segunda maneira requer uma demanda cognitiva em relação ao usuário e pode ocorrer uma lentidão durante o processo de digitação. Esse atraso pode influenciar no resultado da aferição.

Assim, é necessário primeiro definir um texto para ser transcrito. É importante que o literal a ser transcrito contenha o maior número de letras do alfabeto, fazendo com que o usuário passe por todas as teclas do teclado virtual. Os pangramas são frases que utilizam todas as letras do alfabeto com o mínimo de palavras possíveis. Um pangrama famoso na língua inglesa é "*the quick brown fox jumps over the lazy dog*". Essa frase é utilizada como o texto a ser redigitado com o objetivo de medir a performance de digitação dos métodos de entrada de texto.

Em português pode-se utilizar o pangrama "Um pequeno jabuti xereta viu dez cegonhas felizes". Assim que o usuário inserir o caractere "u" o tempo S começa a ser contabilizado. Se ele conseguir inserir todo o texto de 46 caracteres em 50 segundos, o valor WPM do método de entrada será igual a 10,8 palavras por minuto. A Figura 3.24 mostra a utilização da Equação (3.8).

Figura 3.24 – Exemplo de utilização da medição WPM



$$WPM = \frac{|46| - 1}{50} \times 60 \times \frac{1}{5}$$

3.4.2 Caracteres por Minuto

A medida de caracteres por minuto afere o número de caracteres digitados por minuto. (BHATTACHARYA; LAHA, 2012; WOBBROCK, 2007) utilizaram a mesma forma de cálculo da WPM suprimindo apenas a divisão da quantidade de caracteres por cinco. Alguns pesquisadores preferem reportar a performance de entrada em CPM (WOBBROCK, 2007). Na Equação

(3.9) T e S são os mesmos números utilizados na Equação (3.8).

$$CPM = \frac{|T| - 1}{S} \times 60 \tag{3.9}$$

As métricas WPM e CPM não consideram os erros inseridos pelos usuários durante a entrada de texto (WOBBROCK, 2007). (BHATTACHARYA; BASU; SAMANTA, 2008) mostraram a importância de considerar esses erros para aferir a performance de digitação de um teclado virtual. Pois, muitos softwares de entrada de dados podem possuir uma boa performance de comunicação, porém esses mesmos sistemas podem apresentar um alto índice de erro. Por exemplo, sistemas de reconhecimento de fala podem escrever rapidamente, mas podem apresentar vários problemas durante o processo de escrita.

Assim, é necessário a utilização de métricas que aferem os erros inseridos pelos usuários. As medidas teclas por caractere e gestos por caractere consideram esses tipos de erros.

3.4.3 Teclas por Caractere

A métrica KSPC (MACKENZIE, 2002) propõe uma relação entre o número de caracteres transcritos e a quantidade de teclas pressionadas para produzir o texto. Essa medida afere a quantidade de erros. Seu cálculo considera o número de vezes que a tecla de apagar foi pressionada, além das outras teclas necessárias para produzir o texto. Minimizar os KSPC significa diminuir o esforço do usuário para digitar um texto (WOBBROCK, 2007).

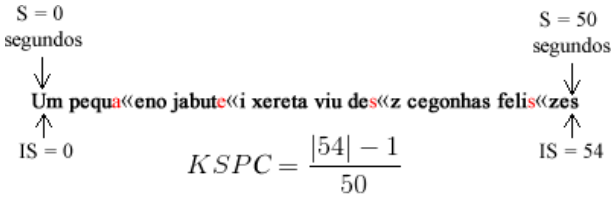
$$KSPC = \frac{|IS| - 1}{T} \tag{3.10}$$

em que: IS é o total de caracteres selecionados e T representa o total de caracteres transcritos.

Na Equação (3.10), diferente do termo T na medida WPM, IS inclui os caracteres não impressos como *backspace* e *delete*.

Como exemplo, pode-se considerar o texto “Um pequeno jabuti xereta viu dez cegonhas felizes”. A Figura 3.25 mostra a utilização da métrica KSPC. Nessa figura, o usuário digita o pangrama em 50 segundos realizando algumas correções. As correções realizadas pelo usuário estão assinaladas pelo caractere “«” e os caracteres errados estão na cor vermelha. Resolvendo esse exemplo dessa figura, o resultado do KSPC é de 1,08.

Figura 3.25 – Exemplo da utilização da métrica KSPS



3.4.4 Gestos por Caractere

Uma extensão do KSPC é a métrica gestos por caractere (GPC) (WOBBROCK, 2007). Essa medida considera a quantidade de ações ou gestos necessários para a entrada de um texto. O gesto é considerado como uma ação atômica de interação entre o sistema de entrada de dados e o usuário. Ao utilizar a GPC é necessário informar o que é exatamente um gesto para o sistema. Por exemplo, um gesto pode ser uma piscada de olho, um sopro, um sinal emitido pelo cérebro. A finalidade dessa medida é capturar a precisão do sistema em relação ao seu mecanismo de comunicação (WOBBROCK, 2007). A Equação (3.11) mostra o cálculo da GPC.

$$GPC = \frac{|IS\varnothing| - 1}{T} \tag{3.11}$$

em que: $IS\varnothing$ é o total de gestos executados durante a digitação do texto e S representa o total de caracteres transcritos.

3.5 Taxa de Erros

A taxa de erros do usuário é aferida como mostrado na Equação (3.12) (BHATTACHARYA; LAHA, 2012).

$$Toral\ Error\ Rate = \frac{INF + IF}{C + INF + IF} \times 100 \tag{3.12}$$

em que: INF é o número de caracteres inseridos incorretamente no texto transcrito, IF representa o número de teclas pressionadas que não são teclas de edição, como *delete* e *backspace*, e que não aparecem no texto transcrito e C é o número de caracteres transcritos corretamente.

3.5.1 Minimum String Distance

A *Minimum String Distance* (MSD) é a medida que afere o quão fiel é a sequência de caracteres transcrita do texto original. Portanto, a estatística MSD fornece a distância entre duas

strings determinando o menor número de operações de correção necessárias para transformar uma *string* na outra Wobbrock (2007). Essa distância é calculada pelo algoritmo proposto por (SOUKOREFF; MACKENZIE, 2001).

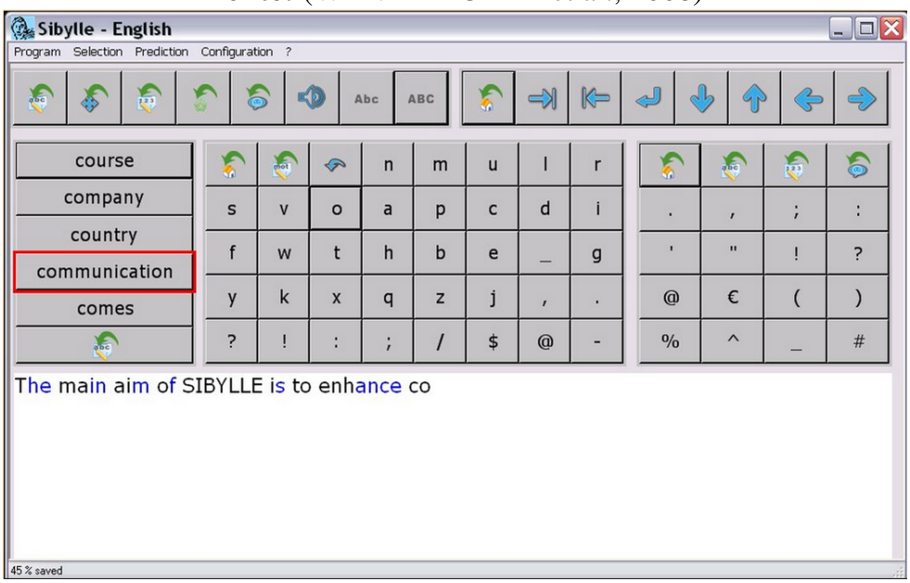
3.6 Trabalhos Correlatos

Os trabalhos correlatos mostraram outras pesquisas que desenvolveram teclados virtuais diferenciados com diversos objetivos distintos. Alguns desses objetivos são a diminuição dos erros de digitação, o aumento da performance de entrada de dados e a diminuição do esforço de digitação. Os teclados descritos nesta seção foram identificados durante o processo de revisão sistemática deste capítulo.

O Sybille (WANDMACHER et al., 2008) é um teclado virtual assistivo não ambíguo construído para os idiomas inglês, alemão e francês. Esse programa utiliza técnicas de predição de palavras e letras usando o método n-gram. O Sybille é dinâmico e adapta a distribuição das letras de acordo com a maneira de escrita do usuário. Esse teclado apresenta todos os símbolos dos teclados físicos agrupados em conjuntos que podem ser selecionados alternadamente. A Figura 3.26 ilustra esse teclado.

Figura 3.26 – Teclado Sibylle porposto por Wandmacher et al. (2008). Esse teclado utiliza diversas técnicas de predição e contempla caracteres especiais.

Fonte: (WANDMACHER et al., 2008)



O Fitaly é um teclado virtual construído com o objetivo de aumentar a performance de digitação utilizando *stylus* (MACKENZIE; ZHANG; SOUKOREFF, 1999). Este teclado possui duas barras de espaço. Além disso, o *layout* posiciona as letras mais comuns da língua inglesa perto das duas teclas de espaço. Esse *layout* foi desenvolvido utilizando o corpus e os

dígrafos da língua inglesa (MACKENZIE; ZHANG; SOUKOREFF, 1999). Trabalhos como os de (PANWAR; SARCAR; SAMANTA, 2012) utilizaram uma variação deste teclado em conjunto com a captura do movimento dos olhos para auxiliar a digitação. A Figura 3.27 ilustra esse teclado.

Figura 3.27 – O teclado Fitaly possui duas barras de espaço e posiciona as letras mais utilizadas da lingua inglêsa perto dessas teclas.

Fonte: (JAIN; BHATTACHARYA, 2010)

O OPTI apresenta uma proposta de teclado similar a do Fitaly (GUERRIER et al., 2011). O teclado virtual OPTI utiliza a lei de Fitts ponderada pela frequência de dígrafos (*Fitts' Digraph*) para posicionar as teclas e as letras (MACKENZIE; ZHANG, 1999). O objetivo deste teclado é aumentar a velocidade de digitação e minimizar a quantidade de erros de entrada de dados. A Figura 3.28 mostra esse teclado.

Figura 3.28 – O teclado OPTI possui quatro barras de espaço e utiliza a lei de Fitts e a frequência entre os digrafos para posicionar as letras entre as teclas.

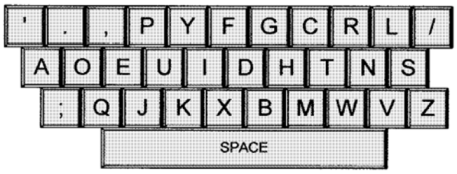
Fonte: (MACKENZIE; ZHANG, 1999)

O Dvorak é um teclado virtual que reorganiza as letras do *layout* QWERTY (MACKENZIE; ZHANG; SOUKOREFF, 1999). Esse *layout* foi desenvolvido com o objetivo de aumentar a performance de digitação dos teclados que podem ser operados utilizando as duas mãos. Esse teclado distribui as letras de acordo com a frequência dos dígrafos. Os dígrafos são posicionados de maneira que suas letras fiquem alternadas entre a mão esquerda e a direita. Além disso,

o *layout* do Dvorak possibilita diminuir a movimentação dos dedos reduzindo o esforço durante a entrada de texto (GUERRIER et al., 2011). A Figura 3.29 ilustra esse teclado.

Figura 3.29 – O teclado virtual Dvorak posiciona as letras entre as teclas fundamentado na frequência de ocorrência dos digrafos da lingua inglêsa.

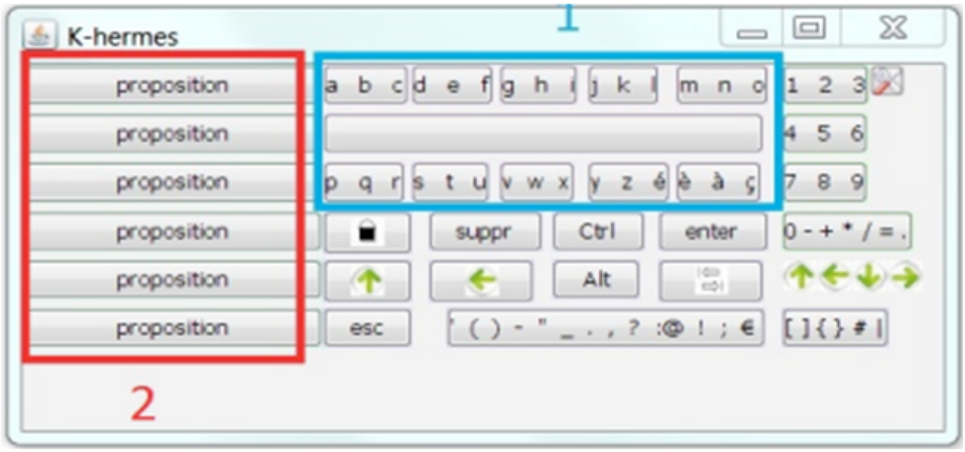
Fonte: (MACKENZIE; ZHANG; SOUKOREFF, 1999)



O teclado virtual K-Hermes é um teclado ambíguo que utiliza o método de multi-toque para realizar a desambiguidade das teclas (GUERRIER et al., 2011). Esse teclado apresenta um método de predição de palavras e distribui as letras entre as teclas utilizando a ordem alfabética. A principal contribuição dessa abordagem é a diminuição do esforço da entrada de texto. A Figura 3.30 mostra o teclado K-Hermes. A seleção um mostra a disposição das teclas e a seleção dois apresenta a lista de palavra sugeridas.

Figura 3.30 – O teclado virtual K-Hermes é um teclado ambíguo que utiliza letras, caracteres especiais e números. Esse taclado também possui um sistema de predição e a disposiação das letras entre as teclas é realizada de forma alfabética.

Fonte: (GUERRIER et al., 2011)

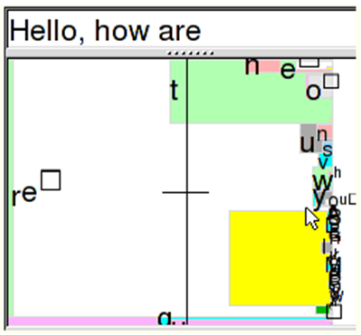


O Dasher é uma interface de entrada de texto orientada por gestos de apontamentos contínuos e naturais (WARD; BLACKWELL; MACKAY, 2000). Esta abordagem utiliza o movimento do mouse para entrada de texto. Dasher posiciona uma lista de caracteres com a maior probabilidade de ocorrência a direita do ponteiro do mouse. Esta abordagem pode ser utilizada quando não é possível utilizar teclados virtuais de tamanho convencional (WARD; BLACKWELL; MACKAY, 2000). A Figura 3.31 ilustra esse teclado.

O Chewing Word é um teclado virtual que apresenta suas teclas divididas em duas linhas (GRANGE, 2010). A sequência das letras é organizada de forma dinâmica de acordo com a

Figura 3.31 – O teclado virtual Dasher é uma forma de comunicação diferenciada. As letras passam no aplicativo de acordo com sua probabilidade de ocorrência.

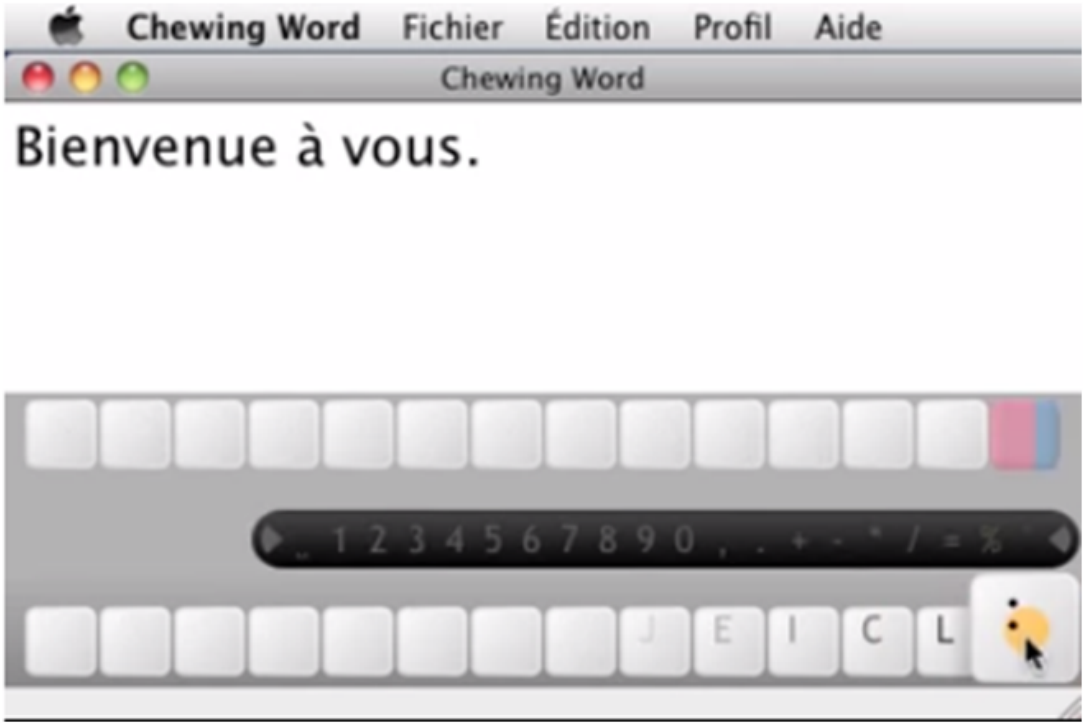
Fonte: (WARD; BLACKWELL; MACKAY, 2000)



digitação do usuário. A reorganização das letras posiciona a letra mais provável de ocorrer o mais próximo da última letra inserida. Além disso, esse teclado também utiliza as técnicas de predição de palavras. A Figura 3.32 mostra o teclado virtual *Chewing Word*.

Figura 3.32 – O teclado virtual Chewing Word foi criado a princípio para a lingua francêsa, mas atualmente ele suporta diversas linguas. Esse teclado possui um sistema de predição de letras que auxilia o usuário a selecionar as letras mais prováveis de ocorrerem.

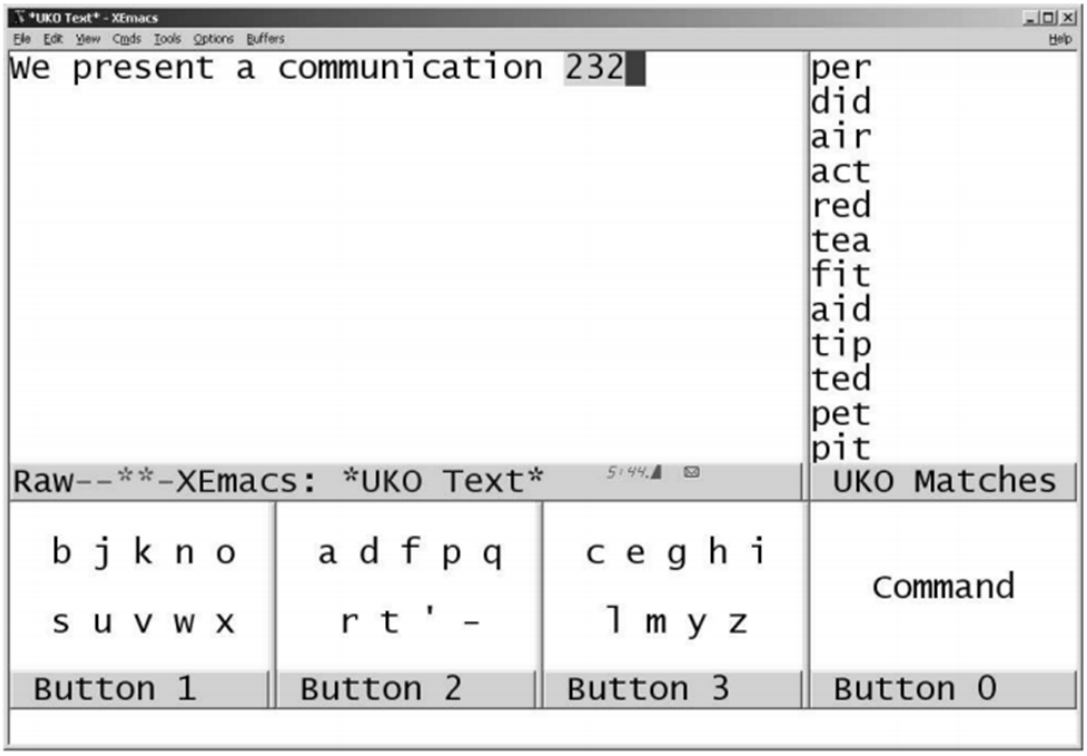
Fonte: (GRANGE, 2010)



O UKO-II é um teclado virtual ambíguo com apenas quatro teclas (HARBUSCH; KÜHN, 2003). Esse teclado utiliza um algoritmo de desambiguidade e permite que o usuário configure a sequência dos símbolos entre as teclas. O principal objetivo do UKO-II é facilitar a entrada de textos dos usuários com paralisia cerebral. A Figura 3.33 ilustra esse teclado.

Figura 3.33 – O teclado virtual UKO-II é ambíguo e possui um sistema de predição de palavras além de um algoritmo de desambiguidade.

Fonte: (HARBUSCH; KÜHN, 2003)



O K-Thot é um teclado virtual desenvolvido para pessoas com deficiência motora. Esse teclado tenta reduzir o número de movimentos necessários para a entrada de texto. Para tornar essa diminuição possível essa proposta aproxima as letras que possuem as maiores frequências de ocorrência em Frances. A Figura 3.34 mostra esse teclado.

Figura 3.34 – O teclado virtual K-Thot tenta diminuir o número de movimentos necessários para digitar uma palavra aproximando as letras com maior frequência de ocorrência.

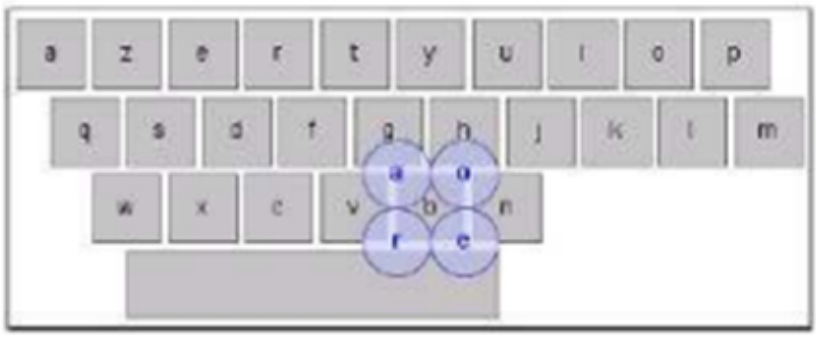
Fonte: (BAAS et al., 2010)

O KeyGlass é um teclado que obedece o *layout* Francês de teclas e letras. O objetivo desse teclado é diminuir o tempo e o esforço de digitação. Para atingir esse objetivo o KeyGlass

apresenta as letras mais prováveis de ocorrerem ao redor da última letra selecionada. A Figura 3.35 ilustra esse teclado.

Figura 3.35 – O teclado virtual *KeyGlass* apresenta as letras mais prováveis de ocorrerem perto da letra que foi selecionada.

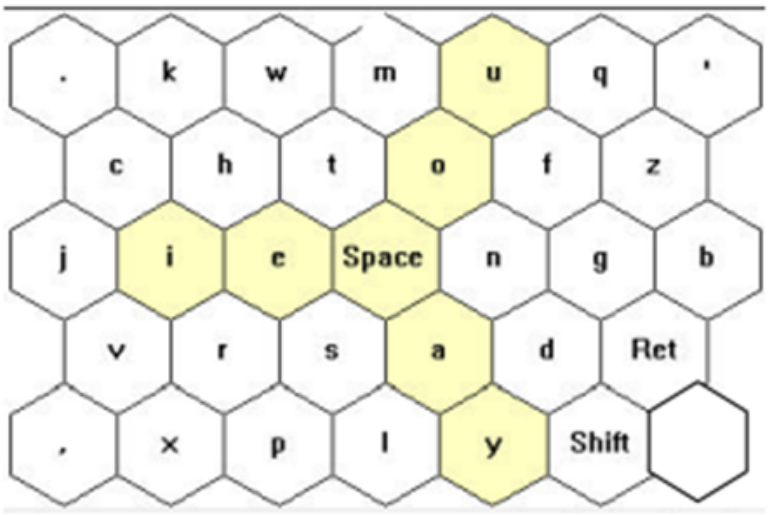
Fonte: (COLAS et al., 2008)



O teclado virtual Metrópolis apresenta um *layout* otimizado desenvolvido a partir da utilização do algoritmo de passeio aleatório usando como função objetivo a lei de Fitts. A finalidade do *layout* proposto por Zhai, Hunter e Smith (2000a) é aumentar a performance de digitação em teclados para dispositivos móveis. A Figura 3.36 ilustra esse teclado.

Figura 3.36 – Zhai, Hunter e Smith (2000a) utilizaram o algoritmo de otimização Metropolis para distribuir as letras entre as teclas do teclado virtual.

Fonte: (ZHAI; HUNTER; SMITH, 2000a)



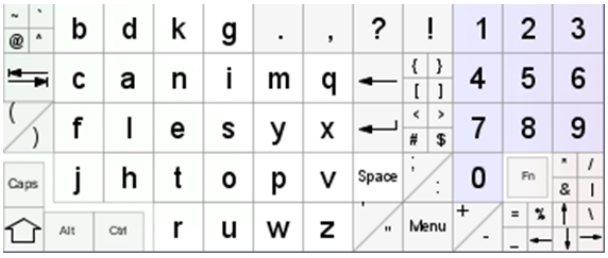
O teclado virtual proposto por Belatar e Poirier (2008) é chamado de HandiGyph. HandiGyph é ambíguo e possui apenas quatro teclas. As letras são distribuídas entre as teclas de acordo com a forma de similaridade entre o símbolo da tecla e a letra. Para realizar a desambiguidade é utilizado um algoritmo de desambiguidade. Esse teclado utiliza o método de varredura linear para auxiliar o usuário a selecionar as teclas. A Figura 3.37 mostra esse teclado.

Figura 3.37 – O teclado virtual *HandiGyph* proposto por Belatar e Poirier (2008) é um teclado ambíguo com quatro teclas que utiliza um algoritmo de desambiguidade.

Fonte: (BELATAR; POIRIER, 2008)

O teclado virtual ATOMIK é um acronomo para a *Alphabetically Tuned and Optimized Mobile Interface Keyboard*. O *layout* desse teclado é otimizado pelo algoritmo de otimização metrópolis. Assim, como o teclado virtual Metrópolis, o ATOMIK utiliza como função objetivo a lei de Fitts. A Figura 3.38 ilustra esse teclado.

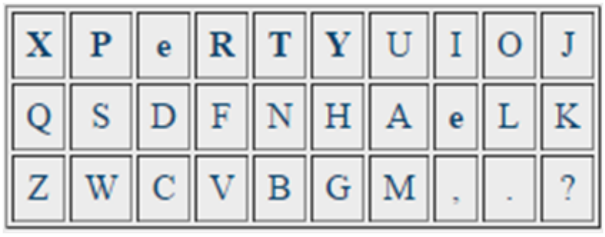
Figura 3.38 – O teclado *Atomik* também utiliza o algoritmo de otimização Metropolis para distribuir as letras entre as teclas. Esse teclado também disponibiliza números e caracteres especiais e de navegação.



Com a finalidade de diminuir a movimentação dos dedos durante a digitação, o teclado virtual Xpert aproxima as letras que compõem os bigrams da língua inglesa. A Figura 3.39 mostra esse teclado.

Figura 3.39 – O teclado virtual *Xpert* distribui as letras entre as teclas utilizando os bigrams da língua inglesa.

Fonte: (XPERTKEYBOARD.COM, 2015)



3.7 Considerações Finais

Neste capítulo foram apresentadas as características essências de um teclado virtual, os métodos e as técnicas de predição de palavras e letras, os métodos de desambiguidade, as métricas para aferir a performance de digitação e a taxa de erros. Finalmente, foram descritos os trabalhos correlatos.

O próximo capítulo mostra uma revisão de literatura realizada com o objetivo de identificar os trabalhos relacionados a otimização da distribuição das letras em relação as teclas do teclado virtual. Nesse capítulo, é apresentado o problema de distribuição de letras em relação as teclas, os métodos aplicados para resolver parcialmente esse problema e uma análise dos trabalhos correlatos. Finalmente, descreve-se uma fundamentação teórica sobre algoritmos genéticos e suas principais características.

CAPÍTULO 4

OTIMIZAÇÃO DE LAYOUTS

4.1 Introdução

A distribuição de letras entre as teclas de um teclado influenciam diretamente na performance e no esforço de digitação desse dispositivo. Distribuir as letras entre as teclas de modo ótimo não é uma tarefa trivial. O problema de distribuição pode ser reduzido polinomialmente ao problema de atribuição quadrática (LADANY, 1975). Esse último é NP-completo (PARDALOS; RENDL; WOLKOWICZ, 1994). Portanto, a solução do arranjo de teclas não pode ser realizada computacionalmente em um tempo polinomial.

Apesar de não possuir uma solução determinística que possa ser encontrada em um tempo viável, vários trabalhos foram desenvolvidos com a finalidade de obter um *layout* de teclado parcialmente ótimo. Cada trabalho utiliza uma técnica de otimização para melhorar uma ou mais características do teclado virtual.

Este capítulo apresenta o problema da distribuição de letras entre as teclas do teclado, as técnicas utilizadas para otimizar essa distribuição e os trabalhos que usam essas técnicas para gerar os teclados otimizados. Finalmente, é analisada de forma mais detalhada a metaheurística dos algoritmos genéticos utilizada nesta tese.

4.2 Problema de Otimização do Teclado

O problema de otimização do arranjo do teclado resume em distribuir uma quantidade de M caracteres em um conjunto de N teclas e identificar a combinação entre as letras e as teclas que otimizam determinados objetivos. Os critérios de otimização são variados e dependem principalmente da finalidade para qual o teclado é construído. Esses objetivos são classificados em duas categorias: critérios ergonômicos e critérios de eficiência relativos a desambiguidade e a predição de texto (YIN; SU, 2011).

Os critérios ergonômicos estão associados a redução do esforço de digitação relacionado a movimentação das mãos e dos dedos. O objetivo do critério de eficiência dos métodos de

desambiguidade e predição é reduzir o número de teclas pressionadas para realizar o processo de desambiguidade e/ou de predição das palavras.

A distribuição de caracteres em várias teclas pertence a uma classe de problemas computacionais que não podem ser resolvidos de maneira trivial. Isso acontece porque para encontrar a melhor solução é necessário verificar todas as combinações possíveis entre as letras e as teclas. O número de possibilidades pode ser calculado pela Equação (4.1) (LESHER; MOULTON; HIGGINBOTHAM, 1998).

$$A_{total} = \frac{M!}{\prod_{i=1}^{1=N}(k_i!) \times \prod_{n=1}^{n=K_{max}}(c_n!)} \tag{4.1}$$

em que: N é a quantidade de teclas, M representa o número de caracteres, c_n é o número de teclas com n caracteres e K_{max} representa o número máximo de caracteres que podem ser agrupados em uma tecla.

Para realizar o cálculo da quantidade de combinações possíveis, Lesher considerou 26 letras distribuídas entre nove teclas (LESHER; MOULTON; HIGGINBOTHAM, 1998). Ele analisou um *layout* que tinha oito teclas cada uma com três caracteres e uma tecla com dois. Esse arranjo permitiu gerar 3×10^{15} combinações de teclados diferentes. Mesmo com os mais avançados computadores, pesquisar todo esse espaço de soluções levaria muito tempo, tornando essa solução inviável.

Embora, problemas da classe NP-completo sejam considerados sem solução, existem métodos que identificam soluções parcialmente ótimas para esse tipo de classe (LIGHT; ANDERSON, 1993). Alguns desses métodos são descritos na seção 4.3 deste capítulo.

4.3 Soluções para Problemas de Otimização

O termo otimização geralmente consiste em minimizar uma ou mais funções $f(x)$. As funções f são chamadas de função mérito ou função objetivo e o parâmetro x representa as configurações das soluções do problema a ser resolvido. O principal objetivo dos algoritmos de otimização é identificar uma configuração de x que minimize a função ou as funções f. Portanto, solucionar os problemas de otimização consiste em três componentes: a função ou as funções f, a definição das configurações de x e o método de resolução. Apesar das funções objetivos e das configurações de solução serem estáticas para cada problema, o método de solução pode variar.

A metaheurística é um conjunto de métodos que coordenam os procedimentos de busca com estratégias de mais alto nível. Esses métodos utilizam um processo capaz de evitar os mínimos locais e realizar uma busca robusta no espaço de soluções de um problema (GLO-

VER; KOCHENBERGER, 2003). Portanto, a metaheuritisca pode obter boas soluções para problemas que não possuem um algoritmo de complexidade polinomial. Algumas metaheuristicas utilizadas para resolver o problema de otimização do teclado são: o recozimento simulado (KIRKPATRICK; VECCHI et al., 1983), os algoritmos genéticos (HOLLAND, 1975b), a nuvem de partículas (KENNEDY, 2010), a colônia de formigas (DORIGO; GAMBARDELLA, 1997) e as buscas locais.

O processo de recozimento simulado é análogo ao processo físico de recozimento dos metais. Nesse processo, as substâncias físicas são fundidas a altas temperaturas e depois resfriadas lentamente até atingir o estado sólido. Essa metaheuristica foi proposta por Kirkpatrick, Vecchi et al. (1983).

O procedimento computacional do recozimento simulado é iniciado a partir de uma solução aleatória S. Na primeira iteração é calculada uma nova solução S'. Em seguida, as duas soluções são comparadas. Se S' for melhor do que S, então S' substitui S. Entretanto, se S for melhor do que S', a substituição de S por S' depende da variável de controle temperatura (T). Nesse caso, T determina a probabilidade de S' substituir S quando a solução S é melhor do que S'.

No início do processo T possui um valor alto, assim, as novas soluções, mesmo que inferiores são aceitas com maior probabilidade. No decorrer do processo, com a redução do valor de T, as soluções menos otimizadas são desconsideradas com maior frequência.

Como o valor inicial de T é alto, o algoritmo de recozimento simulado realiza uma busca global evitando os mínimos locais. Com a diminuição de T, o algoritmo identifica um mínimo ótimo local e procura aperfeiçoar a solução encontrada.

Em 1960, os algoritmos genéticos (AG) foram propostos por John Holland (HOLLAND, 1975b). Esse método é fundamentado na teoria da evolução das espécies de Darwin. Os indivíduos de uma população evoluem conforme os operadores de mutação e de cruzamento.

Computacionalmente, uma população representa um conjunto de possíveis soluções P para o problema que está sendo tratado. Para cada indivíduo i pertencente a P é identificado o nível de adaptação do indivíduo ao meio. Esse nível é calculado de acordo com as funções objetivo do problema. A cada iteração, os indivíduos que obtiveram a melhor adaptação, ou seja, que conseguiram o melhor resultado sujeito as funções objetivo podem ser recombinados. Essa recombinação é realizada conforme a probabilidade de cruzamento configurada no algoritmo. Além disso, outros indivíduos podem ser submetidos a mutações. Essas mutações alteram os novos indivíduos aleatoriamente.

As boas soluções são mantidas e aprimoradas no processo de cruzamento. O processo de mutação evita os mínimos locais e procura encontrar soluções melhores. Existem várias

maneiras de realizar esses dois procedimentos e essas formas devem ser escolhidas de acordo com o problema a ser resolvido.

O método nuvem de partículas foi proposto por Eberhart and Kennedy em 1995 (KENNEDY, 2010). Esse algoritmo é inspirado no comportamento e na dinâmica social dos pássaros que durante o vôo possuem movimentações localmente aleatórias, mas globalmente determinadas. Essa técnica utiliza um grupo de partículas que se move no espaço de busca com o objetivo de identificar o mínimo global.

Enquanto o comportamento da nuvem procura identificar o ótimo global, a finalidade de cada partícula é procurar uma solução local para o problema. Cada local em que a partícula passa representa uma solução. No início do processo as soluções são geradas aleatoriamente. Assim, cada partícula parte de um ponto aleatório do espaço de soluções do problema.

A movimentação da partícula no espaço de busca gera novas soluções. Essa movimentação depende de três parâmetros: sociabilidade, individualidade e velocidade máxima. A sociabilidade determina a convergência de todas as partículas em direção a melhor solução global. O fator de individualidade define o valor de atração entre a partícula e a melhor solução identificada por ela. A velocidade máxima delimita o movimento da partícula.

A cada nova movimentação da partícula uma solução é gerada e a função objetivo é verificada. A partícula guarda todas as suas movimentações definindo um ótimo local. Igualmente aos algoritmos genéticos, o conjunto de soluções tende a preservar os melhores resultados e a desconsiderar as soluções menos adequadas. As alterações na solução ou nas movimentações das partículas podem ser influenciadas por suas partículas vizinhas. A solução parcialmente ótima é atingida quando todas as partículas convergem para um mesmo ponto.

O método da colônia de formigas foi inspirado no comportamento da locomoção delas em busca de comida. Como muitas espécies de formigas são quase cegas a comunicação entre elas é realizada pelo feromônio. Essa substância química é liberada por esse tipo de inseto durante sua movimentação. Assim, esse elemento químico forma uma espécie de trilha. Ao se movimentarem as formigas sentem o cheiro do feromônio e seguem o rastro que possui maior quantidade dessa substância. O método de otimização inspirado no comportamento das formigas foi proposto por Dorigo em 1991 (DORIGO; GAMBARDELLA, 1997).

No processo computacional, cada formiga representa uma possível solução para o problema a ser resolvido. Diferente dos métodos anteriores as soluções são elaboradas de forma construtiva. Cada solução S é modelada conforme a função r. r determina o rastro do feromônio. A cada etapa de formação da solução são alterados os pesos que regulam a intensidade do feromônio.

Assim que o S é gerado é verificada a função objetivo. Se a solução S for melhor do

que a solução anterior, S é armazenado. A outra solução é desconsiderada e uma nova iteração é iniciada.

Outra categoria de métodos heurísticos utilizados para otimização dos teclados virtuais são os algoritmos de busca local. Esses algoritmos iniciam a partir de uma solução qualquer e realizam pequenas alterações nessa solução com o objetivo de melhorá-la. O espaço de busca desse algoritmo é menor porque a solução não é alterada drasticamente. Os métodos aplicados a otimização do arranjo do teclado encontrados neste trabalho foram: n-optmization (CROES, 1958) e subida a colina.

O método *n-optimization* é um método de busca local fundamentado no trabalho proposto por Croes em 1958 (CROES, 1958). O método original é chamado de 2-opt e foi aplicado primeiramente para resolver o problema clássico do caixeiro viajante. Esse algoritmo realiza n permutações entre os elementos da solução inicial em busca de otimizações.

O algoritmo *n-optimization* inicia a partir de uma solução aleatória. Para cada elemento da solução é calculada n permutações com os outros elementos. Considerando n igual a dois, cada componente da solução é permutado apenas uma vez com outro elemento. Assim, o primeiro será permutado com $n-1$ elementos, o segundo com $n-2$ e assim por diante até que todas as permutações sejam realizadas. Toda vez que é efetuada uma troca de elementos é calculada a função objetivo. Se a permutação proporcionou melhoras na solução, a nova solução substitui a anterior. Caso contrário, nada acontece. Esse algoritmo termina quando ocorrerem todas as permutações.

O método de subida a colina é uma opção adequada para encontrar mínimos locais. Computacionalmente, esse método inicia a partir de uma solução S qualquer e realiza pequenas mudanças nessa solução. A cada alteração na solução inicial S é gerada uma nova solução S'. O S' é analisado de acordo com a função objetivo. Se S' for melhor do que a solução S, então S' substitui S. Caso contrário, nada acontece. Ao final da avaliação é iniciada uma nova iteração.

Outros métodos como o algoritmo de otimização Metropolis e aprendizagem estocástica por autômatos também foram utilizados para resolver o problema de arranjo do teclado.

Em 1953, Nicholas Metropolis propôs o algoritmo Metropolis (METROPOLIS et al., 1953). Esse algoritmo é utilizado para procurar o estado mínimo de energia em problemas de física estatística. A princípio é gerada uma solução S aleatória, em seguida são realizadas nesta solução pequenas alterações. A partir dessas mudanças é construída uma nova solução S'. Se essa solução for melhor do que a solução S, então S' substitui S. Entretanto, se S' for menor a solução S gera-se um número aleatório entre 0 e 1. Se esse número for menor do que a razão entre as duas soluções S' substitui S. Caso contrário, S permanece como solução ótima.

O método de aprendizagem estocástica por autômatos utiliza autômatos que aprendem

de acordo com o princípio da recompensa e da punição. O processo computacional funciona de modo simples. De acordo com o problema, é fornecido para o autômato um grupo de ações. Esse autômato interage com o ambiente do problema escolhendo quais ações devem ser selecionadas. De acordo com a ação escolhida, o autômato pode ser recompensado ou punido obedecendo uma certa probabilidade. Os autômatos de aprendizado são os que aprendem quais são as melhores ações para minimizar as penalidades (OOMMEN; VALIVETI; ZGIERSKI, 1990).

4.4 Trabalhos Correlatos

O teclado QWERTY foi desenvolvido por Christopher Sholes em 1873. O principal objetivo desse *layout* era evitar um problema mecânico que ocorria durante a datilografia em máquinas de escrever (SÖRENSEN, 2007; LIGHT; ANDERSON, 1993; GOETTL; BRUGH; JULSTROM, 2005). Portanto, a estrutura desse teclado não foi elaborada com a finalidade de otimizar a performance de digitação, mas sim de diminuir a velocidade com que as teclas eram pressionadas.

Após a construção da primeira máquina de escrever digital e posteriormente dos computadores, os problemas mecânicos se tornaram irrelevantes para a utilização dos teclados. A primeira tentativa de aprimoramento do teclado QWERTY foi realizada por Dvorak em 1986 (YIN; SU, 2011; GOETTL; BRUGH; JULSTROM, 2005). O objetivo do novo *layout* era aumentar a performance de digitação. Para atingir essa finalidade, Dvorak construiu um teclado considerando a frequência de ocorrência das letras no idioma inglês. Caracteres que possuíam maior utilização foram redistribuídos entre as teclas mais fáceis de serem acessadas.

Apesar dos avanços conseguidos por Dvorak, o teclado QWERTY permaneceu como o *layout* padrão dos teclados de computadores (LEVINE; TREPAGNIER, 1990; LIGHT; ANDERSON, 1993; EGGERS et al., 2003). Esse fato pode ser explicado parcialmente pela inconveniência da alteração do padrão do teclado.

Levine e Trepagnier (1990) propuseram a primeira tentativa de otimizar o *layout* do teclado utilizando técnicas computacionais. Nesse trabalho, eles utilizaram algoritmos genéticos e otimizaram os teclados do tipo ambíguo e não ambíguo.

As funções objetivo para os teclados do tipo não ambíguo consideram o tempo de transição das teclas e a sua frequência de ocorrência na língua inglesa. Na otimização dos teclados ambíguos a função mérito utilizada minimiza a quantidade de colisões entre as palavras e o número de teclas pressionadas. Ao comparar os teclados ambíguos propostos por Levine com os teclados convêncionais, o índice de eficiência na desambiguidade das letras foi de 86% usando dez teclas e ao utilizar quatro teclas foi de 62%.

Em uma tentativa de aprimorar o trabalho de Levine e Trepagnier (1990), Oommen, Valiveti e Zgierski (1990) propuseram para teclados ambíguos uma nova forma de minimizar o número de colisões entre as palavras. Oommen, Valiveti e Zgierski (1990) utilizaram a técnica de aprendizagem estocástica por autômatos. O autômato proposto por Oommen *et. al* distribui as letras entre as teclas com a finalidade de reduzir o nível de ambiguidade do teclado.

O processo de construção do teclado proposto em (OOMMEN; VALIVETI; ZGIERSKI, 1990) inicia com um *layout* aleatório. As palavras do dicionário são analisadas uma a uma de acordo com o teclado atual. A cada palavra pesquisada é verificado o número de colisões produzidos por essa palavra e as letras são permutadas de acordo com esse número. Oommen, Valiveti e Zgierski (1990) conseguiram um resultado melhor do que Levine e Trepagnier (1990).

Em 1993, Light e Anderson apresentaram um trabalho utilizando o algoritmo de Recozimento Simulado para otimizar os teclados não ambíguos (LIGHT; ANDERSON, 1993). Nessa abordagem foi usada uma única função objetivo. Essa função considera a frequência de utilização das letras no idioma inglês e o tempo de transição entre as teclas. Os teclados construídos com essa abordagem apresentam graus de eficiência de 8% a 3% melhor do que as abordagens QWERTY e Dvorak. Os trabalhos de Levine e Oommen não foram citados por Light.

Lesher, Moulton e Higginbotham (1998) apresentaram um trabalho de otimização utilizando o método *n-optimization*. Para calcular a eficiência dos teclados construídos usando esse método, Lesher *et al* elaboraram uma matriz de confusão. Essa matriz é construída a partir do corpus da linguagem e apresenta uma estrutura de dados composta bidimensional. Os valores da matriz representam o número total de vezes que o caractere β foi sugerido antes do caractere α quando α era o caractere desejado.

O algoritmo de otimização utilizado por Lehser et al usa a matriz de confusão para identificar os melhores arranjos do teclado. Para teclados com nove teclas esse método obteve bons resultados chegando a atingir 90% de eficiência em relação a teclados não otimizados. Lehser e seus colaboradores compararam seu trabalho com o de Levine e obteveram resultados melhores do que o seu correlato (LEVINE; TREPAGNIER, 1990).

Em 2001, Zhai et al avançaram com as pesquisas de teclados não ambíguo utilizando o algoritmo Metrópolis para construir um novo *layout* (ZHAI; SMITH, 2001). O objetivo de Zhai e seus colaboradores era distribuir as letras de modo a diminuir a movimentação das mãos e dos dedos de acordo com uma linguagem qualquer. Eles fizeram uma nova análise da performance de digitação de alguns teclados como QWERTY, CHUBON, FITALY e OPTI. A performance do teclado proposto por Zhai e Smith (2001) foi superior aos demais teclados e ao teclado QWERTY em até 10% e 43%, respectivamente.

Eggers et al. (2003) usaram a metaheurística da colônia de formigas para otimizar o teclado considerando apenas os aspectos ergonômicos. Esse trabalho foi desenvolvido no ano de

2003 e obteve um teclado diferenciado. Esse teclado agrupa as consoantes do lado esquerdo e o resto dos caracteres como vogais, pontuação e caracteres especiais no lado direito. Apesar deste trabalho ter encontrado um *layout* distinto de todas as outras abordagens, eles não realizaram nenhum tipo de experimento ou comparação para mostrar a performance da sua proposta.

Em 2005, os algoritmos genéticos foram utilizados novamente para a otimização dos teclados virtuais (RAYNAL; VIGOUROUX, 2005; GOETTL; BRUGH; JULSTROM, 2005). Raynal e Vigouroux (2005) usaram essa técnica para aprimorar os teclados não ambíguos. A otimização proposta por eles utilizou a mesma função objetivo de Zhai e Smith (2001). Assim, a principal finalidade era reduzir a movimentação das mãos e dos dedos entre os dígrafos da língua. Os resultados obtidos por Raynal e Vigouroux (2005) foram melhores do que os resultados de Zhai e Smith (2001).

Goettl, Brugh e Julstrom (2005) descreveram um método de otimização de teclados não ambíguos fundamentado nos princípios da eficiência propostos por Norman e Rumelhart (1983). Cada princípio apresentado por Norman e seus colaboradores foi codificado em uma função objetivo. Essas funções são minimizadas utilizando um algoritmo genético. Goettl et al conseguiram construir teclados de 40 a 20% mais eficientes do que os teclados QWERTY e Dvorak, respectivamente.

No ano de 2007, Sorensen desenvolveu um trabalho com o objetivo de aprimorar a digitação das mensagens pelo celular (SÖRENSEN, 2007). Para otimizar o teclado ambíguo de nove teclas foi utilizado um método de busca randômica. Sorensen minimizou o custo total de digitação de cada palavra e o número de colisões entre elas. A análise dos resultados desta otimização mostrou que as vogais não devem ser agrupadas na mesma tecla e alguns conjuntos de caracteres formam uma boa combinação para diminuir a ambiguidade.

Francis e Johnson (2011) utilizaram o algoritmo de subida da colina para redistribuir as letras de maneira ótima. Esse trabalho usou um teclado ambíguo e o método de varredura linha e coluna. Além disso, Francis et al. usaram duas funções objetivo. A primeira considerou o número de passos necessários para selecionar um determinado caractere. A segunda função penaliza determinadas estruturas do teclado, como por exemplo, estruturas que não mantém a sequência de grupos como a de números, caracteres de pontuação e letras. O trabalho de Francis e seus colaboradores não descreveu a eficiência da abordagem adotada.

O objetivo do trabalho proposto por Yin e Su (2011) foi aprimorar o arranjo dos teclados ambíguos utilizando o método de nuvem de partículas. Esse trabalho destacou dos outros, pois considerou quatro funções objetivo para a otimização. As propriedades otimizadas por Yin et al. foram acessibilidade da tecla, conforto da postura de digitação, número de teclas pressionadas e colisões entre as palavras. Eles conseguiram bons resultados e compararam o seu trabalho com outras abordagens como teclado alfabético, a abordagem de Levine e a distribuição de

frequência.

Finalmente, em 2012 Brouillette et al aumentaram a performance de digitação dos teclados não ambíguos utilizando algoritmos genéticos (BROUILLETTE; SHARMA; KALITA, 2012). O trabalho deles também considerou o tempo de aprendizado para os novos usuários. O teclado proposto por Brouillette, Sharma e Kalita (2012) possuía o dobro da performance de digitação do teclado QWERTY.

4.5 Análise

Os trabalhos de Yin e Su (2011), Lesher, Moulton e Higginbotham (1998) e Levine e Trepagnier (1990) otimizaram os esforço necessário para digitar uma palavra. Porém, esses trabalhos consideraram apenas o acesso direto a tecla. Os pacientes com SE utilizam o acesso indireto, pois o teclado assistivo usa o método de varredura para selecionar as teclas. Avaliar a fase de seleção da tecla no processo de otimização é essencial para a construção desse tipo de teclado (MIRÓ-BORRÁS et al., 2009). Apenas o trabalho de (FRANCIS; JOHNSON, 2011) considerou o acesso indireto as teclas.

Apesar de otimizar o teclado minimizando o esforço de varredura, Francis e Johnson (2011) construíram os teclados não ambíguos. Porém, os teclados que possuem uma quantidade de teclas reduzidas e agrupam as letras entre essas teclas são mais indicados para os sistemas assistivos (TOPAL; BENLIGIRAY, 2012; MOLINA et al., 2009).

Os trabalhos encontrados, exceto o de Francis, utilizaram um corpus distinto da língua para a geração e otimização do teclado. Essa abordagem apresenta três deficiências. A primeira desvantagem é que a maneira de escrita e o vocabulário do usuário não são considerados. A segunda deficiência é que os apanhados de textos nem sempre são equivalentes entre as pesquisas. Essa dissemelhança pode resultar em teclados otimizados de diversos tipos tornando difícil a comparação entre as abordagens. Finalmente, a utilização do corpus desconsidera a evolução da linguagem e do modo de escrita do usuário.

Os trabalhos de otimização de teclado constroem um software otimizado para a linguagem e não para o usuário que o utiliza. Características importantes como o vocabulário do usuário e sua forma de escrever são desconsideradas. Além disso, muitas palavras existentes no corpus podem não ser conhecidas pelo usuário. Esse tipo de abordagem pode otimizar a entrada de palavras que não são relevantes para determinados tipos de paciente.

Utilizar teclados otimizados para um tipo particular de texto pode gerar um ganho substancial na redução do esforço e no aumento da performance de digitação (FRANCIS; OXTOBY, 2006). Além disso, a otimização do teclado depende das propriedades estatísticas de cada texto

(FRANCIS; JOHNSON, 2011). Apesar da pesquisa de Francis e Johnson (2011) ressaltar a importância do texto e do contexto na otimização, a grande maioria dos trabalhos desconsidera esta característica.

Outro fator que mostra a influência do texto na otimização do teclado é a diferença dos resultados apresentados no trabalho de Lesher, Moulton e Higginbotham (1998). Nesse trabalho, eles compararam seus resultados aos de Levine e Trepagnier (1990) e observaram uma diferença relevante na eficiência dos teclados com menos de seis teclas. É possível que essa diferença seja relativa a distinção entre os corpus utilizados em cada trabalho.

Finalmente, o corpus da linguagem contém as palavras, jargões e maneiras de escrita de várias pessoas distintas. Todas essas características variam entre indivíduos, além de depender do contexto histórico, temporal, cultura, sócio econômico e regional de cada um. Além disso, a escrita é um processo dinâmico e evolui de acordo com o tempo e relações estabelecidas, as palavras que eram comuns a algum tempo atrás podem ficar obsoletas.

Fundamentado nas pesquisas de Francis e Oxtoby (2006), Francis e Johnson (2011), pode-se observar que o conhecimento do usuário tem influência maior do que o contexto ou até mesmo que o texto durante a otimização do teclado. Por exemplo, as pessoas podem escrever e conversar sobre diversos assuntos, porém as palavras utilizadas estão limitadas ao vocabulário de cada um. Avaliando uma conversa sobre álgebra entre um professor doutor em matemática e um aluno do ensino médio. Nota-se que as palavras utilizadas pelo professor são muito mais específicas do que as palavras usadas pelo aluno. Nesse caso, o assunto é o mesmo, porém o vocabulário de cada um determina as palavras utilizadas durante a discussão.

Uma alternativa de otimização que pode diminuir esse problema é a geração de teclados de acordo com o vocabulário de cada usuário. Construir um teclado personalizado e evolutivo para um usuário específico, pode ser uma solução adequada para aumentar a sua capacidade de entrada de dados e principalmente diminuir o esforço de digitação.

4.6 Algoritmos Genéticos

Como citado na seção 4.3 deste capítulo, os algoritmos genéticos simulam a teoria da evolução das espécies de Darwin. Essa teoria mostra a seleção natural, em que ocorre uma competição entre indivíduos por sobrevivência. Nessa seleção os indivíduos mais aptos estão propensos a se reproduzirem e os menos aptos a se extinguirem (ENGELBRECHT, 2007). Outro conceito é a teoria da hereditariedade, introduzida por Mendel, Nessa teoria, os filhos herdam as características genéticas dos pais (MICHALEWICZ, 1996).

Não existe uma definição rigorosa aceita por toda a comunidade de computação evolu-

tiva que diferencia AGs de outros métodos de computação evolutiva. No entanto, pode-se dizer que a maioria dos métodos chamados de "AGs" têm, pelo menos, os seguintes elementos em comum: as populações de cromossomos (indivíduos), a seleção de acordo com a aptidão calculada por meio da função objetivo, o cruzamento para gerar novos descendentes e a mutação aleatória (MELANIE, 1999). Decompondo esses elementos, pode-se reorganizá-los em componentes e processo. As subseções 4.6.1 e 4.6.2 deste capítulo apresentam respectivamente, o detalhamento dos componentes e do processo padrão dos AGs.

4.6.1 Compontes

Nesta seção são apresentados os principais componentes presentes nos AGs.

4.6.1.1 Indivíduo

Os indivíduos representam as possíveis soluções para o problema a ser resolvido. Cada indivíduo, também chamado de cromossomo, é composto por genes. O gene foi um termo elaborado em 1909 por Johannsen. Na definição da genética clássica, gene é a unidade funcional da hereditariedade na qual estão presentes os ácidos nucleicos, portadores de informações genéticas que proporcionam a diversidade entre os indivíduos. Cada gene pode possuir valores distintos, cada valor possível de um gene é denominado Alelo. O Alelo ocupa uma determinada posição em um cromossomo. Essa posição é definida como *locus* (SADLER; LANGMAN, 2007).

Uma parte importante na implementação de um AG é definir a codificação do indivíduo. Essa codificação determina a estrutura de dados do cromossomo e pode ser chamada também de genótipo. As principais formas de codificação são: codificação binária, codificação real e permutação.

Na codificação binária, os indivíduos são compostos por cadeias binárias de 0's e 1's. Essa abordagem é motivada pela teoria dos esquemas que justifica como benefício o desempenho do algoritmo em maximizar o paralelismo implícito inerente ao AG (HOLLAND, 1975a).

Na codificação real cada indivíduo é composto por cadeias de números reais. O objetivo dessa codificação é abordar problemas de otimização numérica com parâmetros reais, em que na maioria dos casos, apresentam resultados superiores à codificação binária (GOLDBERG, 1989).

A codificação por permutação é definida como um arranjo linear de elementos de um conjunto finito. Essa codificação normalmente é aplicada em problemas de ordenação. Para n objetos distintos, existem $n!$ permutações destes objetos (KNUTH, 1998).

4.6.1.2 População

A população de um AG é definida por um conjunto finito de indivíduos. Essa população possui uma dinâmica em que as características importantes dos indivíduos de uma população são propagadas para os indivíduos descendentes a cada geração. Essa dinâmica é possível por meio do processo evolutivo, descrito na subseção 4.6.2 deste capítulo.

O dimensionamento da população é uma característica importante para o AG. Algumas questões relevantes em relação a esse dimensionamento são (JONG, 2006):

- Qual deve ser o tamanho da população?
- Quantos descendentes devem ser produzidos por geração?
- Qual a influência desses tamanhos na execução do AG?
- A escolha de valores é crítica para o desempenho da resolução do problema?
- Se a escolha dos valores é crítica, quais são os valores apropriados para o problema?

A dimensão da população pode ser vista como uma medida do grau de pesquisa paralela suportada pelo AG. Uma vez que é a partir dessa população que novos pontos de pesquisa são construídos a cada geração. Essa dimensão pode ser ajustada conforme a complexidade do problema a ser resolvido. Frequentemente, em problemas significativamente complexos são utilizadas dimensões de 100 à 1000 indivíduos (JONG, 2006).

Algumas características importantes da população são:

(a) **Geração**: é a representação do número de vezes que a população passou pelo processo evolutivo;

(b) **Adaptação**: é a média dos resultados do valor de aptidão de cada indivíduo;

(c) **Grau de Convergência**: representa o quão próxima a média de adaptação da geração atual está em relação as gerações anteriores;

(d) **Diversidade**: é a variação entre os genótipos da população. Ela é fundamental para ampliar o espaço de busca. Um valor baixo de diversidade pode ocasionar a convergência prematura do AG; e

(e) **Elite**: são os melhores indivíduos da população. Uma técnica comum nos AGs é o elitismo. Nessa técnica, os melhores indivíduos são preservados para a próxima geração.

4.6.1.3 Função objetivo

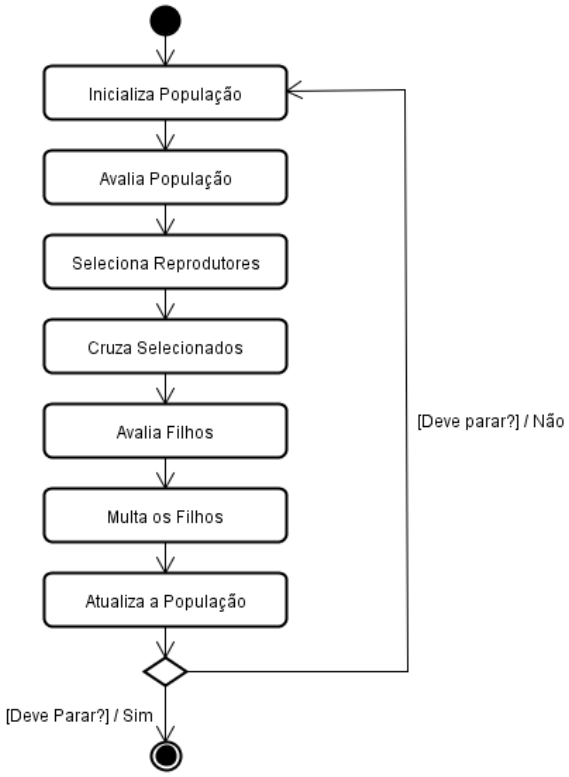
A função objetivo de um problema de otimização é construída a partir dos parâmetros relacionados ao problema analisado. Normalmente, essa função pode ser representada por uma equação ou um algoritmo. A função objetivo determina a qualidade do indivíduo, ou seja, indica se um determinado conjunto de parâmetros é bom ou não como solução para o problema. Essa qualidade é definida como aptidão ou *fitness* (HOLLAND, 1975a; MICHALEWICZ, 1996; GOLDBERG, 1989).

É importante que as funções objetivos sejam codificadas para não terem uma complexidade de tempo elevada. Isso porque, essas funções são executadas diversas vezes, mais especificamente uma vez para cada indivíduo. Assim, se uma função objetivo possui um tempo de execução muito elevado, o algoritmo genético pode ficar lento a ponto de se tornar inviável.

4.6.2 Processo evolutivo

A Figura 4.1 ilustra o processo evolutivo de um AG. Nas seções a seguir são explicadas cada etapa desse processo (HOLLAND, 1975a; MICHALEWICZ, 1996; GOLDBERG, 1989).

Figura 4.1 – Processo de execução do algoritmo genético.



4.6.3 Inicialização

A primeira etapa de um AG é a inicialização. Essa etapa consiste na geração da sua população inicial de soluções. Costuma-se utilizar funções aleatórias para gerar os indivíduos. O objetivo dessa técnica é minimizar a convergência prematura, ampliando a diversidade dos indivíduos e consequentemente aumentar o espaço de busca. Existem alternativas para as funções aleatórias. A finalidade dessas alternativas é reduzir as possíveis deficiências quando a representação do indivíduo é mais complexa ou quando se deseja um melhor desempenho (SCHULZ, 1997; GOLDBERG, 1989).

Os operadores mais tradicionais de inicialização são (SCHULZ, 1997; GOLDBERG, 1989):

(a) **Inicialização aleatória uniforme**: esse operador atribui um valor do conjunto de alelos para cada gene do indivíduo que é sorteado de forma aleatoria e uniforme;

(b) **Inicialização aleatória não uniforme**: neste operador, a escolha do valor a ser atribuído ao gene depende da frequência dele em relação aos demais valores;

(c) **Inicialização aleatória com dope**: indivíduos otimizados são inseridos na população. A presença desses indivíduos pode guiar o AG a uma convergência prematura; e

(d) **Inicialização parcialmente enumerativa**: os indivíduos são inseridos de tal forma que a população começa o processo evolutivo possuindo todos os formatos possíveis de uma determinada ordem.

4.6.4 Avaliação

Na etapa de avaliação, cada indivíduo da população tem sua aptidão calculada utilizando a função objetivo. Sempre que um novo indivíduo é gerado, o cálculo dessa função é realizado repetidamente ao longo do processo evolutivo. Em alguns casos, quando a complexidade da função objetivo requer um alto custo computacional, é comum o uso de funções não determinísticas. Essas funções avaliam apenas parte das características dos indivíduos (GOLDBERG, 1989).

4.6.5 Seleção

Esta etapa é responsável por determinar a preservação dos bons genes, ou seja, das características dos indivíduos que tiveram boa adaptação ao problema. Os genes dos indivíduos

mais adaptados são mantidos de geração em geração, tornando o processo de evolução cumulativo e adaptativo (DAWKINS, 1996).

Durante o processo de seleção, o grau de adaptação pode ser considerado isoladamente, sem nenhum tipo de alteração ou pode-se aplicar uma função sobre esse valor. Alguns exemplos de funções são: escala linear, escala truncada ou escala ponderada. A escala linear aplica em uma função linear sobre o valor da função de avaliação. A escala truncada subtrai um múltiplo do desvio médio padrão dos valores da função de avaliação e altera os valores negativos para zero. Finalmente, a escala ponderada eleva o valor da função de avaliação a uma determinada constante (SCHULZ, 1997; GOLDBERG, 1989; JONG, 2006; DAWKINS, 1996).

Após calcular o grau de adaptação de cada indivíduo é necessário realizar o processo de seleção para determinar quem participará da fase de reprodução. Os principais métodos para realizar a seleção desses indivíduos são: *ranking*, giro da roleta, torneio e uniforme (SCHULZ, 1997; GOLDBERG, 1989).

O método de *ranking* ordena os indivíduos de acordo com o seu grau de adaptação. As probabilidades de cruzamento são definidas conforme a posição de cada indivíduo no *ranking*. Assim, as soluções que possuem um grau de adaptação melhor tem uma maior probabilidade de serem escolhidas para participarem da fase de cruzamento (SCHULZ, 1997; GOLDBERG, 1989).

A seleção por giro da roleta calcula o somatório de todos os graus de adaptação de cada indivíduo da população. Em seguida é sorteado um valor que pertence ao intervalo entre zero e o valor total. Assim, é selecionado o indivíduo que possui o valor de adaptação que se encontra na faixa do valor sorteado (DAWKINS, 1996).

O método de torneio constrói diversos grupos aleatórios compostos com os indivíduos da população. Em cada grupo gerado é eleito o melhor indivíduo para o cruzamento de acordo com seu grau de adaptação. Os indivíduos eleitos são selecionados para participarem da reprodução (SCHULZ, 1997; GOLDBERG, 1989).

Finalmente, a forma uniforme define que todos os indivíduos tem a mesma probabilidade de serem selecionados. Apesar dessa forma possibilitar uma maior pesquisa do espaço de soluções, esse tipo de seleção possui uma chance pior de realizar uma melhora durante o cruzamento (SCHULZ, 1997; GOLDBERG, 1989).

4.6.6 Cruzamento

Na fase de cruzamento, os indivíduos selecionados são agrupados em pares e os genes de cada um são utilizados para a formação de um novo indivíduo. A combinação dos escolhidos pode ser realizada de várias formas, entre elas estão: escolha aleatória, *inbreeding*, *line breeding*

e a auto fertilização.

Na escolha aleatória os indivíduos formam um casal de maneira randômica, ou seja, dois indivíduos são escolhidos entre os selecionados ao acaso. No método *inbreeding* são combinados os indivíduos que são parentes. O *line breeding* cruza um indivíduo otimizado com um subconjunto de indivíduos e os filhos são selecionados para realizarem um novo cruzamento. Finalmente, no método de alto fertilização o indivíduo é cruzado com ele mesmo.

O cruzamento nem sempre ocorre com os elementos que foram eleitos para reproduzirem. A reprodução dos indivíduos pais está condicionada a uma probabilidade de ocorrência do operador de cruzamento.

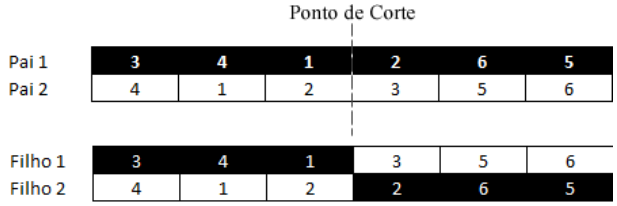
Para realizar o cruzamento é necessário determinar o modo como os genes dos pais são combinados para gerar um novo indivíduo. Novamente, existem diversas maneiras de combinar esses genes. O modo de combinação é denominado operador de cruzamento. Alguns exemplos desses operadores são: cruzamento de um ponto (1PX), cruzamento de múltiplos pontos (MPX), cruzamento segmentado (SX), cruzamento uniforme (UX) e cruzamento por combinação parcial (PMX) (GOLDBERG, 1989).

O operador de cruzamento por um ponto sorteia um número aleatório que está entre o intervalo da faixa de valores do comprimento dos indivíduos da população. Por exemplo, se os indivíduos forem representados por um vetor de 10 posições, o número sorteado deve estar entre o intervalo de 0 a 10.

Para realizar o cruzamento, o operador gera o primeiro filho com os genes que vão do início do vetor até o número sorteado pertencente ao primeiro pai. O restante dos genes do primeiro filho recebe do número sorteado até o final do vetor os genes do segundo pai. Para gerar o segundo filho é realizada a operação inversa.

A Figura 4.2 mostra um exemplo de cruzamento de uma população de indivíduos compostos por um vetor de seis posições de números que variam de um a seis. Esse cruzamento é realizado utilizando o operador 1PX.

Figura 4.2 – Cruzamento utilizando o operador 1PX de indivíduos compostos por um vetor de seis posições de números que variam de um a seis.



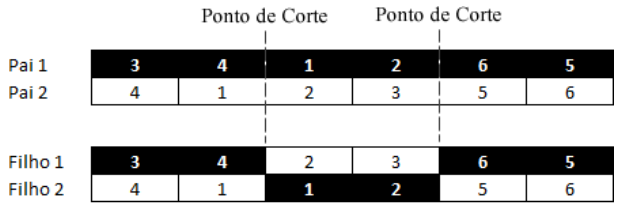
O método de cruzamento com múltiplos pontos sorteia um número n fixo de pontos p que pertencem ao intervalo da faixa de valores do comprimento dos indivíduos da população. Para cada ponto sorteado é gerado um intervalo. Esse intervalo é composto por dois pontos,

um ponto de início e outro de fim. Para o primeiro ponto sorteado, p_1, o intervalo começa no início do vetor e termina no ponto em questão. Para os demais pontos o intervalo varia do ponto anterior, p_{x-1} até o ponto atual p_x. O intervalo final inicia no último ponto p_n e segue até o final do vetor de genes do indivíduo.

O filho número um recebe o primeiro intervalo de genes do primeiro pai e o segundo intervalo de genes do segundo pai. Assim, os genes dos filhos são compostos de permutações entre os dois pais até terminar o número de intervalos. Para o segundo filho esse processo se inverte, ou seja, o primeiro intervalo de genes é fornecido pelo segundo pai.

A Figura 4.3 ilustra um exemplo de cruzamento de uma população de indivíduos compostos por um vetor de seis posições de números que variam de um a seis. Esse cruzamento é realizado utilizando o operador MPX com dois pontos iguais a dois e quatro.

Figura 4.3 – Cruzamento utilizando o operador MPX de indivíduos compostos por um vetor de seis posições de números que variam de um a seis.



O operador de cruzamento segmentado funciona do mesmo modo que o MPX. A única diferença entre esses dois operadores é que toda vez que o método segmentado é executado ele sorteia a quantidade de pontos.

O método uniforme de cruzamento percorre os genes dos pais e para cada gene analisado é sorteado aleatoriamente se o filho receberá o gene do primeiro ou do segundo pai.

O operador de cruzamento por combinação parcial, também conhecido como operador de dois pontos, escolhe dois pontos de corte e os filhos herdam os genes que estão entre os pontos de corte dos dois pais. Os genes restantes são preenchidos com valores considerados mais adequados para cada filho.

4.6.7 Mutação

A fase de mutação inicia após o cruzamento. Essa fase é importante, pois tende a evitar os mínimos locais com o objetivo de explorar o espaço de soluções. Assim, a mutação altera as características intrínsecas dos indivíduos resultantes do cruzamento para variar à população buscando novas soluções. Existem diversas formas de realizar a mutação, algumas delas são: mutação por *flip*, mutação por troca ou *swap* e mutação *creep* (SCHULZ, 1997; GOLDBERG, 1989).

Na mutação por *flip*, o gene a ser mutado recebe um valor sorteado aleatoriamente que pertence ao domínio da solução. Assim, o novo valor atribuído ao gene deve ser um valor válido para o problema. A mutação por troca ou *swap* sorteia *n* pares de genes. Cada par sorteado é permutado entre si. Finalmente, na mutação *creep* um valor aleatório é somado ou subtraído do valor do gene (SCHULZ, 1997; GOLDBERG, 1989; JONG, 2006; DAWKINS, 1996).

A mutação não ocorre para todos os elementos gerados na fase de cruzamento. Essas alterações são condicionadas a uma probabilidade de ocorrência do operador de mutação. Uma taxa muito alta de mutação pode reduzir ou até mesmo eliminar da população características fundamentais para a solução do problema (SCHULZ, 1997; GOLDBERG, 1989).

4.6.8 Atualização

Após concluir as etapas de cruzamento e mutação é necessário inserir os novos indivíduos na população. Essa atualização depende da política adotada pelo AG. Na abordagem tradicional, o número de indivíduos da população se mantém fixo. Portanto, o número de indivíduos gerados na etapa de cruzamento é igual ao número de indivíduos da população original. Assim, todos os novos indivíduos substituem os indivíduos da população anterior (SCHULZ, 1997; GOLDBERG, 1989; JONG, 2006).

Algumas alternativas diferentes do método de atualização tradicional são: o número de indivíduos gerados pode ser menor do que o tamanho da população, o tamanho da população pode variar a cada geração e pode variar o critério de inserção dos novos indivíduos. Nessa última alternativa, pode-se considerar que somente os indivíduos descendentes com valores de aptidão melhores do que seus pais serão inseridos na população. Outra alternativa é inserir todos os indivíduos mantendo somente os *n* melhores (SCHULZ, 1997; GOLDBERG, 1989; JONG, 2006).

4.6.9 Finalização

Na fase final do AG não é executada nenhuma operação genética. Nessa fase é realizado apenas uma verificação do critério de parada. Esse critério pode avaliar o número de gerações realizadas ou o grau de convergência da população atual. Além disso, o critério de parada depende do problema a ser resolvido (GOLDBERG, 1989).

4.7 Considerações Finais

Este capítulo apresentou uma revisão de literatura que auxiliou no desenvolvimento da metodologia evolutiva de teclados assistivos. Primeiro, foi apresentado o problema da distribuição de letras entre as teclas do teclado. Em seguida, foram descritos os métodos de otimização que podem solucionar parcialmente esse problema. Os trabalhos correlatos foram apresentados após a descrição dos métodos. Finalmente, foi descrita a análise desses trabalhos e realizada uma fundamentação teórica sobre algoritmos genéticos.

O próximo capítulo descreve o teclado assistivo, compacto, otimizado e evolutivo proposto neste trabalho e define todas as características, os métodos e as métricas utilizadas para desenvolver esse teclado. Todas as funcionalidades adotadas para construir o teclado virtual são justificadas a partir da literatura pesquisada nesta tese. Finalmente, é descrita a metodologia evolutiva de teclados virtuais assistivos.

CAPÍTULO 5

TECLADO VIRTUAL PROPOSTO

5.1 Introdução

A primeira revisão sistemática realizada para desenvolver esta tese possuía como tema central a Tecnologia Assistiva e a Comunicação Alternativa e Aumentativa (CAA). Esse estudo tinha a finalidade de identificar as maneiras de realizar a comunicação entre um paciente com SE e o ambiente externo. Notou-se que esse problema era muito mais abrangente do que o esperado. Com a finalidade de diluir o problema foi proposto um sistema de CAA dividido em cinco módulos. O objetivo desta tese é elaborar o módulo responsável pela comunicação, mais especificamente o teclado virtual assistivo.

Para elucidar o estado da arte e esclarecer os vários requisitos de um teclado assistivo foi elaborado uma nova revisão sistemática. O ponto central dessa revisão foi os teclados assistivos. Esse estudo pesquisou os problemas relativos a esse tipo de dispositivo ou *software*, assim como, suas principais características, as diferentes maneiras de otimizá-lo e as métricas utilizadas para medir a sua performance. Além disso, essa pesquisa identificou os padrões, os métodos e as definições que agrupados podem produzir um teclado assistivo otimizado e compacto para pacientes com SE.

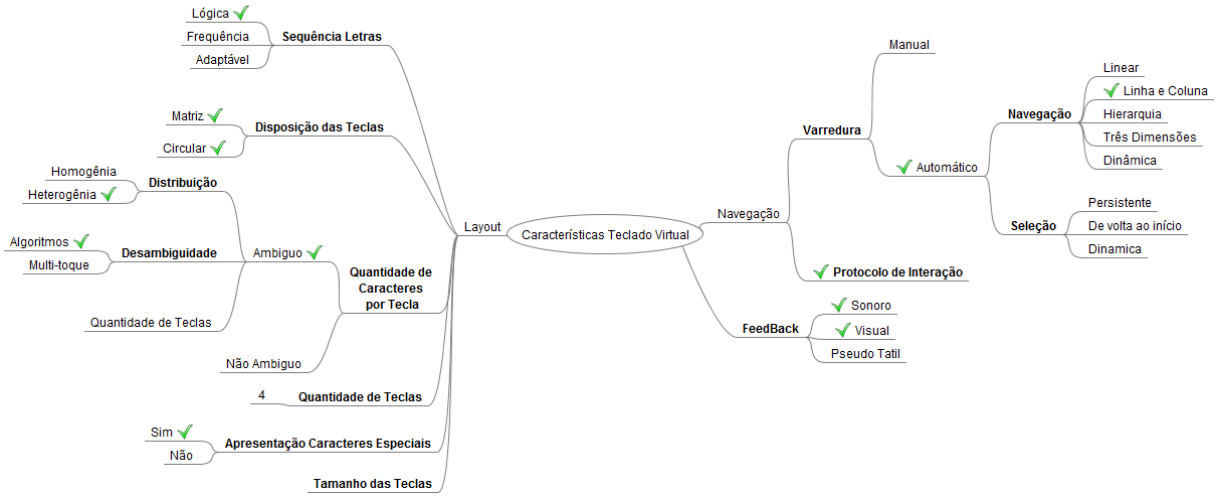
Apesar da revisão sistemática sobre teclados assistivos ter identificado diversos métodos de otimizar esse teclado, não foi encontrado uma maneira de otimizar a distribuição das letras entre as teclas do teclado virtual. Assim como, uma forma eficiente de tornar o teclado adaptável a cada tipo de usuário. Finalmente, foi executada uma revisão de literatura para identificar os métodos utilizados para otimizar a distribuição das letras entre as teclas do teclado virtual. Essa revisão originou a metodologia evolutiva de otimização dos teclados virtuais assistivos proposta nesta tese.

O objetivo deste capítulo é apresentar um *software* de teclado virtual para pessoas com SE. Nesse *software* foram implementadas as características, os métodos de otimização, as métricas de medição e a metodologia evolutiva de teclados virtuais elaboradas a partir de todo conhecimento adquirido a partir das três revisões.

5.2 Características do Teclado Proposto

O primeiro passo para definir um teclado virtual é projetar a maneira como ele deve ser apresentado para o usuário. O mapa mental ilustrado na Figura 5.1 mostra as características destacadas no Capítulo 3 desta tese e o valor de cada característica selecionada para compor o teclado virtual assistivo.

Figura 5.1 – Características selecionadas para o teclado virtual assistivo. Os valores das características selecionadas estão marcados com a figura de um V.



Em negrito na Figura 5.1 estão destacadas as características do teclado virtual, os valores que possuem a figura de um V são os valores selecionados para cada atributo. As características do teclado virtual assistivo foram escolhidas de acordo com as abordagens mais adequadas para pacientes com SE. O objetivo dessas escolhas é minimizar o esforço de digitação e aumentar a performance de entrada dos dados. A explicação de cada escolha é realizada neste capítulo.

5.2.1 Quantidade de caracteres por tecla

Teclados virtuais ambíguos possuem uma performance de digitação melhor do que os teclados não ambíguos (TOPAL; BENLIGIRAY, 2012; MOLINA et al., 2009). Além disso, um dos principais objetivos do teclado ambíguo é reduzir o esforço de digitação (BHATTACHARYA; LAHA, 2012). Teclados que possuem mais de um caractere por tecla reduzem a interação entre o usuário e o teclado diminuindo o esforço de digitação (GUERRIER et al., 2011). Assim, optou-se por adotar a abordagem ambígua em relação a quantidade de caracteres por tecla.

5.2.2 Quantidade de teclas

O número de teclas do teclado virtual é quatro, pois não ocorre um ganho substancial na diminuição do esforço quando a quantidade de teclas é menor do que esse número (TANAKA-ISHII; INUTSUKA; TAKEICHI, 2002b). Espera-se que reduzindo o número de teclas a performance de digitação aumente. Isso porque, quanto maior a quantidade de teclas maior é o tempo necessário para percorrer todas as opções do teclado (FRANCIS; JOHNSON, 2011). Assim, com a diminuição do número de opções, o tempo total de varredura pode ser minimizado melhorando a performance de digitação (MOLINA; RIVERA; GóMEZ, 2009).

Além de diminuir o tempo de varredura, um número menor de opções no teclado pode facilitar a busca por um caractere (SHARMA et al., 2012). Diminuindo esse tempo, o usuário poderá selecionar a tecla desejada mais rapidamente.

Quando possível, as letras são distribuídas entre as teclas de forma homogênea. Na impossibilidade desse tipo de distribuição, as letras restantes são distribuídas entre as primeiras teclas. A finalidade dessa distribuição é maximizar as chances das letras desejadas estarem entre as duas primeiras teclas, sem prejudicar o grau de ambiguidade do teclado. Portanto, o teclado virtual possui a primeira e a segunda teclas com sete caracteres e a terceira e a quarta com seis caracteres.

5.2.3 Apresentação dos caracteres especiais

Os caracteres especiais como pontuação, números e outros caracteres permitem ao usuário escrever textos formais. Excluir esses caracteres do teclado virtual traria uma grande perda no poder de expressão do teclado. Entretanto, incluir esses caracteres ao teclado pode aumentar o número de teclas e consequentemente diminuir a performance de entrada de dados e aumentar o esforço de digitação. Para resolver esse problema optou-se por desenvolver conjuntos de teclas.

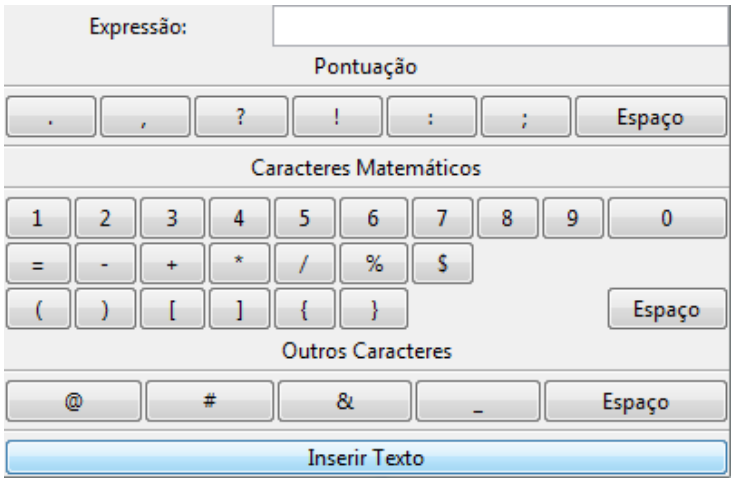
Os conjuntos de teclas são responsáveis por agruparem as teclas com objetivos específicos. Esses conjuntos podem ser alternados utilizando o protocolo de interação que é explicado na sessão 5.2.5.1 deste capítulo. Foram definidos dois conjuntos de teclas: escrita e edição. O conjunto de escrita é composto pelas quatro teclas com os caracteres do alfabeto, uma tecla de espaço, a tabela de desambiguidade e a predição de palavras. O conjunto de teclas de edição é composto por cinco teclas: apagar, parágrafo, outros caracteres, inserir e correção. A Figura 5.2 mostra a disposição desses componentes.

Figura 5.2 – Conjunto de teclas de escrita e conjunto de teclas de edição.

A tecla “Apagar” permite ao usuário remover o caractere anterior a letra. Essa tecla foi posicionada estrategicamente para permitir ao usuário apagar sequencialmente os caracteres. A tecla “Pararágrafo” permite ao usuário inserir um salto de linha com tabulação para iniciar um novo parágrafo.

A pontuação e os caracteres especiais são apresentados selecionando a tecla “Outros Caracteres”. Assim que essa opção é selecionada é exibida uma janela com uma matriz com os caracteres especiais como, pontuação, números e tabulações. A Figura 5.3 ilustra essa matriz. Para permitir o usuário selecionar esses caracteres é utilizado o método de varredura linha e coluna.

Figura 5.3 – Matriz de caracteres especiais que é disponibilizada pelo sistema para inserção desse tipo de caractere.



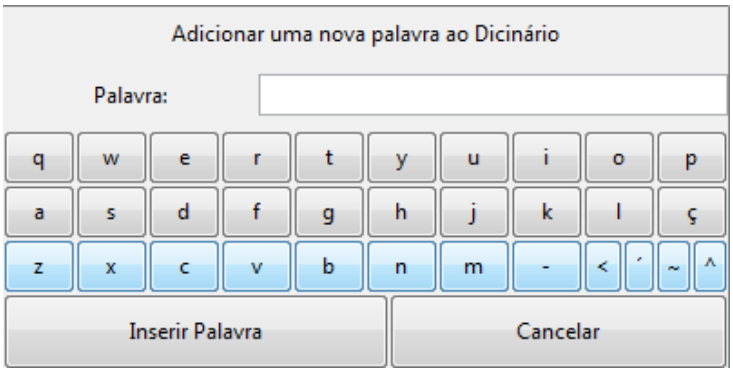
Nem sempre todas as palavras do vocabulário do usuário estarão presentes no léxico do sistema. Segundo (SHARMA et al., 2010), novas palavras devem ser adicionadas ao dicionário dinamicamente. O objetivo da tecla inserir é adicionar palavras que ainda não estão presentes no dicionário do sistema. Ao selecionar essa opção é apresentado um teclado virtual não ambíguo. Esse teclado utiliza a varredura linha e coluna e por essa funcionalidade o paciente pode inserir novas palavras. A Figura 5.4 mostra o teclado virtual não ambíguo.

Finalmente, a tecla correção auxilia o usuário a corrigir o texto. A princípio essa funcionalidade permiti o paciente alternar entre os parágrafos. Assim que um parágrafo for selecionado, inicia um processo de varredura entre as palavras. O usuário pode editar qualquer palavra e inserir textos antes ou depois da palavra.

5.2.4 Sequência das letras

A distribuição de letras utilizando suas frequências é uma técnica promissora para aumentar a performance de digitação (ZHAI; HUNTER; SMITH, 2000b). Porém, a distribuição

Figura 5.4 – Interface do teclado virtual disponibilizada para permitir que o usuário insira novas palavras.



das letras em forma alfabética é mais usada para usuários deficientes, entretanto esse tipo de distribuição limita a performance de digitação (TOPAL; BENLIGIRAY, 2012).

Por esta característica ser importante para o aumento da performance de digitação e principalmente para a diminuição do esforço de entrada dos dados foi realizado uma nova revisão. A finalidade dessa revisão foi identificar um método que distribuisse as letras entre as teclas de maneira ótima. Essa pesquisa foi apresentada no Capítulo 3 deste trabalho. A partir dessa revisão desenvolveu-se uma metodologia de otimização evolutiva de teclados virtuais que é apresentada na sessão 5.5 deste capítulo. A metodologia proposta é responsável por realizar a distribuição das letras entre as teclas.

5.2.5 Método de seleção

Para realizar a seleção das teclas é utilizado uma estratégia híbrida. Essa estratégia usa duas técnicas: protocolo de interação e método de varredura. Optou-se por utilizar esses dois métodos com o objetivo de minimizar o esforço do usuário.

5.2.5.1 Protocolo de interação

O protocolo de interação é usado para possibilitar que o usuário alterne entre os conjuntos de teclas: escrita e edição. A princípio o sistema executa o método de varredura no grupo de escrita. Se o usuário desejar selecionar as teclas relacionadas ao grupo de edição ele pode fechar os olhos e aguardar um sinal sonoro que é emitido pelo sistema. Esse sinal indica que um novo conjunto de teclas foi selecionado. Assim que uma nova seleção for realizada o sistema inicia o processo de varredura no conjunto de teclas selecionado.

5.2.5.2 Método de varredura

O método de varredura adotado é linear, automático e assim que uma tecla for escolhida volta ao início do teclado. O método de varredura linear é indicado em teclados com alta ambiguidade, ou seja, teclados que possuem poucas teclas (MOLINA; RIVERA; GóMEZ, 2009). O objetivo desse método é diminuir o esforço do usuário.

A varredura automática tende a minimizar a quantidade de interações necessárias entre o usuário e o teclado virtual. Isso porque esse tipo de varredura necessita apenas de um tipo de estimulo do usuário: seleção (MOLINA; RIVERA; GóMEZ, 2009). Infelizmente, as técnicas de varredura são lentas e atingem de 5 a 20 caracteres por minuto (TANAKA-ISHII; INUTSUKA; TAKEICHI, 2002a).

O método de voltar o foco de varredura ao início do conjunto de teclas assim que uma tecla é selecionada é uma estratégia válida para auxiliar na otimização das teclas. Esse método possibilita agrupar as letras mais utilizadas na primeira tecla. Teclados otimizados permitem que os usuários usem as teclas mais acessíveis $43,5\%$ das vezes (EGGERS et al., 2003). Desenvolver um teclado que possibilite o usuário selecionar a tecla mais fácil na maior parte das interações, pode aumentar consideravelmente a performance de digitação. Assim, colocar as letras mais usadas no início do teclado é uma estratégia adequada para melhorar a performance de entrada de dados.

A princípio o tempo de seleção da tecla é configurado pelo usuário. Porém, esse tempo é dinâmico e pode ser alterado de acordo com a velocidade de escolha das teclas. Portanto, quanto mais rápido as teclas forem selecionadas, menor é o tempo entre elas. Caso o teclado identifique um elevado número de erros efetuados pelo usuário, esse tempo pode ser reajustado.

5.2.5.3 Tamanho das teclas

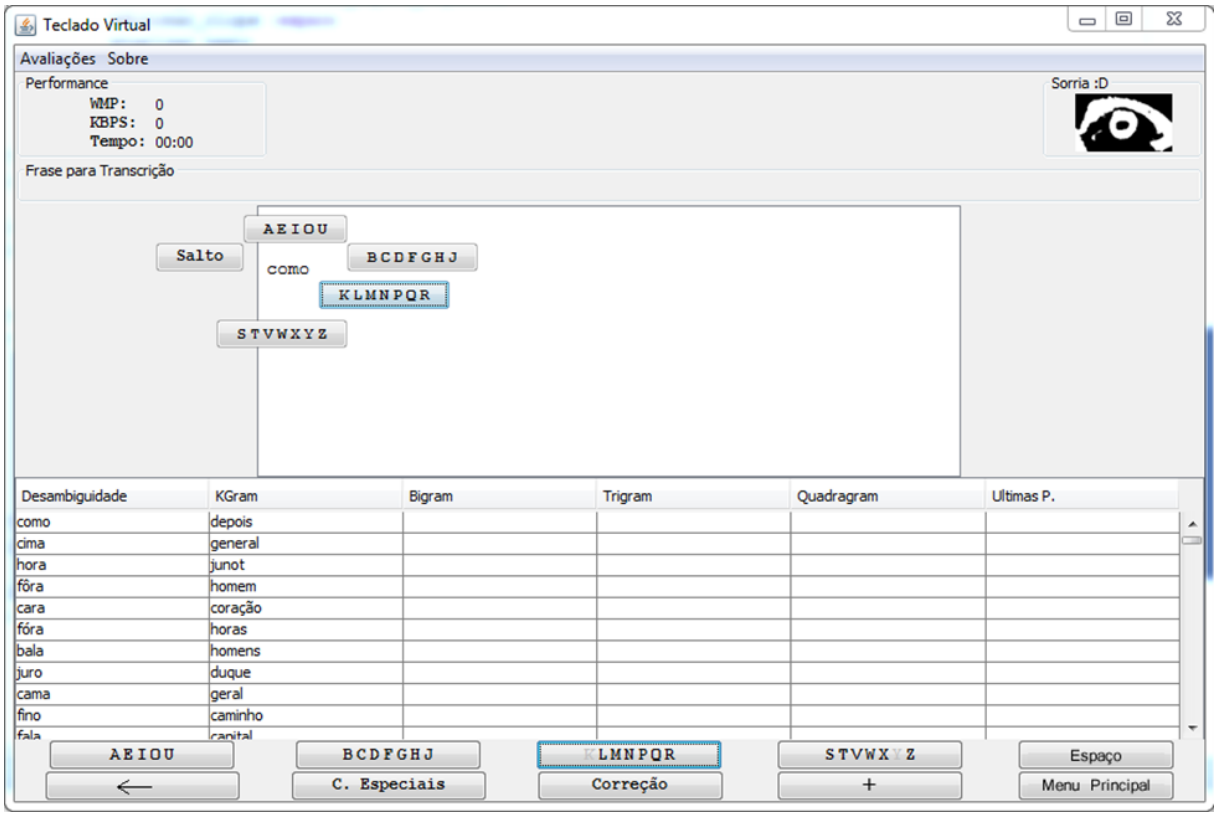
A lei de Fitts determina que o tamanho das teclas e a distância entre elas influência diretamente na performance de digitação. Entretanto, esta regra se aplica a teclados de navegação manual. A distância entre as teclas ou suas dimensões não são relevantes para a construção de teclados que utilizam métodos de varredura. Isso acontece porque a navegação entre as teclas é realizada pelo método de varredura e não pelo próprio usuário. Neste caso é mais importante diminuir o tempo do ciclo de varredura, porque esse processo consome muito tempo de digitação (MIRó-BORRáS et al., 2009).

Como o tamanho das teclas e a distância entre elas não é importante para o teclado virtual assistivo desenvolvido neste trabalho decidiu-se disponibilizar uma opção de configuração na qual o usuário pode escolher o tamanho das teclas e das letras. O objetivo dessa estratégia é aumentar a acessibilidade do teclado para pessoas com deficiência visual.

5.2.6 Disposição das teclas

Segundo a lei de Hick-Hyman (HICK, 1952; HYMAN, 1953), o tempo de reação do usuário é diretamente proporcional ao número de objetos na interface. Assim, para reduzir o tempo de reação é necessário diminuir o número de teclas (SHARMA et al., 2012). Porém, os *designers* de interface alegam que um teclado deve conter todas as teclas necessárias para compor um texto (SHARMA et al., 2012). Com o objetivo de superar esse desafio, este trabalho propõe um *layout* que mescla a disposição das teclas em matriz e círculo. A Figura 5.5 mostra esse *layout* misto.

Figura 5.5 – *Layout* misto do teclado virtual assistivo desenvolvido nesta tese. O teclado circular envolve a palavra que está sendo escrita. O teclado matricial fica logo abaixo das listas de sugestão de palavras.



A distância entre a área de escrita do texto e as teclas do teclado virtual deve ser mínima, pois para digitar o usuário deve alternar o olhar entre os dois lugares (SHARMA et al., 2010). O objetivo do teclado circular é minimizar essa distância. O *layout* circular mostra apenas o grupo de teclas que está sendo utilizado pelo método de varredura. O círculo formado pelas teclas fica sempre em volta da letra que está sendo inserida.

A disposição das teclas em forma de matriz mostra todas as teclas necessárias para compor um texto (SHARMA et al., 2012). Essa abordagem foi adotada neste trabalho com a

finalidade de diminuir a curva de aprendizado para a utilização do teclado. Além disso, ela serve de guia para mostrar ao usuário todas as opções do teclado.

5.2.7 *Feedback* do teclado

Os usuários digitam mais rápido se possuírem os dois tipos de *feedback* visual e auditivo (MAJARANTA et al., 2003). Assim, nesta tese o método de *feedback* do teclado assistivo é realizado de maneira sonora e de forma visual. A cada tecla digitada o som do pressionar da tecla é emitido pelo *software* e a tecla selecionada é destacada das demais teclas. O *feedback* pseudo tátil não é utilizado, pois o sistema foi construído para funcionar em computadores pessoais.

5.3 Técnicas de Otimização

Para minimizar o esforço de digitação e aumentar a performance de entrada de dados, vários princípios são utilizados em conjunto como: métodos de varredura, predição de letras, predição de palavras e teclados ambíguos (POLácEK; MíKOVEC; SLAVíK, 2012). O objetivo desta sessão é apresentar os métodos de predição de letras e palavras utilizados no teclado virtual proposto.

5.3.1 Dicionário

Antes de descrever os métodos de predição é necessário relatar como foi construído e estruturado o dicionário utilizado pelo teclado virtual proposto. O léxico é responsável por fornecer uma base de palavras para os métodos de predição e desambíguidade. Pois nele, além das características dessas palavras estão todos os termos que poderão ser utilizados pelo usuário.

5.3.1.1 Construção

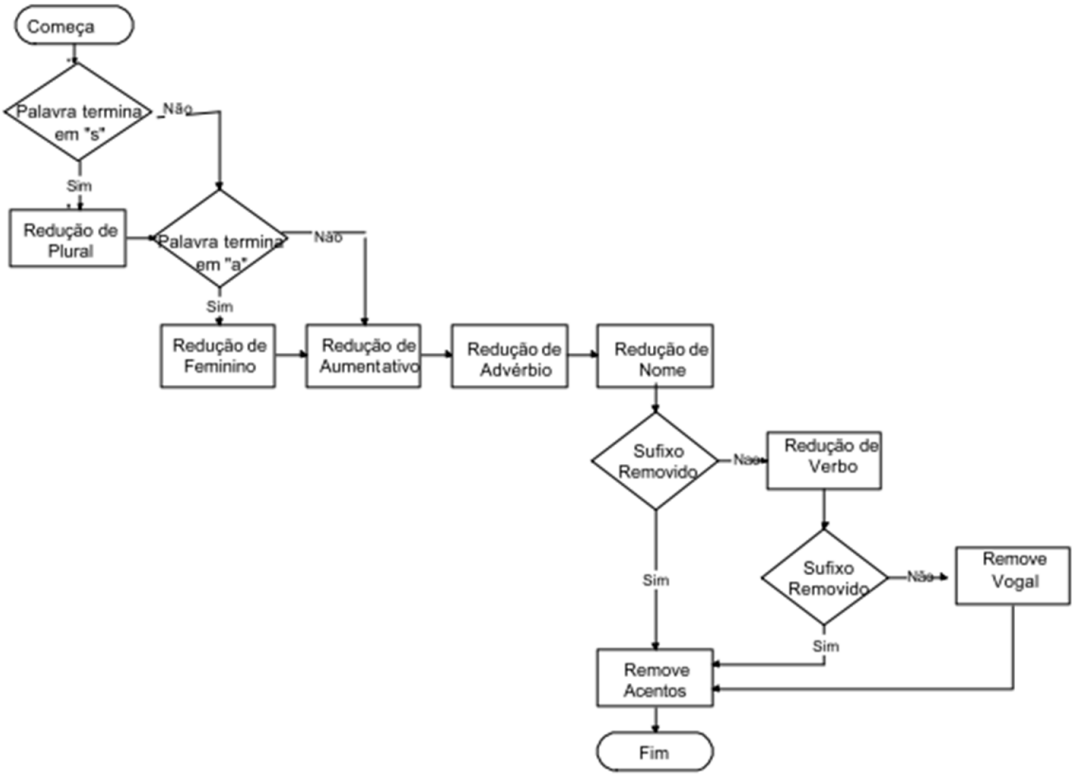
Para construir o dicionário foi utilizado um corpus da língua portuguesa. Esse corpus e a frequência das palavras contidas nele estão disponíveis em (SARDINHA; FILHO; ALAMBERT, 2014). Esse apanhado de textos possui mais de quatro milhões de palavras das quais 170 mil são distintas. Entretanto, muitas dessas palavras possuem erros gráficos ou são palavras de origem inglesa. Para minimizar esse problema foi realizado um processo de remoção dessas palavras.

A princípio para filtrar os termos com erros de grafia ou palavras estrangeiras bastaria verificar em um dicionário em português se a palavra a ser excluída estava presente. Se essa palavra não estivesse contida no dicionário ela poderia ser removida. Entretanto, a língua portuguesa possui um grau de flexão moderado o que torna o processo de filtragem de palavras mais complexo (GARAY-VITORIA; ABASCAL, 2005). Uma linguagem é flexionada quando é possível produzir formas morfológicas a partir de uma raiz ou lema. Portanto, não seria suficiente verificar no dicionário se a palavra existia, pois os dicionários não possuem todas as flexões de cada raiz.

Devido a característica de flexão da língua portuguesa foi necessário executar um processo de redução de palavras denominado *stemming*. O objetivo desse processo é extrair o radical de uma palavra. O método de *stemming* utilizado neste trabalho foi proposto por (ORENGO; HUYCK, 2001) e é conhecido por Removedor de Sufíxos para a língua portuguesa (RSLP). A Figura 5.6 ilustra o funcionamento do RSLP.

Figura 5.6 – Funcionamento do algoritmo RSLP apresentado por Lopes (2004).

Fonte: (LOPES, 2004)



O processo do RSLP é composto de diversas regras. A cada passo uma regra é aplicada de acordo com algumas condições. Ao aplicar uma regra do processo, uma parte do sufixo da palavra é removida. O sufixo mais longo é retirado primeiro e assim que o algoritmo é processado os demais sufixos são extraídos. Quando o processo chega ao fim apenas a raiz da palavra permanece.

Após codificar o RSLP aplicou-se o processo de *stemming* a todas as palavras do léxico. Logo, as raízes obtidas a partir desse processo foram comparadas com as raízes das palavras de dois dicionários em português, o Dicionário Aberto (DICIONARIO-ABERTO.NET, 2015) e Michaeles (MICHAELIS.UOL.COM.BR, 2015).

Se a raiz da palavra pesquisada fosse encontrada em pelo menos um desses dicionários a palavra era mantida, caso contrário ela era removida. Das 170 mil palavras ficaram apenas 100 mil. As palavras restantes formaram o léxico utilizado pelo sistema deste trabalho.

5.3.1.2 Armazenamento

O armazenamento do dicionário foi realizado de duas formas: em lista e em árvore. A forma de lista foi utilizada para armazenar as palavras de forma persistente. Portanto, utilizou-se um banco de dados para guardar as palavras e suas características como frequência de ocorrência, significado e tipo gramatical. A estrutura de árvore foi usada para armazenar as palavras durante a utilização do sistema para realizar a predição de texto e a desambiguidade dos termos.

A forma de armazenamento estruturada em árvore foi desenvolvida de acordo com (SHANG; MERRETTAL, 1996). Esse tipo de armazenamento permite a compactação do dicionário em até 50% e diminui o tempo de busca por palavras.

A estrutura em árvore é composta de nós e arestas. Os nós representam as letras e as arestas os vínculos entre as letras. Cada nó armazena uma letra, os apontadores para as próximas letras e seu peso de utilização.

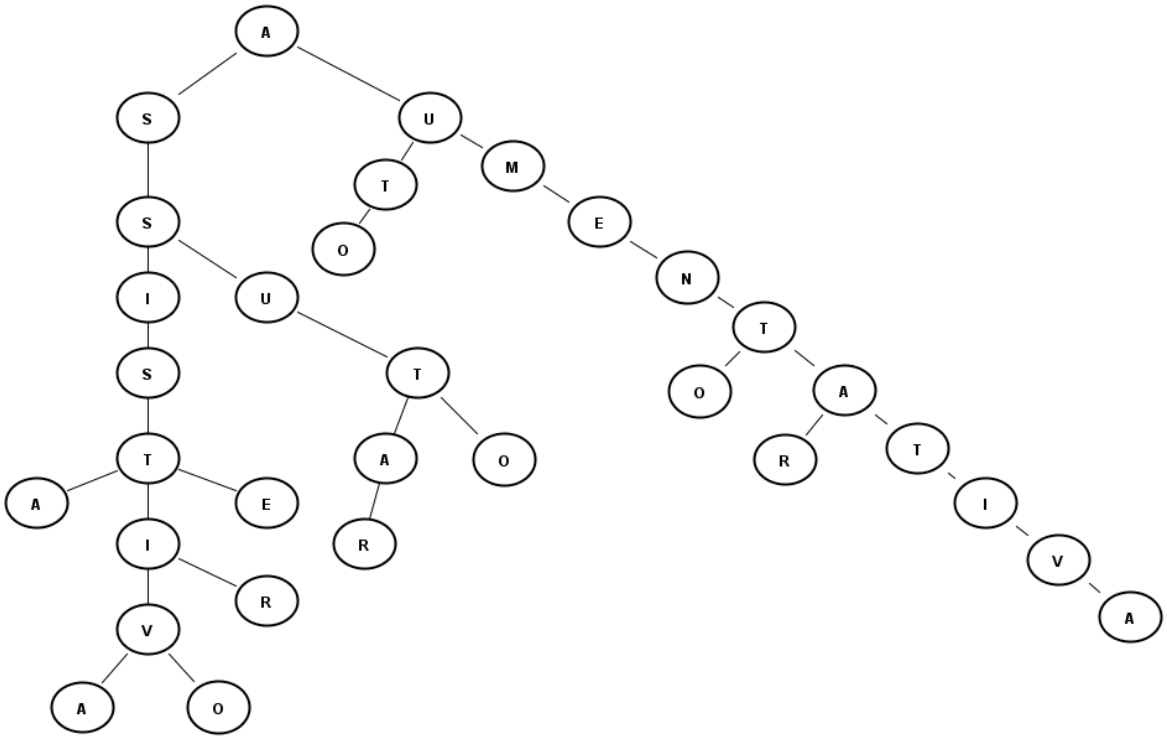
O peso de utilização representa o quanto determinada combinação de letras é utilizada pelo usuário. Por exemplo, considerando duas palavras “assistiva” e “assistivo”, supondo que a palavra assistiva tenha frequência de ocorrência igual 10 e a palavra assistivo tenha frequência de ocorrência igual a 20. O peso de utilização dos nós de “a” até “v” será igual a 30, enquanto os nós “a” a “o” terão pesos iguais a 10 e 20, respectivamente. Esses pesos são utilizados na predição de letras. Os nós que armazenam a última letra de cada palavra são marcados como finais e guardam a frequência de uso da palavra.

A Figura 5.7 ilustra a representação de um dicionário com as palavras: assistiva, assistivo, assistir, assista, assiste, assunto, assuntar, aumentativa, aumento, aumentar e auto.

5.3.2 Técnicas de predição de texto

A predição de texto é uma das características mais utilizadas para aprimorar a performance de digitação dos teclados virtuais (SHARMA et al., 2010). Nas três revisões realizadas para desenvolver esta tese não foi encontrada nenhuma técnica de predição que fosse ideal ou

Figura 5.7 – Representação do dicionário em árvore. Nessa árvore estão representadas as palavras: assistiva, assistivo, assistir, assista, assiste, assunto, assuntar, aumentativa, aumento, aumentar e auto.



muito superior a outra. Portanto, resolveu-se adicionar diversas técnicas de predição ao teclado virtual proposto neste trabalho.

5.3.2.1 Predição de palavras

Com a finalidade de otimizar a técnica de predição, o teclado proposto apresenta cinco listas de predição de palavras. Cada lista utiliza um método de predição diferente. Os métodos implementados no teclado são: k-gram, bigram, trigram, quadragram e regência de uso. Esses métodos apresentam uma lista de palavras sugeridas. Todas as listas são ordenadas primeiro pelas palavras mais utilizadas pelo usuário e depois pelos termos com maior frequência de ocorrência no corpus da língua. No decorrer da utilização do teclado a base de dados de cada método é atualizada de acordo com os textos digitados pelo paciente.

O método k-gram utiliza o dicionário e as frequências das palavras mais digitadas para autocompletar o termo que está sendo escrito. Esse método torna possível diminuir o esforço de digitação, pois assim que o usuário insere as primeiras sequências de letras de uma palavra é possível completar o termo que está sendo escrito.

Os métodos bigram, trigram e quadragram possibilitam a sugestão de palavras antes mesmo do usuário iniciar a digitação. Esses métodos são eficientes para diminuir o esforço de

digitação (GARAY-VITORIA; ABASCAL, 2005). Para realizar essa predição é necessário implementar listas de frequência de ocorrência das palavras em sequência. Por exemplo, considerando três termos distintos "comunicação", "suplementar" e "alternativa". Para determinar qual palavra deve ser sugerida se a termo "comunicação" for escrito é necessário verificar quantas vezes os termos "suplementar" e "alternativa" ocorrem precedidos da palavra "comunicação".

Para construir as três listas de sugestões foi desenvolvido um programa que percorreu todas as palavras do corpus da linguagem. Esse *software* construiu as três listas de sugestões analisando a precedência das palavras. Foram necessárias mais de 48 horas de processamento para percorrer as mais de quatro milhões de palavras presentes no corpus da língua.

Finalmente, a lista de "Regência de Uso" apresenta as palavras que foram utilizadas recentemente pelo usuário. Todas as palavras que o usuário insere no sistema são apresentadas sequencialmente nesta lista.

Para selecionar qualquer palavra da lista de sugestões, basta que o usuário selecione a tabela de predição. O sistema inicia uma varredura entre as listas de predição. O usuário seleciona a lista e depois pode selecionar a palavra desejada por meio da varredura de linha. A palavra é inserida no texto com a adição de um caractere de espaço. De acordo com a configuração do teclado, o número de palavras disponíveis em cada lista poderá variar entre 5 a 15 termos.

Como o teclado virtual proposto implementa o método k-gram para completar a palavra concluiu-se que não seria produtivo implementar um método para predizer letras. Os métodos de predição de letras serão utilizados para diminuir a ambiguidade do teclado virtual.

5.3.3 Método de desambiguidade

Como método de desambiguidade optou-se por utilizar algoritmos de desambiguidade. A finalidade dessa escolha é minimizar a interação entre o usuário e o teclado virtual. Além disso, os teclados que usam algoritmos de desambiguidade tem uma performance de digitação melhor do que os teclados que utilizam multi-toque (MOLINA; RIVERA; GóMEZ, 2009).

A performance de digitação é mais eficiente usando algoritmos, pois os métodos multi-toque exigem um quantidade maior de interação com o usuário para realizar a desambiguidade tecla a tecla (TANAKA-ISHII; INUTSUKA; TAKEICHI, 2002b). Isso acontece porque o usuário precisa pressionar mais de uma tecla para obter a letra desejada. Esse processo de repetição aumenta o esforço realizado pelo usuário, além do número de interações entre ele e o sistema. Por outro lado, os algoritmos de desambiguidade permitem que o usuário selecione a tecla desejada apenas uma vez, diminuindo o esforço e a interação com o sistema.

Outro fator que levou a adoção dos algoritmos de desambiguidade é a diferença entre as línguas inglesa e portuguesa. Ao contrário do inglês, o português possui vogais que muitas

vezes são acentuadas. Adicionar mais teclas para acentuar as palavras aumentaria o esforço do usuário e provavelmente diminuiria sua capacidade de digitação. O método de desambiguidade pode minimizar esse problema identificando as palavras que são acentuadas e acentuando-as corretamente.

Essa otimização é possível porque o usuário está restrito a digitar apenas palavras que estejam cadastradas no dicionário do sistema de CAA. O método de desambiguidade não diferencia vogais acentuadas e sem acentuação. Portanto, mesmo que o paciente não saiba acentuar as palavras corretamente o algoritmo o auxiliará.

Durante o processo de desambiguidade das palavras o dicionário pode ser ajustado para o vocabulário do usuário com o objetivo de reduzir a lista de desambiguidade e sugerir as palavras mais utilizadas pelo paciente (MIRó-BORRáS et al., 2009). Assim, o usuário pode inserir novas palavras usando um teclado não ambíguo.

5.3.3.1 Algoritmo de desambiguidade de palavras

Para realizar a desambiguidade das palavras é utilizado o método TNK (MOLINA; RIVERA; GóMEZ, 2009). Esse método foi escolhido pois ele independe da quantidade de teclas do teclado virtual. A princípio não foi encontrado nenhum algoritmo que implementasse esse método. Portanto, foi preciso codificar o TNK.

A codificação desse método foi dividida em três etapas, cada uma foi codificada por uma função. A primeira etapa decodifica a sequência digitada pelo usuário. Essa etapa é implementada pela função decodificação 1. A segunda fase produz uma sub-árvore do léxico correspondente a sequência decodificada. Para implementar essa etapa foi desenvolvida a função montarArvoreSugestao 2. A terceira etapa verifica as possíveis palavras existentes nesta sub-árvore. Para essa fase foi implementada a função montarListaPalavras 3.

A função de decodificação recebe o teclado e a sequência de caracteres que foi digitada. Essa sequência pode conter três tipos de dados numéricos, uma estrutura composta simples ou um literal. Os dados numéricos estão relacionados ao número da tecla do teclado virtual. A estrutura composta representa um vetor de caracteres e finalmente o dado literal é uma única letra.

Quando o dado é do tipo numérico ele é convertido na sequência de letras que corresponde a tecla que foi digitada. Por exemplo, considerando um teclado que tenha a distribuição de teclas em ordem alfabética, como o teclado ilustrado na Figura 5.8. Se o usuário digitasse a primeira tecla as letras correspondentes iriam de “a” a “g”. Se a segunda tecla fosse escolhida as letras relacionadas a essa tecla estariam entre o intervalo de “h” a “n”, e assim para as demais teclas. Assim, que o número é convertido em uma sequência de caracteres, o resultado da

conversão é adicionado ao vetor da sequência.

Figura 5.8 – Teclado com quatro teclas no qual as letras estão distribuidas entre as teclas em ordem alfabética.

Se o valor da sequência é uma estrutura composta, então não é necessário nenhuma conversão e o dado é adicionado ao vetor da sequência decodificada. Entretanto, se o dado é um literal ele é transformado em uma estrutura composta e adicionado ao vetor da sequência.

Essa conversão é realizada com todos os elementos da estrutura composta *Sdec*. Quando o algoritmo termina é retornado o vetor de sequência de letras.

O vetor de saída da função decodificação é utilizado como parâmetro de entrada da função montarArvoreSugestao. Além disso, essa função recebe como entrada uma variável para construir o conjunto de sub-árvores do dicionário e o léxico do sistema. Essa função inicia atribuindo a primeira sequência de caracteres da sequência decodificada a uma variável denominada *elementos*.

Algoritmo 1: ALGORITMO DE DECODIFICAÇÃO DA SEQUÊNCIA

Entrada: sequencia codificada de caracteres, teclado que codificou a sequência
Saída: Um vetor com a sequência de elementos decodificada

```
1  função decodificao(S,T) é
2      Sdec ← vetor vazio
3      para i ← 1 até S_tamanho faça
4          e ← S_i
5          se e_tipo = "numero" então
6              Sdec_i
               ← um vetor de letras correspondente a tecla da posição i no teclado T
7          senão se e_tipo = "vetor" então
8              Sdec_i ← e
9          senão
10             Sdec_i ← um vetor com a tecla e
11         fim
12     fim para
13     retorna Sdec
14 fim função
```

Para cada caractere *c* contido no vetor de *elementos* é verificado se existe algum nó raiz que tenha uma letra igual a ele. Se existe um nó raiz com esse caractere, então é gerado um nó na árvore *Adec*. Esse nó recebe o caractere pesquisado e todos os seus atributos. Finalmente, novamente é chamada a função montarArvoreSugestao. Nessa chamada é passado como parâ-

metro a sequência de caracteres menos o primeiro elemento, a sub-árvore que está sendo gerada e finalmente a sub-árvore do léxico que está em processo de replicação.

Ao final de todas as chamadas recursivas, a variável *Adec* contém um conjunto de sub-árvores do dicionário do sistema.

Finalmente, para produzir a lista de desambiguidade é executada a função montarListaPalavras. Essa função recebe como parâmetro de entrada a saída da função montarArvoreSugestao *Adec*, assim como, uma lista vázia P e uma palavra vázia p. O algoritmo dessa função percorre os nós da sub-árvore *Adec* concatenando as letras que estão registradas nos nós dessa árvore.

Assim que o algoritmo encontra um nó que possuia o atributo fim com valor igual a *verdadeiro* a palavra é adicionada ao vetor P. Ao final do algoritmo a lista P está preenchida com todas as palavras existentes nas sub-árvores de *Adec*.

Algoritmo 2: ALGORITMO DE CRIAÇÃO DAS SUBARVORES EQUIVALENTES

Entrada: sequência decodificada de caracteres, árvore decodificada e dicionário
Saída: Uma árvore de sugestões com as subarvores do léxico equivalentes a sequência decodificada

```
1  função montarArvoreSugestao(Sdec,Adec,L) é
2      se Sdec ≠ ∅ então
3          elementos ←Sdec_1
4          para cada c ∈ elementos faça
5              se c ∈ nosRaiz(L) então
                   /* Adiciona um nó a árvore Adec com caractere c e
                      seus outros atributos                          */
6                  montarArvoreSugestao(Sdec[2..tamanho],Adec → c,L → c))
7              fim
8          fim para
9      senão
10         retorna Adec
11     fim
12 fim função
```

A lista de sugestão é ordenada de acordo com a frequência de uso do usuário. Se a palavra nunca foi utilizada pelo paciente o termo é ordenado pela frequência de ocorrência da palavra no corpus da língua. O algoritmo que realiza todo o processo de desambiguidade é apresentado em 4.

Algoritmo 3: ALGORITMO DE CRIAÇÃO DA LISTA DE PALAVRAS DE DESAMBIGUIDADE

Entrada: Subarvore montada a partir da função montarArvoreSugestao, lista de sugestão de palavras que será montada e palavra que está sendo montada

Saída: Uma árvore de sugestões com as subarvores do léxico equivalentes a sequência decodificada

```
1  função montarListaPalavras(Adec,P, p) é
       /* A função nosRaiz retorna todos os nós que são raiz em L.
          */
2      nos ←nosRaiz(Adec)
3      para cada no ∈ nos faça
4          p ← concatenar(p,no.letra)
5          se no.fim = verdadeiro então
6              A palavra p é adcionada a lista P
7          fim
8          montarListaPalavras(Adec → no,P, p)
9      fim para
10     retorna P
11 fim função
```

Algoritmo 4: ALGORITMO DE DESAMBIGUIDADE

Entrada: Sequência digitada pelo usuário, teclado que originou a sequência e o léxico do sistema

Saída: Lista de desambiguidade

```
1  função realizarDesambiguidade(S,T,L) é
2      Sdec ← decodifio(S,T)
3      Adec ← arvore vazia
4      montarArvoreSugestao(Sdec,Adec,L)
5      P ← lista vazia
6      montarListaPalavras(Adec,P, p)
7      P ← orderLista(P)
8      retorna P
9  fim função
```

O método de desambiguidade implementado nesta tese, além de ser independente da quantidade de teclas também não depende da distribuição das letras. Isso porque a decodificação da sequência pode ser realizada com um teclado ou apenas com as sequências das letras desejadas.

5.4 Algoritmo de Desambiguidade das Letras

O teclado virtual proposto nesta tese pode proporcionar um alto grau de ambiguidade entre as palavras. O nível de ambiguidade do teclado virtual está diretamente relacionado ao número de teclas, a distribuição das letras entre essas teclas e ao léxico do sistema. Um teclado que possuia um número elevado de colisões entre as palavras pode diminuir a performance de digitação do usuário.

O sistema proposto nesta tese procura minimizar esse problema propondo um algoritmo de desambiguidade por letra. Esse método foi denominado de desambiguidade parcial, pois realiza a desambiguidade de apenas uma das teclas da sequência da palavra.

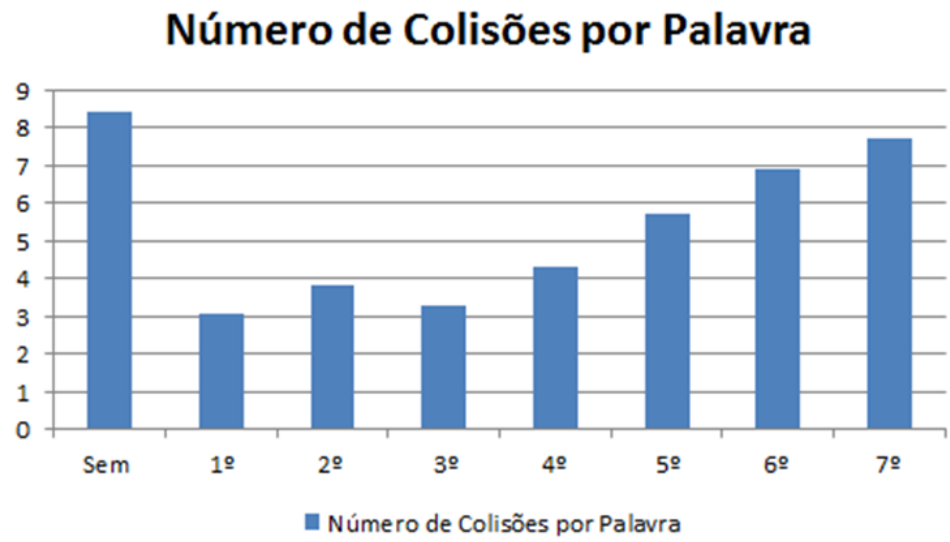
A finalidade principal dessa técnica é realizar a desambiguidade de uma tecla com o objetivo de diminuir o número de colisões entre as palavras. Por exemplo, supondo que o usuário deseja escrever a palavra "você" usando o teclado alfabético ilustrado na Figura 5.8. Para digitar esse pronome, o paciente precisa selecionar a sequência de teclas 4-3-1-1 e o número de colisões para essa sequência é 15. Entretanto, se o usuário digitasse a mesma sequência, mas realizasse a desambiguidade da segunda tecla, a sequência digitada seria 4-o-1-1. Nesse caso, o número de colisões diminuiria para três.

Para verificar a eficácia do método foi realizado um pequeno experimento. Nesse experimento foi calculada a quantidade de colisões entre as palavras se uma das teclas do termo digitado não possuísse ambiguidade. Para realizar esse experimento utilizou-se o léxico desenvolvido para o sistema e os teclados virtuais com quatro teclas. Esses teclados possuíam sete letras distribuídas entre as duas primeiras teclas e seis letras organizadas entre as demais teclas.

Foram gerados 1000 teclados com uma ordenação aleatória entre as letras. Para cada teclado gerado foi verificado qual o número de colisões por palavra que o léxico possuía. Esse teste foi realizado com todas as teclas ambíguas, depois foi executada a desambiguidade de uma única tecla da sequência da palavra. O processo de desambiguidade foi aplicado entre a primeira e a sétima tecla da sequência do termo.

Após os testes calculou-se a média aritmética do número de colisões por palavra pela quantidade de teclados. O Figura 5.9 mostra o resultado desse experimento. Pode-se verificar nessa figura que se a primeira, a segunda ou a terceira tecla não possuir ambiguidade o número de colisões diminuiu em mais de 100%.

Figura 5.9 – Gráfico do número de colisões por palavra. Verifica-se que o número de colisões diminui consideravelmente se não existir ambiguidade nas três primeiras teclas.



Após os testes realizados, pôde-se concluir que durante a digitação do usuário é válido realizar a desambiguidade nas três primeiras teclas. Entretanto, executar o método de desambiguidade utilizando a técnica de multi-toque consumiria tempo e poderia diminuir a performance de digitação. Assim, para executar a desambiguidade parcial foi desenvolvido um algoritmo de sugestão de letras 5.

O algoritmo de sugestão desenvolvido neste trabalho calcula qual a letra da tecla selecionada pelo usuário possui maior probabilidade de ser o caractere que ele deseja. Para realizar essa predição, primeiro é decodificada a sequência digitada pelo usuário usando a função 1. A partir dessa sequência é calculado o peso de sugestão de cada letra. Esse cálculo é realizado somando os pesos de uso de cada sequência até o último nó folha.

Por exemplo, considerando um subconjunto do léxico do sistema composto de cinco palavras: tecnologia, técnico, teclado, tecla e técnica. Essas palavras teriam as frequências 10, 12, 5, 13 e 8, respectivamente. Supondo que o usuário esta utilizando o teclado mostrado na Figura 5.8 e digite a palavra “tecnologia”. A sequência correspondente a palavra é 3-1-1-3. A Figura 5.10 Ilustra as sub-árvores relacionadas a essa sequência.

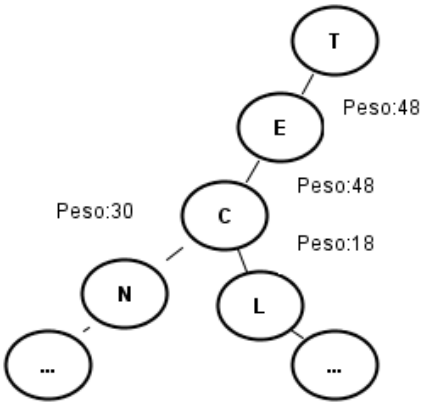
Neste caso, o somatório dos pesos da letra “T” a letra “N” é igual a 126, enquanto que o somatório de “T” a “L” equivale a 114. Portanto, a letra sugerida para o usuário é a letra “N”. A Figura 5.11 mostra o funcionamento do método de desambiguidade parcial.

Algoritmo 5: ALGORITMO DE DESAMBIGUIDADE PARCIAL

Entrada: sequência decodificada de caracteres, árvore decodificada, dicionário do sistema
Saída: Lista de desambiguidade

```
1  função desambiguidadeParcial(Sdec, L, listaPesos ,peso) é
       /* Verifica se não é o último nó da sequência                    */
2      se Sdec_2 ≠ ∅ então
3          elementos ←Sdec_1
4          para cada c ∈ elementos faça
5              se c ∈ nosRaiz(L) então
                   /* A função noRaizCaractere retorna o nó
                      correspondente ao caractere c em L.                */
6                  no ← noRaizCaractere(L,c)
7                  peso ← peso + no.peso
8                  desambiguidadeParcial(Sdec[2..tamanho],L → c, listaPesos,peso)
9              fim
10         fim para
11     senão
12         elementos ←Sdec_1
13         para cada c ∈ elementos faça
14             se c ∈ nosRaiz(L) então
15                 no ← noRaizCaractere(L,c)
                   /* Soma o peso do caractere c ao seu peso geral e
                      atualiza a listaPesos na posição que está o
                      caractere c.                                       */
16                 listaPesos_c ← peso + no.peso + listaPesos_c
17             fim
18         fim para
19     fim
20     retorna listaPesos
21 fim função
```

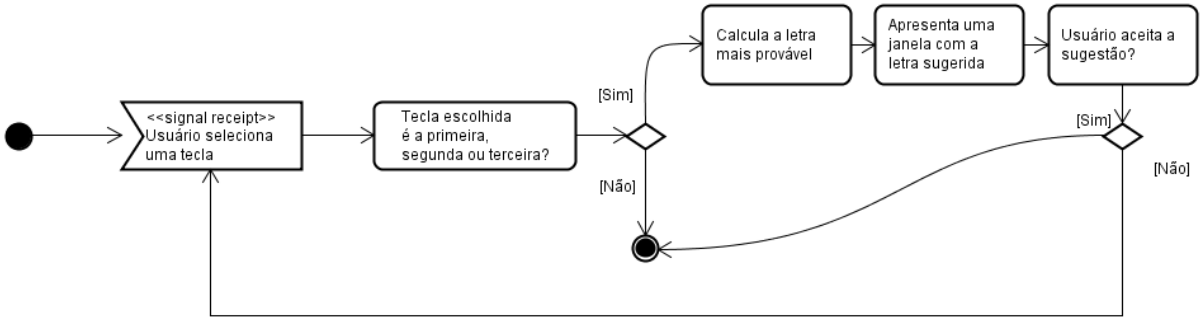
Figura 5.10 – Sub-árvore para sugestão da letra



O método de desambiguidade parcial é executado sempre que o usuário seleciona a

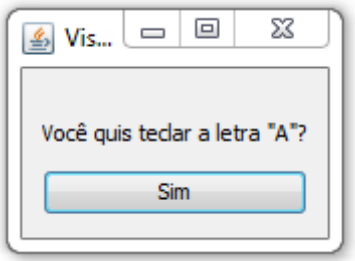
segunda, a terceira ou a quarta tecla da palavra. Assim que uma dessas teclas é escolhida a janela ilustrada na Figura 5.12 é mostrada ao usuário.

Figura 5.11 – Método de desambiguidade parcial.



Na janela mostrada na Figura 5.12, o usuário pode escolher se aceita a sugestão do sistema, ou pode aguardar até que a sugestão desapareça. Se o usuário optar por aceitar a recomendação do sistema, a letra sugerida substitui a tecla selecionada. Caso o paciente não aceite a sugestão, a tecla selecionada permanece na sequência de teclas digitadas.

Figura 5.12 – Janela de desambiguidade.



Assim que uma sugestão de letra é aceita ou se a tecla digitada não está entre as três primeiras teclas, o sistema para de executar o método de desambiguidade parcial. Entretanto, se o usuário não aceita a letra sugerida e a próxima tecla selecionada é a terceira ou a quarta, o método de desambiguidade parcial realiza o processo de desambiguidade da tecla novamente.

O método de desambiguidade de tecla proposto nesta tese diminui o grau de ambiguidade do sistema. Além disso, esse método pode trazer a palavra desejada mais próxima das primeiras sugestões. Isso acontece porque o número de palavras na lista de sugestão diminui de acordo com a desambiguidade da letra.

5.5 Metodologia Evolutiva de Teclados Virtuais

Teclados que alteram o posicionamento das letras entre as teclas durante a digitação não possuem uma boa aceitação pelos usuários (POLácEK; MíKOVEC; SLAVíK, 2012; TOPAL;

BENLIGIRAY, 2012; POUPLIN et al., 2014). A alteração da ordem das letras deixa o usuário confuso e dificulta a localização do caractere desejado (POLácEK; MíKOVEC; SLAVíK, 2012). Entretanto, a longo prazo, desenvolver teclados que se adaptem ao vocabulário e a maneira de escrita do usuário pode aumentar a performance de digitação. Isso porque, a maneira que as letras são distribuídas entre as teclas influência diretamente no esforço e na velocidade de digitação (LESHER; MOULTON, 2000). O objetivo da metodologia evolutiva de teclados virtuais proposta nesta tese é solucionar esse problema.

Para desenvolver esse método foi definido de acordo com a literatura o modelo de teclado assistivo mais indicado para pacientes com SE. Logo, foi determinado qual o algoritmo de otimização seria utilizado para aprimorar o *layout* do teclado. Em seguida, elaborou-se a metodologia evolutiva de teclados virtuais assistivos.

5.5.1 Escolha do algoritmo de otimização

O algoritmo genético foi escolhido para ser utilizado no processo de otimização do teclado assistivo. Quatro fatores contribuíram para que essa meta-heurística fosse adotada: (1) o problema que será resolvido é discreto; (2) os resultados dos trabalhos correlatos; (3) a capacidade de aceitar múltiplas funções objetivo; e (4) a facilidade em modelar o problema da distribuição das letras entre as teclas.

Uma abordagem para otimizar a distribuição das letras entre as teclas é a utilização de algoritmo genético. Essa técnica tem apresentado bons resultados tornando-se uma alternativa válida para solucionar parcialmente o problema de otimização do teclado (LEVINE; TREPAGNIER, 1990; BROUILLETTE; SHARMA; KALITA, 2012; GOETTL; BRUGH; JULSTROM, 2005; RAYNAL; VIGOUROUX, 2005).

A solução do problema da distribuição das letras entre as teclas é discreta. As soluções para esse tipo de problema são representadas por um vetor de valores discretos. Esses valores são decodificados em uma configuração de um teclado virtual. Os algoritmos genéticos podem chegar a boas soluções nesse tipo de problema, porque a solução apresentada por esse algoritmo não é uma solução aproximada como ocorre nos problemas de variáveis continuas.

Os algoritmos genéticos possuem algoritmos que permitem a otimização multi-objetivo. Alguns exemplos de algoritmos genéticos multi-objetivo são NSGA-II, NSGA-III (DEB; JAIN, 2014) e SPEA2. Esses algoritmos fornecem um conjunto de possíveis soluções, denominadas grupo de soluções Pareto-ótimas. Neste estudo de caso, as funções objetivo são: o cálculo do esforço de varredura e o número de colisões. Essas duas funções podem ser conflitantes.

O problema da distribuição de letras por teclas foi modelado utilizando a representação cromossômica por permutação. Os indivíduos da população possuem 26 genes. Os alelos desses

genes são números inteiros que variam de zero a 25. Cada número inteiro define uma letra do alfabeto. Deste modo, o zero representa a letra "a", o número um representa a letra "b" e assim por diante.

A população inicial foi gerada com 1000 indivíduos e foram realizados 9000 ciclos de evolução. O objetivo da quantidade moderada de indivíduos iniciais é aumentar o espaço de busca do algoritmo. Enquanto, a finalidade dos 9000 ciclos é aprimorar consideravelmente a solução. Executando esse algoritmo em um máquina com 8 *gigas* de memória primária e um processador de intel *core* i7 foram necessários 30 minutos de processamento para concluir a otimização.

Na etapa de seleção dos indivíduos para o cruzamento utilizou-se o método de seleção por torneio e no cruzamento foi usado o operador PMX. Esse operador é específico para permutações dos genes. A probabilidade de cruzamento aplicada é de 100% para garantir que a cada ciclo seja gerado uma nova solução candidata. Na mutação, utilizou-se o operador *Swap* com a probabilidade de 50% para minimizar as chances das soluções convergirem prematuramente para mínimos locais.

Para a implementação do algoritmo genético foi utilizado o *Multi-Objective Evolutionary Algortithms* (MOEA) framework (HADKA, 2015). Esse framework é um projeto de código fonte aberto e gratuito desenvolvido na linguagem Java. O principal objetivo desse software é auxiliar o desenvolvimento de aplicações que utilizam algoritmos genéticos multi-objetivo. Além do algoritmo genético, ele implementa diversos métodos de otimização como nuvem de partículas, evolução diferencial e programação genética.

Para algoritmos genéticos o MOEA framewok inclui vários tipos de operadores de cruzamento e mutação. As probabilidades de cruzamento, mutação, população inicial e número de ciclos podem ser configurados de acordo com o problema. Além disso, esse *framework* implementa os principais algoritmos genéticos, dentre eles o NSGA-II, NSGA-III e o SPEA2.

Este trabalho utilizou esse framework para agilizar a performance de programação e o processo de otimização. Além disso, adotar um framework garante uma maior robustez no processo. Essa robustez é devido a minimização dos erros de implementação, pois esse framework possui mais de 1100 casos de teste validados.

Existem diversos tipos de algoritmos genéticos, para esta pesquisa optou-se pelo algoritmo "NSGA-III". Esse algoritmo foi proposto recentemente por (DEB; JAIN, 2014). A vantagem do NSGA-III em relação a outros algoritmos é a capacidade de gerenciar problemas com múltiplas funções objetivo. Além disso, ainda não existe nenhuma pesquisa que aplicou esse tipo de algoritmo ao problema de otimização do teclado.

5.5.2 Funções objetivo

Dois critérios foram escolhidos para serem utilizados pelo algoritmo de otimização como funções objetivo. O primeiro está relacionado ao esforço de varredura necessário para digitar uma determinada palavra. O segundo critério está relacionado ao grau de ambiguidade do dicionário. Assim, a função desenvolvida para calcular o primeiro critério considera a quantidade de alterações do foco do teclado entre as teclas necessárias para digitar uma palavra. A segunda função objetivo calcula a quantidade de colisões entre as palavras existentes no léxico.

As funções objetivo foram definidas de acordo com as necessidades dos pacientes que possuem SE. Pois, os dois critérios de otimização influenciam diretamente no esforço e na performance de digitação. Isso porque, quanto maior é o tempo de varredura maior é o esforço e mais tempo o usuário precisa para inserir uma tecla. Da mesma forma, se o número de colisões é muito alto o paciente tem que alternar entre várias palavras para selecionar o termo desejado. Essa seleção pode tomar muito tempo e diminuir a performance de digitação.

Para mostrar as equações das funções objetivo é necessário definir algumas variáveis. Nas próximas equações deve-se considerar:

t − teclado assistivo;

t_i − i-ésima tecla do teclado;

p − palavra;

p_i − i-ésima letra da palavra p;

p_{qdt} − quantidade de letras da palavra;

d − dicionário;

d_i − i-ésima palavra do dicionário; e

d_{qdt} − quantidade de palavras do dicionário.

5.5.2.1 Função objetivo do esforço de varredura

Para calcular o esforço de varredura é necessário primeiro definir a função que determina qual tecla pertence determinada letra. A Equação (5.1) define essa função.

$$f_{pt}(t,p,i) = \underset{n}{arg}\{t_n = p_i\} \tag{5.1}$$

em que: i é a posição da letra na palavra e n representa o número da tecla.

A Equação (5.1) verifica qual tecla determinada letra está e retorna o número dessa tecla. Esse número equivale a quantidade de vezes que o teclado terá que selecionar determinada tecla para atingir a letra. Por exemplo, se o usuário escolheu a letra "A" e essa letra está na primeira posição o teclado terá que selecionar apenas na primeira tecla. Entretanto, se o usuário selecionar a letra "Z" que está na última tecla, o teclado terá que varrer a primeira, a segunda, a terceira até chegar a quarta tecla.

Para definir o esforço necessário para digitar uma única palavra em um determinado teclado é necessário calcular o somatório do esforço de varredura de todas as letras dessa palavra utilizando esse teclado. A Equação (5.2) calcula esse esforço.

$$E_{palavra}(t,p) = \sum_{i=1}^{p_{qdt}} f_{pt}(t,p,i) \tag{5.2}$$

em que: i é o índice da letra da palavra.

O esforço total de varredura de cada palavra deve ser multiplicado pela frequência dessa palavra em um determinado dicionário para definir seu esforço de digitação no dicionário. Essa ponderação é importante para que as palavras mais utilizadas tenham um esforço de varredura menor do que as palavras que são menos utilizadas. A Equação (5.3) calcula a frequência de uma palavra em um determinado dicionário.

$$f_{fr}(p,d) = f_d(p) \tag{5.3}$$

A Equação (5.4) calcula o esforço de varredura de um teclado em um dicionário.

$$E_g(t,d) = \sum_{i=1}^{d_{qdt}} E_{palavra}(d_i,t,d) \times f_{fr}(d_i,d) \tag{5.4}$$

A Equação (5.4) encontra o somatório do esforço de varredura necessário para digitar todas as palavras de um dicionário utilizando um determinado teclado. Essa função calcula o resultado final da primeira função objetivo.

5.5.2.2 Função objetivo do grau de ambiguidade

A segunda função objetivo encontra a porcentagem de palavras que possuem o número de colisões maiores do que cinco. O objetivo de escolher o número cinco é apresentar uma lista de desambiguidade mínima. As listas de desambiguidades com poucos elementos são mais

difíceis de serem construídas. Esse número garante que todos os itens dessa lista poderão ser visualizados pelo usuário sem a necessidade de paginação ou barra de rolagem.

Para calcular o número de colisões é necessário verificar se a combinação de teclas de uma palavra é igual a combinação de teclas de outra palavra. Assim, é importante primeiro definir a equação que determina a sequência de teclas necessárias para definir uma palavra. A Equação (5.5) determina essa sequência.

$$f_{seq}(t,p) = [f_{pt}(t,p,1) \quad \cdots \quad f_{pt}(t,p,p_{qdt})] \tag{5.5}$$

Para verificar se a sequência de uma palavra é igual a sequência de outra palavra é necessário comparar as duas sequências. A Equação (5.6) mostra este cálculo.

$$f_{com}(t,p,p') = \begin{cases} 1, & \text{se } f_{seq}(t,p) = f_{seq}(t,p') \\ 0, & \text{se } f_{seq}(t,p) \neq f_{seq}(t,p') \end{cases} \tag{5.6}$$

em que: p' é a palavra que está sendo analisada.

O número de colisões de uma única palavra em um dicionário pode ser calculado utilizando a Equação (5.7).

$$C_{qdt}(p,t,d) = \sum_{i=1}^{d_{qdt}} f_{com}(t,p,d_i) \tag{5.7}$$

A Equação (5.7) compara a sequência de teclas da palavra pesquisada com todas as sequências dos termos do dicionário. A cada vez que as duas sequências são iguais, o somatório é acrescido de um. Para calcular a quantidade de palavras que possuem o número de colisões maiores do que cinco foi utilizado a Equação (5.8).

$$f_{cont}(p,t,d) = \begin{cases} 1, & \text{se } C_{qdt}(p,t,d) \geq 5 \\ 0, & \text{se } C_{qdt}(p,t,d) < 5 \end{cases} \tag{5.8}$$

Finalmente, para calcular o índice de ambiguidade do teclado basta realizar o somatório da Equação (5.8) de todas as palavras do dicionário. O resultado deve ser dividido pela quantidade de palavras do dicionário multiplicada por 100. Assim é possível obter a porcentagem de

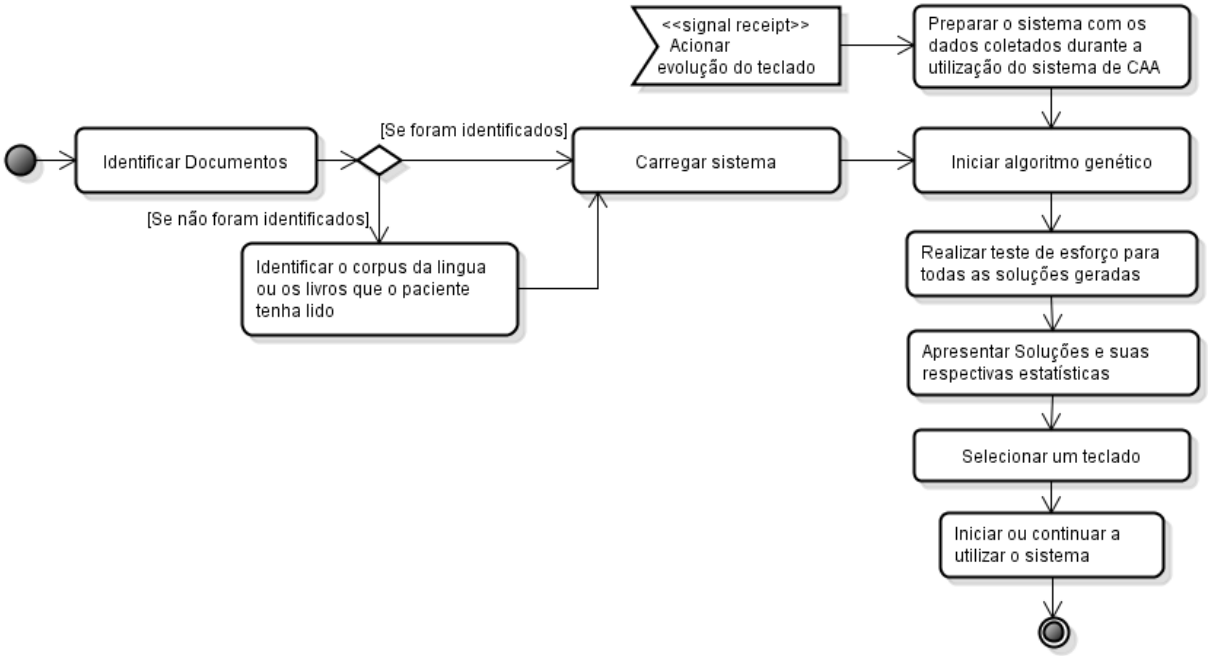
palavras que possuem o número de colisões maiores do que cinco. Esse cálculo é mostrado na Equação (5.9).

$$C_{total}(t,d) = \frac{\sum_{i=1}^{d_{qdt}} f_{cont}(d_i,t,d)}{d_{qdt}} \times 100 \tag{5.9}$$

5.5.3 Processo de evolução do teclado

A Figura 5.13 ilustra o processo de evolução do teclado assistivo.

Figura 5.13 – Processo da métodologia evolutiva de teclados virtuais.



O primeiro passo da metodologia é identificar os documentos que o usuário escreveu. Os documentos coletados são usados como entrada inicial do sistema. Se não for possível encontrar nenhum tipo de registro escrito realizado pelo usuário, o corpus da linguagem ou os livros que o paciente tenha lido podem ser utilizados como a primeira entrada.

Após definir a entrada inicial é executado o algoritmo genético. Esse algoritmo gera várias composições de letras e teclas minimizando o esforço de varredura e o número de colisões entre as palavras. Devido à utilização de mais de uma função mérito, o algoritmo genético gera múltiplas soluções. Cada solução gerada representa um teclado assistivo aprimorado.

Para cada teclado construído é realizado um teste de esforço. Esse teste verifica qual o esforço que o paciente teria para digitar os documentos que compõem a entrada inicial utilizando o teclado gerado. Além de determinar esse valor, o teste também apresenta o esforço

geral do teclado, a porcentagem de palavras que possuem ambiguidade maior do que cinco e a porcentagem de utilização das quatro teclas ao digitar um texto.

Finalmente, a partir das opções de teclado geradas pelo algoritmo e fundamentado pelos dados estatísticos gerados pelos testes é possível escolher um teclado que será utilizado inicialmente. Após a definição do teclado inicial, o usuário poderá usar o sistema de CAA para compor seus próprios textos. Ao digitar novos textos esse sistema armazenará as palavras, as suas respectivas frequências e os textos inseridos pelo usuário.

De acordo com a utilização do sistema, o usuário pode solicitar uma nova otimização do teclado. Assim que a ação de otimização é executada com algumas adequações é iniciada uma nova iteração no ciclo de evolução do teclado.O teclado será otimizado de acordo com o vocabulário que o usuário inseriu durante a utilização do sistema. O teste de esforço será realizado com base em todos os textos que o usuário digitou após a última otimização. A finalidade dessa abordagem é adequar o teclado ao modo de escrita mais atual do usuário.

Além do teste de esforço, é utilizado um algoritmo dissemelhança baseado na distância Euclidiana entre as letras para comparar os teclados otimizados com o teclado assistivo atual. A finalidade desse teste é suavizar a migração entre o teclado atual e as outras abordagens propostas, pois a partir dessa verificação o usuário pode visualizar a similaridade entre os teclados verificando qual teclado pode proporcionar a ele menor curva de aprendizado.

5.5.4 Função de dissemelhança

Usuários que possuem uma certa familiaridade com um determinado tipo de teclado tendem a ter uma performance de digitação melhor do que os usuários iniciantes. Além disso, uma vez que o usuário se sente confortável e acostumado ao *layout* do teclado dificilmente ele altera seu padrão. Esse fato pode ser comprovado facilmente com o exemplo do teclado QWERTY. Isso porque, mesmo com vários outros teclados que podem permitir ao usuário digitar com mais eficiência o modelo QWERTY nunca foi substituido (LEVINE; TREPAGNIER, 1990; LIGHT; ANDERSON, 1993; EGGERS et al., 2003).

Entretanto, para pessoas com SE o mais importante é que o teclado transmita a informação de maneira rápida e com o mínimo de esforço. Para esse tipo de usuário a alteração do *layout* do teclado pode trazer mais vantagens do que desvantagens. Entretanto, a mudança de paradigma requer um tempo de adaptação e muitas vezes pode ser árdua. Para auxiliar na transição entre os teclados virtuais foi desenvolvido um algoritmo que mostra o grau de diferença entre os teclados. Com base nessas informações, o usuário pode escolher o teclado que fornece a ele uma curva de aprendizado mais curta e que seja melhor do que o teclado atual.

A função de dissemelhança calcula a soma do deslocamento das letras entre as teclas.

Por exemplo, considerando os teclados virtuais mostrados nas Figuras 5.8 e 5.14. Para calcular o valor de dissemelhança é verificado quantas teclas cada letra se deslocou. Assim, a letra "A" estava na primeira tecla e passou para a quarta, deslocando três teclas. A letra "Z" estava na última tecla e passou para a terceira, portanto seu deslocamento foi igual a duas teclas. Finalmente, a letra "S" passou da terceira tecla para a primeira, deslocando duas teclas. O valor total de dissemelhança entre os dois teclados é igual a seis.

Figura 5.14 – Teclado alfabético com permutações entre as letras A, Z e S.

Além de apresentar o resultado final da função, o sistema também mostra todas as permutações e o número total de alterações entre as letras. Espera-se que utilizando essas informações, o usuário consiga decidir qual é o teclado que fornecerá a ele a melhor performance de digitação com o menor esforço.

5.6 Métricas do Teclado Virtual

Os dados de medição de tecla de varredura são imprecisos, isso dificulta a comparação entre os teclados virtuais (MIRó-BORRáS et al., 2009). (MIRó-BORRáS et al., 2009) atribuiram esta dificuldade as diferenças entre a interface do teclado, o tempo de varredura, os diferentes tipos de predição e os tamanhos dos dicionários.

Apesar das dificuldades citadas, a maioria dos trabalhos utiliza o WPM para comparar a performance de digitação dos teclados virtuais (MACKENZIE, 2007). Portanto, a primeira medida adotada por este trabalho para aferir a performance do teclado virtual será o WPM. Entretanto, de acordo com (MACKENZIE, 2007) essa medida não afere o número de erros cometidos pelo usuário.

Para aferir a quantidade de erros optou-se por utilizar a medida KSPC que além de calcular a quantidade de erros do usuário também analisa o esforço de digitação. As medidas de performance, o esforço e o erro de digitação implementados pelo teclado virtual auxiliarão o usuário a realizar a mudança entre os teclados virtuais.

Antes de adotar um *layout* de teclado o usuário pode realizar um teste de digitação para verificar a performance do novo teclado. Esse teste pode auxiliar o paciente a decidir se adotará o teclado sugerido pelo sistema ou continuará com o mesmo.

5.7 Considerações Finais

Neste capítulo foi apresentado um teclado virtual desenvolvido especificamente para pessoas com SE. As características desse sistema foram definidas de acordo com a literatura com base na segunda revisão realizada para esta tese. Todos esses atributos foram determinados com o objetivo de construir um teclado assistivo que tivesse o menor esforço de digitação e aumentasse a performance de entrada de dados.

As técnicas de predição de texto foram utilizadas com a finalidade de auxiliar o usuário a digitar mais rápido. Foram implementadas cinco técnicas de predição de texto: bigram, trigram, quadragram, k-gram e regência de uso. Além disso, para esse sistema foi codificado um algoritmo de desambiguidade baseado no método TNK. Esse método é independente da quantidade de teclas ou da distribuição das letras do teclado virtual.

Como o teclado proposto possui alta ambiguidade, devido ao número reduzido de teclas, foi proposto um algoritmo de desambiguidade de letras. Esse algoritmo apresenta um novo método de desambiuidade que prevê a distinção de apenas uma letra das teclas da sequência da palavra. Até o momento não foi identificado nenhum teclado virtual ambíguo que fizesse a desambiguidade parcial das teclas.

Finalmente, com o objetivo de otimizar a distribuição das letras entre as teclas do *software* do teclado foi proposta uma metodologia de evolução de teclados virtuais. Diferente do método de otimização tradicional a finalidade dessa metodologia é adaptar o teclado virtual de acordo com o vocabulário e o modo de escrita do usuário. Além disso, esse método pode sugerir teclados mais eficientes com base na evolução do vocabulário e no modo de escrita do usuário.

Para auxiliar o usuário a se adaptar a novos *layouts* de teclados virtuais foi proposto um algoritmo de dissemelhança. Esse algoritmo mostra ao paciente o grau de diferença entre o teclado atual e o teclado proposto. E, antes de alternar o teclado o usuário pode analisar a performance e o esforço de digitação com o teclado sugerido. Esses testes auxiliam o usuário a tomar a decisão se deve trocar ou não de teclado.

O próximo capítulo apresenta um experimento que compara a metodologia de evolução dos teclados virtuais com o método de otimização tradicional para teclados virtuais assistivos.

CAPÍTULO 6

AVALIAÇÃO DA METODOLOGIA E RESULTADOS OBTIDOS

6.1 Introdução

Com o objetivo de comparar o método de otimização dos teclados tradicionais e a metodologia de evolução dos teclados assistivos proposta neste trabalho foi elaborado e executado um experimento científico. A metodologia adotada nesse experimento foi fundamentada em (WOHLIN et al., 2012).

Este capítulo apresenta o protocolo, a coleta, a análise e as conclusões obtidas nesse experimento.

6.2 Protocolo do Experimento

6.2.1 Objetivo

Existem diversas formas de definir os objetivos. Basili e Rombach (1988) propuseram que os objetivo fossem definidos considerando o formato apresentado na Tabela 6.1.

Tabela 6.1 – Formato para a definição do objetivo

Analisar a	<Objetos de estudo>
Com o propósito de	<Propósito>
No que diz respeito a	<Ponto central da qualidade>
Do ponto de vista da	<Perspectiva>
No contexto do	<Contexto>

A Tabela 6.2 mostra o objetivo deste experimento. Esse objetivo é descrito de acordo com a definição apresentada na Tabela 6.1.

Tabela 6.2 – Definição dos objetivos deste experimento

Analisar a	metodologia evolutiva de otimização dos teclados virtuais assistivos
Com o propósito de	avaliar
No que diz respeito a	redução do esforço de digitação
Do ponto de vista do	pesquisador
No contexto do	desenvolvimento dos teclados assistivos para pacientes com SE.

6.2.2 Seleção do contexto

O contexto de um experimento está relacionado com o ambiente em que ele é realizado. Esse contexto pode ser classificado de acordo com quatro dimensões (WOHLIN et al., 2012):

1. On-line ou off-line;
2. Estudante ou profissionais;
3. Problema fictício ou problema real; e
4. Específico ou geral.

O experimento executado neste trabalho foi orientado a tecnologia e foi realizado inteiramente em um ambiente virtual. O fato de realizar esse experimento em um ambiente controlado o caracteriza como *off-line*.

O problema abordado é fictício, pois apesar da tentativa da redução do esforço de digitação em um teclado virtual assistivo ser uma questão real, esse problema foi desenvolvido e realizado em um ambiente virtual. Logo, os sujeitos para esse experimento são todos agentes virtuais. Isso impossibilita definir a segunda dimensão para este experimento.

6.2.3 Seleção das variáveis

Antes de projetar o experimento é necessário definir as variáveis dependentes e independentes que serão utilizadas e analisadas durante a execução do processo de teste das hipóteses. As variáveis independentes são aquelas que podem ser alteradas e controladas durante o experimento. Essas variáveis são modificadas com o objetivo de possibilitar as alterações nas variáveis dependentes.

A **variável independente** selecionada neste experimento é a **metodologia utilizada para gerar teclados assistivos otimizados**. Essa metodologia possui dois tratamentos: a abordagem tradicional da geração dos teclados virtuais ou a metodologia da geração dos teclados assistivos personalizados proposta neste trabalho.

A abordagem tradicional utilizada pela maioria dos trabalhos de otimização usa o corpus de uma língua como entrada para o processo de otimização. Esse conjunto de textos é pré-processado fornecendo informações como a frequência de ocorrência das palavras e das letras. A partir dessas informações é utilizada uma meta-heurística qualquer para gerar um teclado virtual otimizado.

A metodologia de evolução dos teclados virtuais assistivos proposta nesta tese, diferente do método tradicional, prevê como entrada para a meta-heurística um conjunto de textos que o usuário em alguma época tenha digitado ou que ele tenha lido. Esses textos, assim como no método tradicional, são pré-processados e servem de entrada para o algoritmo genético que gera um conjunto de teclados virtuais. Outra diferença é que o processo de otimização tradicional é realizado apenas uma vez, enquanto a metodologia proposta é executada periodicamente de acordo com a evolução do vocabulário do usuário.

A **variável dependente** é o **esforço de digitação** necessário para escrever determinado texto utilizando o teclado virtual gerado por cada método de otimização. Essa variável é calculada usando os teclados virtuais obtidos pelas duas abordagens e verificando computacionalmente qual é o esforço necessário para digitar cada texto. É importante ressaltar que os caracteres especiais, os números e os caracteres de pontuação não são considerados para realizar esse cálculo.

6.2.4 Formulação das hipóteses

A hipótese é uma predição ou uma tentativa de afirmação sobre a relação entre as variáveis. Ela é apresentada normalmente nas pesquisas quantitativas como a realizada neste trabalho (GIVEN, 2008). Existem dois tipos de hipóteses: nula e alternativa. A hipótese nula é a hipótese que o pesquisador deseja rejeitar com o maior nível de confiança possível. Enquanto, a hipótese alternativa é a aquela que é aceita caso a hipótese nula seja rejeitada.

As hipóteses deste experimento são apresentadas a seguir:

- **Hipotese nula (H_0):** Teclados virtuais gerados a partir do algoritmo genético e do corpus de uma língua resultam em teclados **com desempenhos superiores** aqueles obtidos a partir da metodologia evolutiva dos teclados assistivos personalizados proposta neste trabalho.

- **Hipotese alternativa 1 (H_1):** Teclados virtuais gerados a partir do algoritmo genético e do corpus de uma língua resultam em teclados **com desempenhos inferiores** aqueles obtidos a partir da metodologia evolutiva dos teclados assistivos personalizados proposta nesta tese.

6.2.5 Seleção dos sujeitos

A seleção de sujeitos é conhecida como amostragem (WOHLIN et al., 2012). Neste experimento, os sujeitos seriam pessoas com SE. Entretanto, realizar a validação da metodologia proposta com essas pessoas levaria bastante tempo. Isso porque seria necessário finalizar a construção do teclado assistivo. Posteriormente, identificar as pessoas com SE que pudessem usar o sistema proposto neste trabalho. Logo, seria preciso acompanhar a utilização desse sistema pelos pacientes durante alguns anos para finalmente coletar uma quantidade suficiente de dados para avaliar a metodologia proposta.

Devido a essas restrições, para mostrar a efetividade da metodologia de evolução do teclado assistivo, foi necessário desenvolver um método para simular os sujeitos reais. Essa simulação considerou o fator da evolução do vocabulário do sujeito virtual. Portanto, esses sujeitos devem possuir um vocabulário inicial e esse vocabulário deve evoluir de acordo com o tempo. Assim como a frequência das palavras devem ser alteradas conforme a evolução dos textos escritos por eles.

Para simular a evolução do vocabulário do usuário e o seu modo de escrever, foram identificados escritores da literatura que tivessem um conjunto expressivo de obras publicadas. Cada escritor representou um sujeito virtual e suas respectivas obras, o seu vocabulário e o seu modo de escrever. Assim, a cada obra inserida no sistema é atualizado o vocabulário do escritor como se ele estivesse utilizando o teclado virtual para digitar as suas obras.

Os critérios de seleção dos autores foram: (1) disponibilidade dos textos em formato digital e gratuito; e (2) quantidade de obras maior ou igual a sete. Dessa forma, um número maior de obras simularia a aquisição do vocabulário pelo sujeito, pois uma das características do método desenvolvido é a evolução do vocabulário do usuário. A Tabela 6.3 mostra a ordem de análise de cada escritor, o nome dos autores e a quantidade de obras analisadas por escritor.

6.2.6 Projeto do experimento

Com a finalidade de realizar conclusões mais abrangentes de um experimento, é necessário analisar estatisticamente o método durante a coleta dos dados para interpretá-los. Para atingir esse objetivo, é importante que o experimento seja projetado corretamente, pois o método estatístico que será utilizado depende diretamente do tipo de projeto selecionado (WOHLIN et al., 2012).

O tipo de projeto escolhido para este experimento utiliza apenas **um único fator e dois tratamentos**. Portanto, será analisado a redução do esforço de digitação, variável dependente, para o método padrão de otimização e para a metodologia de evolução dos teclados virtuais

Tabela 6.3 – Apresentação dos Sujeitos e a quantidade de obras utilizadas no experimento

Sujeito	**Quantidade de Obras**
Aldous Huxley	7
Agatha Christie	7
André Vianco	7
Anne Perry	7
Arthur C. Clarke	7
Clarice Lispector	11
Edgar Allan Poe	7
Jorge Amado	14
José de Alencar	16
José Saramago	7
Isaac Asimov	7
Machado de Assis	10
Guimaraes Rosa	7
Paulo Coelho	13
Regina Dforges	8

personalizados proposto neste trabalho, variável independente.

O modo mais comum de fazer essa comparação é analisar a variável dependente de acordo com o método aplicado. Assim, optou-se por utilizar o método de **comparação pareada**. Nesse método, cada sujeito utiliza dois tipos de tratamento e posteriormente os resultados são comparados.

6.2.7 Instrumentação

Foram necessários um computador, um software que implementa as metodologias de geração dos teclados virtuais e os *softwares* de apoio.

As configurações do computador foram: memória ram de 8GB, processador intel I5 com 4 núcleos de processamento e o sistema operacional Windows 7.

O sistema desenvolvido foi projetado para receber como entrada a configuração do teclado virtual, o número de teclas e a sequência de distribuição das letras, e um texto a ser digitado. Esse *software* também calcula o esforço de digitação dos teclados gerados pelo processo de otimização em relação ao texto de entrada. Com o objetivo de realizar a análise dos resultados, as informações geradas por esse sistema foram armazenadas em planilhas eletrônicas.

6.2.8 Métricas

Para a coleta dos dados necessários para a verificação das hipóteses foram definidas duas métricas: esforço de digitação de uma palavra e esforço de digitação de um texto. A seguir são apresentados os detalhes dessas métricas.

Número da métrica: 1
Nome: Esforço de digitação de uma palavra
Objetivo: Quantifica o quanto de esforço é requerido para digitar uma palavra utilizando um determinado teclado virtual
Forma de coleta:

1. Contar a quantidade de teclas necessárias para digitar essa palavra (f_{ct}) utilizando um teclado t.
2. Contar o número de vezes que o método de varredura altera entre as teclas para produzir a sequência de teclas necessárias para compor a palavra (f_{cv}) utilizando um teclado t.
3. Se a palavra desejada é a primeira na lista de ambiguidade, então soma-se mais 5, para simular a seleção da tecla de espaço.
4. Se não é a primeira palavra, soma-se 6 ao total, simulando a seleção da lista de desambiguidade. Além disso, soma-se o índice da palavra (f_{ip}) o qual representa a posição da palavra na lista. Assim, se a lista é composta de "Abc Bcd Cde Def"e deseja-se selecionar a palavra "Cde", o índice da palavra é igual a três, pois ela está localizada na terceira posição da lista.

Equação da métrica:

$$f_c(t,p) = \begin{cases} f_{cv}(t,p) + f_{ct}(p) + 1, & se f_a(t,p) \leq 1 \\ f_{cv}(t,p) + f_{ct}(p) + 6 + f_{ip}(t,p), & se f_a(t,p) > 1 \end{cases} \tag{6.1}$$

em que: t é o teclado que está sendo analisado, p representa a palavra que está sendo analisada, $f_{cv}(t,p)$ é a função que calcula o esforço de varredura, $f_{ct}(p)$ representa a função que calcula a quantidade de teclas necessárias para compor a palavra, $f_a(t,p)$ é a função que calcula o índice da palavra na lista de desambiguidade e $f_c(t,p)$ representa o esforço necessário para digitar uma palavra qualquer.
Tipo de escala: racional
Forma de interpretação: Quanto menor é o resultado menor é o esforço para digitar a palavra

A Equação (6.2) mostra o cálculo do esforço de digitação de um texto. Esse cálculo aplica a Equação (6.1) a todas as palavras desse texto.

Número da métrica: 2
Nome: Esforço de Digitação de um Texto
Objetivo: Determina o quanto de esforço é requerido para digitar um texto utilizando a métrica (6.1) em todos os termos do texto em análise. **Forma de coleta:**

1. Aplicar a métrica "esforço de digitação de uma palavra"para todas as palavras do texto ($f_ct(t)$)
   .

Equação da métrica:

$$f_{ct}(t,o) = sum_{x=1}^{n_o} f_c(o_p(x)) \tag{6.2}$$

em que: n é o número de palavras do texto, t representa a configuração do teclado que está sendo analisado, o é o texto que está sendo analisado, o_p representa a função que determina uma palavra da obra do autor e n_o é o número de palavras no texto.
Tipo de escala: racional
Forma de interpretação: Quanto menor é o resultado menor é o esforço para digitar o texto

6.2.9 Planejamento da análise dos dados

A estatística descritiva é a área da matemática que aborda o processamento numérico e a representação de um grupo de dados. Esse tipo de estatística pode ser utilizado para consolidar os dados e apresentá-los de uma maneira mais clara e inteligível (WOHLIN et al., 2012). O processo de consolidação da informação pode ser realizado após a fase da coleta dos dados. Alguns exemplos de estatísticas descritivas são: média aritmética, mediana, moda, média geométrica e percentil.

Os métodos estatísticos são aplicados de acordo com a escala de medição de cada variável. Portanto, o tipo de escala determina qual a estatística que deve ser utilizada para o experimento. Por exemplo, nos experimentos em que as variáveis dependentes possuem escalas nominais, a única estatística possível de ser utilizada é a moda. Entretanto, se as variáveis forem do tipo intervalar aplica-se a média, a mediana, o percentil, a variância e o intervalo. A Tabela 6.4 mostra a relação entre as escalas e a estatística.

Tabela 6.4 – Escalas por métrica (WOHLIN et al., 2012)

Tipo de Escala	**Medidas de tendência central**
Nominal	moda
Ordinal	moda, mediana e percentil
Intervalar	moda, mediana, percentil, média, variância e intervalo
Racional	moda, mediana, percentil, média, variância, intervalo, média geométrica

Para consolidar os dados deste trabalho optou-se por utilizar a média aritmética dos esforços realizados para digitar todos os textos de um escritor utilizando cada tratamento, o método comum e o método proposto.

Após o agrupamento dos dados é necessário realizar o teste de hipótese. O objetivo desse teste é verificar se é possível rejeitar determinada hipótese nula, H_0, fundamentado pelos dados consolidados. A rejeição dessa hipótese possibilita o pesquisador considerar, com um determinado grau de confiança, que as hipóteses alternativas estão corretas. A escolha do teste de hipótese está relacionada ao tipo de escala das variáveis e ao tipo de projeto do experimento.

Para o experimento realizado neste trabalho optou-se por utilizar o t-Teste. Esse teste de hipótese é indicado quando ocorre um fator com dois tratamentos e os dados coletados seguem uma distribuição normal, exatamente o que acontece nesta tese (WOHLIN et al., 2012). A Equação (6.3) mostra o cálculo necessário para realizar o t-Teste.

$$t_{calc} = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}} \tag{6.3}$$

em que: $\bar{x}$ é a média da amostra, μ_0 representa o valor fixo usado na comparação com a média da amostra, s é o desvio padrão amostral e n representa o tamanho da amostra.

Para certificar que a amostra das variáveis dependentes obedece uma distribuição normal é utilizado o teste de Kolmogorov Smirnov (MASSEY, 1951). Esse teste avalia duas hipóteses:

- **Hipotese nula (H_0):** Os dados **seguem** uma distribuição normal; e
- **Hipotese alternativa 1 (H_1):** Os dados **não seguem** uma distribuição normal.

Se a hipótese nula é rejeitada pode-se concluir que o conjunto de dados que foi submetido ao teste não obedece a uma distribuição normal. A Equação (6.4) mostra o cálculo do teste de Kolmogorov Smirnov.

$$D_n = \sup_n |F_n(x) - F(x)| \tag{6.4}$$

em que: $F_n(x)$ é a função de distribuição acumulada empírica dos dados e $F(x)$ representa a função de distribuição acumulada assumida para os dados.

Com a finalidade de facilitar o cálculo estatístico dos dados, foi utilizado o software SPSS. No SPSS é possível obter as médias, os gráficos e realizar os testes de normalidade e de hipótese.

6.2.10 Limitações do experimento

Como somente quinze autores foram selecionados para serem sujeitos do experimento, o número de amostras foi reduzido. Assim, o teste de hipótese ficou comprometido. Entretanto, encontrar autores que possuíam obras disponíveis para esse tipo de análise foi uma tarefa complexa. Isso ocorreu porque muitas vezes essas obras estavam em um formato que não era possível extrair os textos, como por exemplo, pdf's protegidos por senha ou epub's.

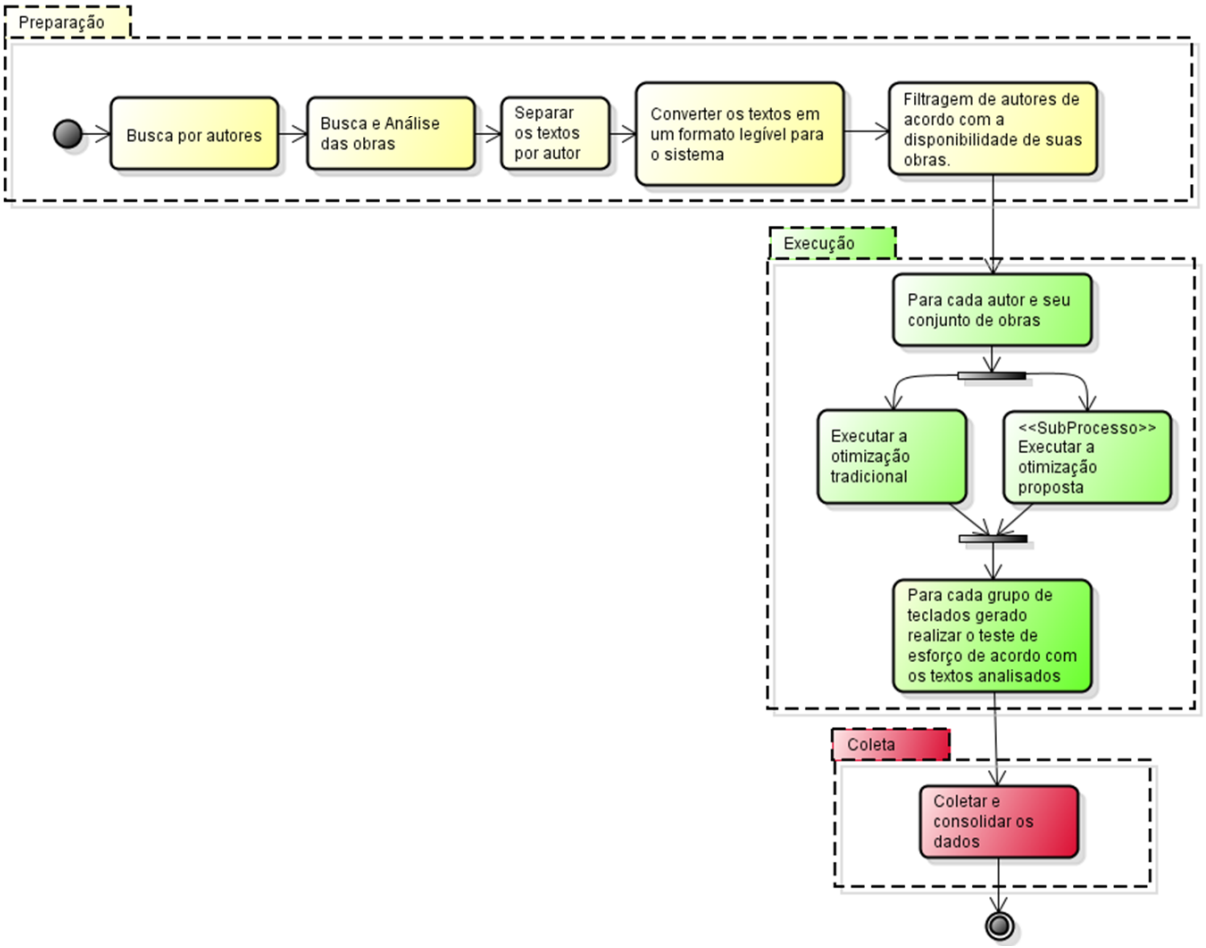
O pesquisador que desenvolveu o método é o mesmo que executou o experimento. O interesse desse pesquisador era de que o método apresentasse um bom desempenho. Portanto, existia um risco de que o pesquisador inconscientemente, ou não, selecionasse as métricas que fossem favoráveis ao método investigado. Para evitar esse risco, o protocolo do experimento foi revisado por outro pesquisador que apesar de não conhecer profundamente o problema, possuía prática na execução e na elaboração de experimentos.

6.3 Relato Operacional do Experimento

Este experimento foi executado em duas etapas. A primeira foi iniciada a partir de 25 de maio de 2015 e foi finalizado no dia 20 de julho de 2015. A segunda etapa teve início em 5 de outubro de 2015 e foi finalizada em 29 de outubro de 2015. A Figura 6.1 ilustra o diagrama de todo o processo de execução desse experimento. Esse processo foi realizado em três fases distintas, preparação, execução e coleta. Na primeira fase foi preparado o material utilizado pelos métodos de otimização, na segunda esses métodos foram executados e finalmente na última fase os dados foram coletados.

Na primeira fase foi realizada uma **busca por autores** que tivessem um número de obras igual ou maior do que sete. Foram identificados diversos autores que poderiam ser utilizados neste trabalho tais como: José de Alencar, Machado de Assis, Clarice Lispector, Carlos Drummond de Andrade, Guimarães Rosa, Paulo Coelho, Jorge Amado, Regina D' Forgers, J K Rowling, dentre outros. Todos os autores citados foram considerados possíveis sujeitos deste experimento.

Figura 6.1 – Processo de execução do experimento proposto para mostrar a eficiência da metodologia desenvolvida nesta tese.



Após identificar os escritores que poderiam fazer parte desta pesquisa, foi iniciado o processo de **busca e análise das obras** desses autores. A coleção dos textos deveria estar disponível em formato digital e ser gratuita. Além disso, os livros digitais deveriam ser passíveis de serem transformados em um formato que pudesse ser carregado pelo sistema de leitura.

Essa busca foi realizada em duas semanas e meia e foram encontradas diversas obras em vários formatos. Depois de copiar as obras para o computador utilizado nesta pesquisa, foram agrupados os textos de acordo com seus autores. Finalmente, foi iniciado o processo de **conversão dos textos em um formato legível para o sistema**.

Assim que todos os textos foram convertidos, foi realizada uma **filtragem de autores de acordo com a disponibilidade de suas obras**. Nessa segunda triagem foram eliminados da lista de possíveis sujeitos, os autores dos quais a coleção de obras não estava disponível em formato digital e gratuito. Autores que possuíam apenas obras que não podiam ser carregadas pelo sistema de leitura foram desconsiderados. A Tabela 6.3 mostra os escritores e a quantidade de obras por escritor que foram utilizadas no experimento.

Após definir as obras e os escritores, sujeitos deste experimento, foi iniciado o processo de otimização do teclado virtual. Assim, **para cada autor e o conjunto de suas obras** foram utilizados dois métodos de otimização. O primeiro usava o corpus da língua portuguesa, o mesmo que gerou o léxico do sistema, como entrada inicial para a otimização do teclado virtual. O segundo método utilizava a metodologia evolutiva de otimização dos teclados virtuais assistivos proposta nesta tese.

Primeiro foram gerados teclados otimizados a partir do corpus da linguagem. Para isso foi necessário construir um léxico com base no apanhado dos textos extraindo seu vocabulário e a frequência de cada palavra. O desenvolvimento desse dicionário foi mostrado no Capítulo 5 desta tese. Para garantir que todas as palavras que pertenciam ao vocabulário do autor estivessem contidas no léxico foi preciso carregar todo vocabulário do autor para o dicionário. Esse procedimento foi importante, pois no teste de esforço é calculado o esforço de digitação de todas as palavras escritas nos textos. Se o dicionário não tivesse todas as palavras contidas no texto o sistema não funcionária.

Finalmente, o algoritmo genético foi utilizado para otimizar o teclado virtual considerando além do vocabulário do autor, todas as palavras do dicionário e suas frequências. É importante ressaltar que as frequências de ocorrência das palavras utilizadas pelo autor não foram adicionadas ao léxico construído a partir do corpus. Portanto, os termos que não constavam no léxico e pertenciam ao vocabulário do autor foram inseridos com a frequência de ocorrência igual a 0. Porém, se a palavra estivesse contida no léxico, a sua frequência de ocorrência era mantida.

Ao final do processo de otimização foram encontradas diversas soluções de teclados assistivos. O teste de esforço foi executado em cada teclado desenvolvido. Esse teste calculava o esforço necessário para realizar a digitação de cada texto do sujeito utilizando o teclado construído. Esse cálculo foi executado usando a Equação (6.2) para cada teclado desenvolvido em relação a todos os textos de cada sujeito.

Ao final da otimização foi construída uma planilha eletrônica com os seguintes dados: o nome do arquivo de teste, o esforço de varredura, o grau de ambiguidade, o esforço necessário para digitar cada texto com o teclado desenvolvido e a porcentagem de utilização de cada tecla do teclado assistivo. A Figura 6.2 ilustra uma linha desta planilha.

Figura 6.2 – Linha da planilha de saída do processo de otimização pelo corpus.

Arquivo Teste	Teclado	Esforco teclado	Ambiguidad	Esforco Di	Tecla 1%	Tecla 2%	Tecla 3%	Tecla 4%
Cidade Sitiada	[25, 4, 9, 12, 3	9097350	5,599	833765	33,70	29,85	22,11	14,34

Após executar a otimização com o corpus da lingua portuguesa, foi realizado o teste com a metodologia proposta neste trabalho. Os textos de cada sujeito foram organizados em

ordem cronológica de escrita. Em seguida foi desenvolvido um léxico inicial com o vocabulário e a frequência de ocorrência dos termos utilizados na primeira obra do autor. Em seguida, foi executado o algoritmo genético de otimização com base no léxico que tinha acabado de ser construído. Finalmente, para cada teclado desenvolvido a partir desse processo de otimização, foi realizado o teste de esforço. Assim, todos os teclados foram utilizados para digitar todos os textos do sujeito em análise.

Novamente, foi construída uma planilha eletrônica com os resultados dos testes. Essa planilha era semelhante a planilha de otimização do corpus. A única diferença entre esses dois documentos é que o segundo continha um campo com o nome do arquivo que originou a otimização. A Figura 6.3 ilustra uma linha desta planilha.

Figura 6.3 – Linha da planilha de saída do processo de otimização por texto.

Arquivo Gerador	Arquivo Teste	Teclado	Esforco teclado	Ambiguidad	Esforco Di	Tecla 1%	Tecla 2%	Tecla 3%	Tecla 4%
A bela e a Fera	Cidade Sitiada	[25, 4, 9, 12, 3	9097350	5,599	833765	33,70	29,85	22,11	14,34

O processo de geração dos teclados foi repetido para todos os textos do sujeito, com a diferença de que, para cada texto analisado, o léxico era atualizado. Essa atualização adicionava ao dicionário os termos do novo texto. Além de adicionar novas palavras ao léxico, a frequência das palavras eram atualizadas de acordo com o número de ocorrências dos termos nos novos textos. Esse procedimento simulava o crescimento do vocabulário do usuário.

Após coletar os dados de todas as planilhas construídas a partir da aplicação do processo de otimização, foi iniciado o processo de **coleta e consolidação dos dados**. E também, foi desenvolvida uma nova planilha de consolidação da informação. O objetivo desta planilha é resumir os dados extraídos do processo de otimização.

Para cada planilha de otimização foi extraído o valor do esforço de digitação do melhor teclado otimizado considerando o texto em análise. Com o objetivo de coletar esse valor, foi necessário identificar o teclado que obteve o menor esforço de digitação para o texto que gerou os teclados otimizados. Em seguida, foi preciso recuperar o esforço de digitação que este teclado teve em todos os outros textos.

Nos dados que foram obtidos a partir do processo de otimização do corpus da língua foi necessário utilizar um processo de extração diferente. Isso porque, a otimização foi realizada a partir do corpus e não a partir de uma obra específica. Assim, não era possível definir o melhor teclado para a obra geradora. Para definir esse valor optou-se por extrair os melhores resultados de cada teste de esforço realizado entre os teclados otimizados e as obras.

No final do processo de coleta, esses dados foram consolidados aplicando a média aritmética no esforço de digitação. Assim, para cada sujeito foi somado todos os valores dos testes de esforço e finalmente esse valor foi dividido pela quantidade de obras.

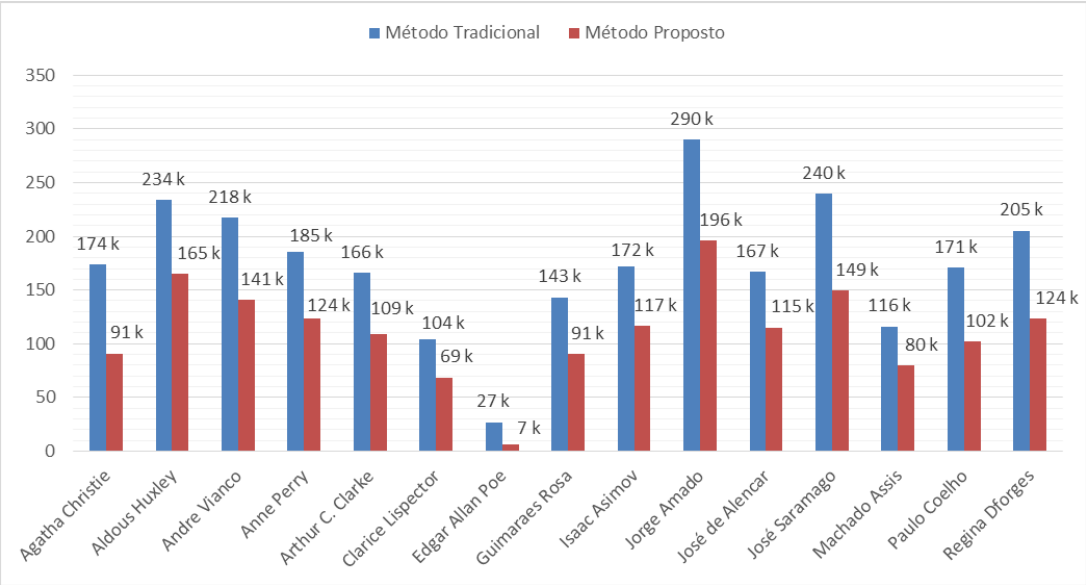
A Tabela 6.5 foi construída a partir da consolidação dos dados coletados durante a execução do experimento. A coluna "Sujeito" informa o escritor avaliado durante o experimento e a segunda coluna apresenta o esforço necessário para digitar as obras analisadas deste autor usando um teclado otimizado com o método tradicional. Finalmente, a coluna "Método Proposto" informa o esforço realizado para escrever as obras do mesmo autor utilizando o teclado otimizado gerado pela última obra de cada autor usando o método proposto.

Tabela 6.5 – Dados consolidados.

Sujeito	**Método Tradicional**	**Método Proposto**
Aldous Huxley	23405775	16521147
Agatha Christie	17376125	9097358
André Vianco	21775025	15854408
Anne Perry	18514774	12378364
Arthur C. Clarke	16578615	10916351
Clarice Lispector	10435003	6864294
Edgar Allan Poe	26960	6569
Jorge Amado	28972661	19613576
José de Alencar	16724315	11518699
José Saramago	23973157	14927397
Isaac Asimov	17179028	11658122
Machado de Assis	11620193	7957869
Guimaraes	14321419	9066213
Paulo Coelho	1710563	1022712
Regina Dforges	20528559	12362739

A Figura 6.4 ilustra a diferença entre os dois métodos de acordo com cada sujeito segundo a Tabela 6.5.

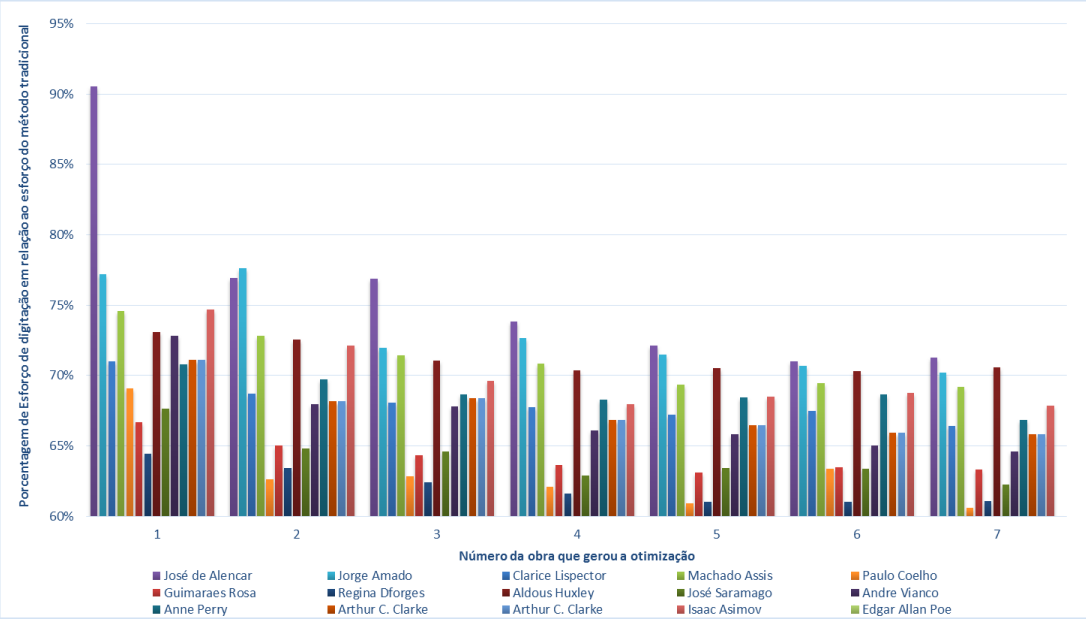
Figura 6.4 – Comparação entre o esforço de digitação entre os teclados desenvolvidos pelos dois métodos.



Pode-se verificar que o método proposto apresenta melhores resultados do que o método tradicional e as diferenças entre esses dois métodos variam de 10% no pior caso e 40% no melhor caso.

A Figura 6.5 mostra os resultados da metodologia proposta aplicada sequêncialmente as obras dos autores selecionados para o experimento. Com a finalidade de tornar o gráfico mais claro apenas as primeiras sete evoluções são apresentadas.

Figura 6.5 – Evolução da redução do esforço de digitação de acordo com a metodologia proposta comparada com o esforço de digitação realizado pelo método tradicional.



Finalmente, as figuras de 6.6 a 6.20 apresentam a evolução dos teclados virtuais de acordo com cada autor.

Figura 6.6 – Gráfico de Agatha Cristie.

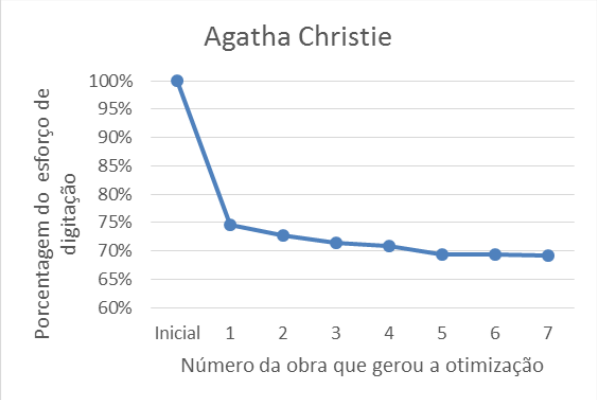


Figura 6.7 – Gráfico de Aldos Huxley.

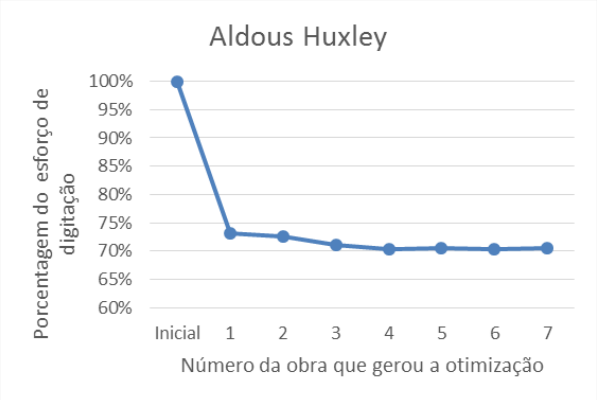


Figura 6.8 – Gráfico de André Vianco.

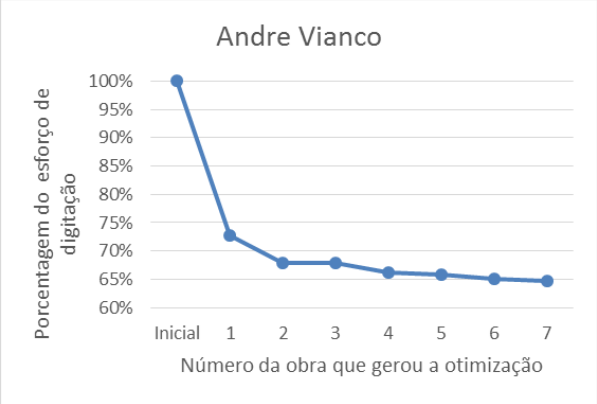


Figura 6.9 – Gráfico de Anne Perrie.

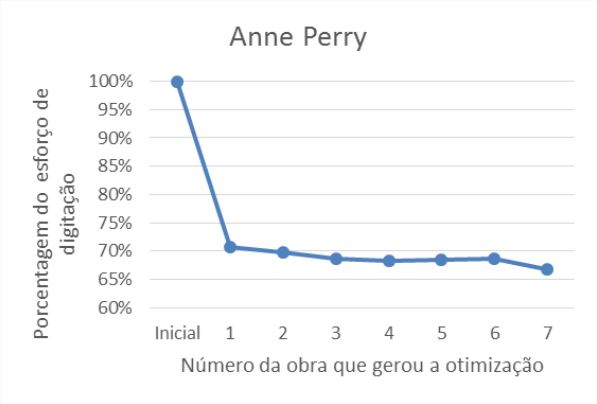


Figura 6.10 – Gráfico de Arthur Clarck.

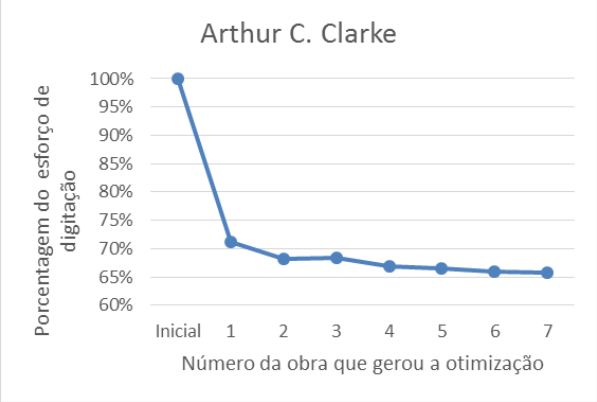


Figura 6.11 – Gráfico de Clarice Lispector.

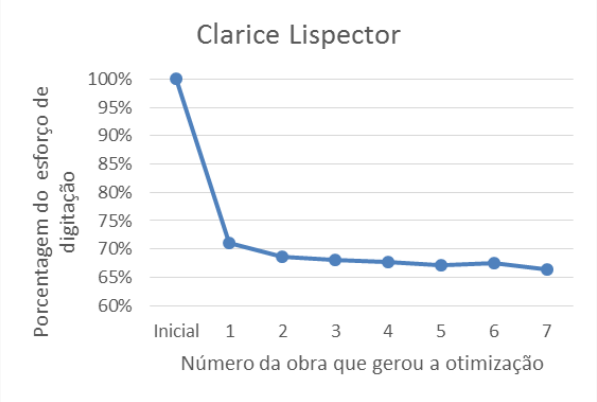


Figura 6.12 – Gráfico de Edgar Allan Poe.

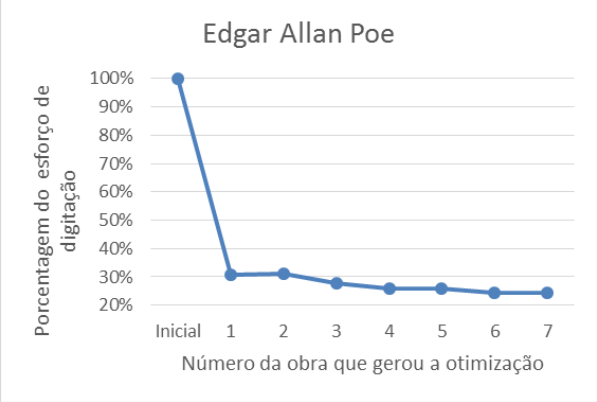


Figura 6.13 – Gráfico de Guimaraes Rosa.

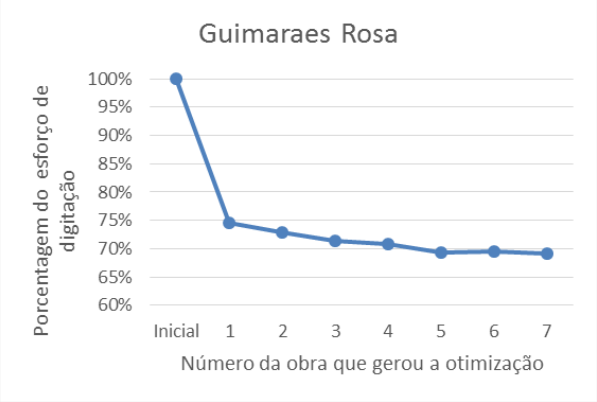


Figura 6.14 – Gráfico de Issac Assimov.

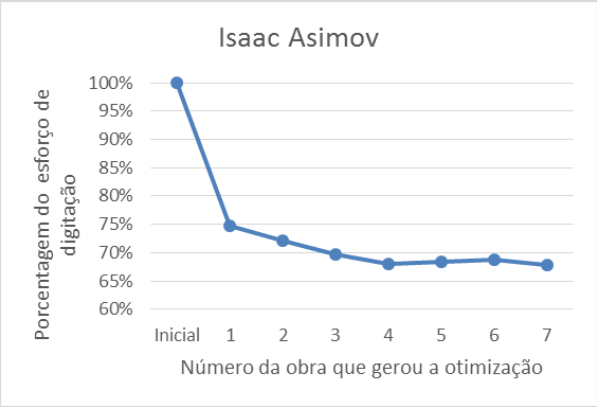


Figura 6.15 – Gráfico de Jorge Amado.

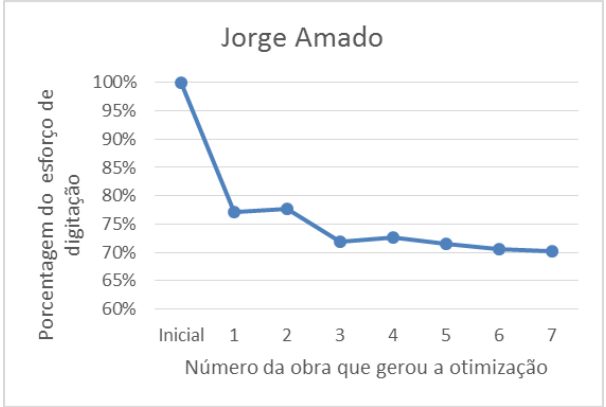


Figura 6.16 – Gráfico de José de Alencar.

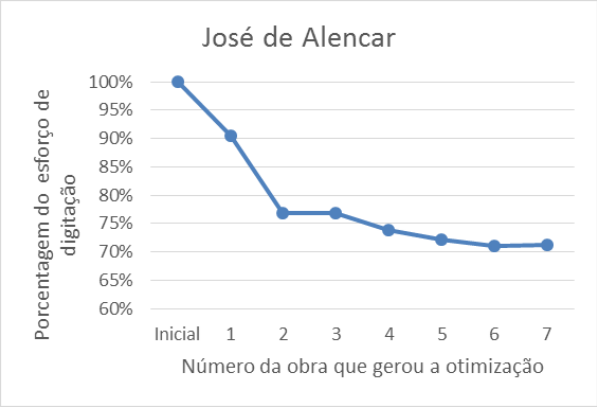


Figura 6.17 – Gráfico de José Saramago.

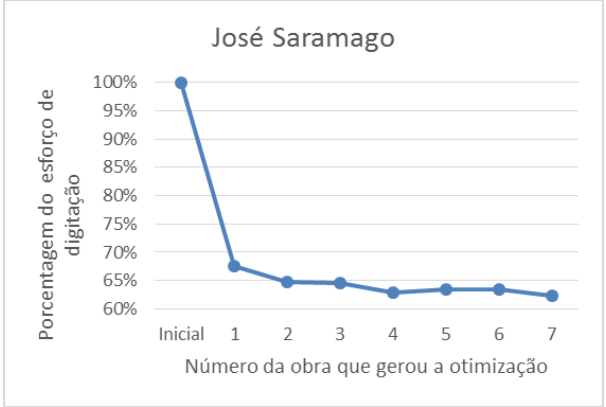


Figura 6.18 – Gráfico de Machado de Assis.

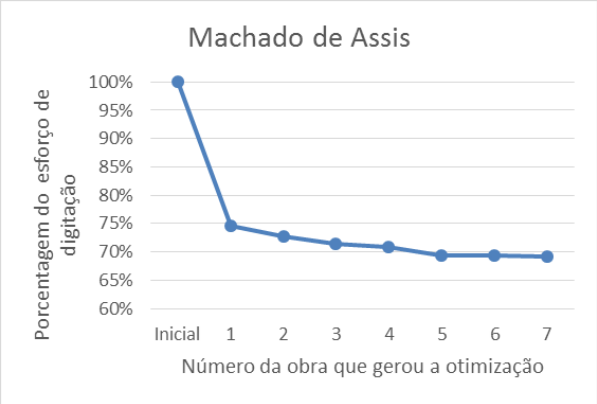


Figura 6.19 – Gráfico de Paulo Coelho.

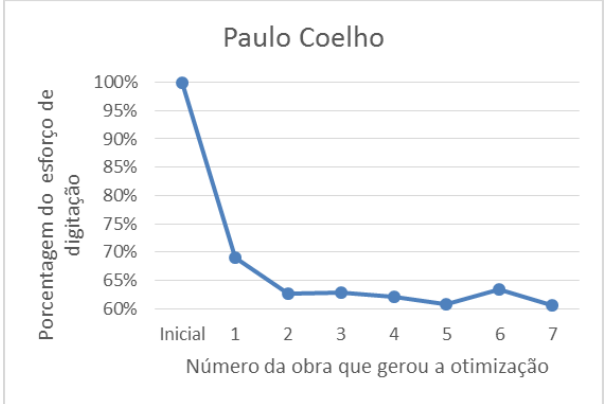
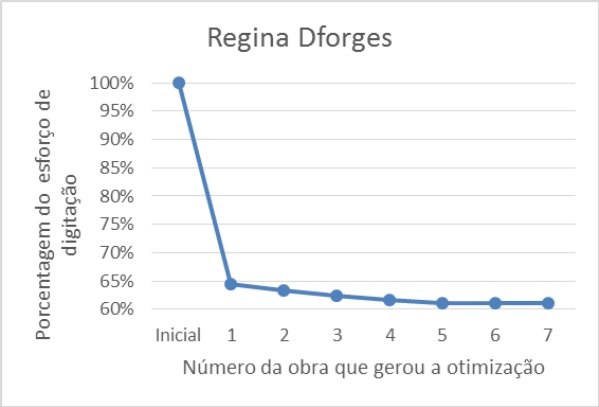


Figura 6.20 – Gráfico de Regina Dforges.



Apesar de não ser ilustrado no gráfico o esforço do método tradicional foi considerado como 100%. Assim, para cada evolução é mostrado o quanto do esforço tradicional deveria ser utilizado para produzir o mesmo texto utilizando a metodologia proposta. Por exemplo, considerando a primeira evolução dos textos de José de Alencar, a diminuição do esforço em relação ao método tradicional foi de 10% na pmétrimeira otimização. Entretanto, na segunda otimização esse valor passa a ser igual a mais de 20%. Na sétima evolução o esforço de digitação diminui para mais de 23% em relação ao método tradicional.

6.4 Análise dos Resultados

Aplicando o teste de hipótese nos dados extraídos do experimento é possível rejeitar a hipótese nula e validar as hipóteses alternativas. Para realizar esta fase do experimento foi utilizado o software SPSS (IBM, 2015). O objetivo do SPSS é auxiliar os pesquisadores nos cálculos estatísticos aplicados em experimentos. Esse *software* realiza diversos testes estatísticos e de hipótese. Entre os métodos implementados pelo programa estão os testes de Kolmogorov Sirminov e o t-Test.

Antes de realizar o teste de hipótese foi necessário aplicar o teste de Kolmogorov Smirnov nas duas amostras. O objetivo desse teste é garantir que os dados coletados tenham uma distribuição normal. A Figura 6.21 ilustra o teste de Kolmogorov Sirminov para a amostra dos dados relativos ao método tradicional.

Pode-se verificar nas Figuras 6.21 e 6.22 que os resultados obtidos no teste estatístico para os métodos tradicional e proposto neste trabalho foram 0,185 e 0,120, respectivamente. Como os resultados dos dois testes foram maiores do que 0,05 pode-se concluir que a hipótese nula não pode ser rejeitada. Para o teste de Kolmogorov a hipótese nula define que a amostra segue uma distribuição normal. Como essa hipótese é verdadeira, conclui-se que as duas amostras seguem esse tipo distribuição. Assim, pode-se aplicar o t-Test para avaliar as hipóteses deste

Figura 6.21 – Resultado do teste de Kolmogorov Smirnov para a amostra dos dados relativas ao método tradicional.

Teste de Kolmogorov-Smirnov de uma amostra

		Tradicional
N		15
Parâmetros normais[a,b]	Média	16209478,13
	Erro Desvio	7849795,757
Diferenças Mais Extremas	Absoluto	,185
	Positivo	,101
	Negativo	-,185
Estatística de teste		,185
Significância Sig. (2 extremidades)		,175[c]

a. A distribuição do teste é Normal.

b. Calculado dos dados.

c. Correção de Significância de Lilliefors.

Figura 6.22 – Resultado do teste de Kolmogorov Smirnov para a amostra dos dados relativas ao método proposto

Teste de Kolmogorov-Smirnov de uma amostra

		Proposta
N		15
Parâmetros normais[a,b]	Média	10651054,53
	Erro Desvio	5348071,455
Diferenças Mais Extremas	Absoluto	,120
	Positivo	,107
	Negativo	-,120
Estatística de teste		,120
Significância Sig. (2 extremidades)		,200[c,d]

a. A distribuição do teste é Normal.

b. Calculado dos dados.

c. Correção de Significância de Lilliefors.

d. Este é um limite inferior da significância verdadeira.

experimento.

Finalmente, foi aplicado o t-Test nos dados coletados para verificar se a hipótese nula pode ser rejeitada e assim garantir que existe uma diferença positiva entre os métodos tradicional e proposto. A Figura 6.23 mostra o resultado do t-Test das amostras utilizando o sistema SPSS.

O resultado do grau de significância do t-Test para as amostras é igual a 0,000. Esse

Figura 6.23 – Resultado do t-Teste para as amostras dos dados relativos ao método tradicional e proposto.

Estatísticas de amostras emparelhadas

		Média	N	Desvio Padrão	Erro padrão da média
Par 1	Proposta	10651054,53	15	5348071,455	1380866,112
	Tradicional	16209478,13	15	7849795,757	2026808,549

Correlações de amostras emparelhadas

		N	Correlação	Sig.
Par 1	Proposta & Tradicional	15	,985	,000

Teste de amostras emparelhadas

		Diferenças emparelhadas					t	df	Sig. (2 extremidades)
		Média	Desvio Padrão	Erro padrão da média	95% Intervalo de Confiança da Diferença				
					Inferior	Superior			
Par 1	Proposta - Tradicional	-5558423,600	2741173,979	707768,078	-7076435,152	-4040412,048	-7,853	14	,000

resultado é menor do que 0,05 e, conforme as definições do t-Test, valores menores do que esse comprovam a rejeição da hipótese nula, H_0. Portanto, pode-se concluir que é válida a hipótese alternativa H_1. Assim, conclui-se que os teclados virtuais gerados a partir do algoritmo genético e do corpus de uma língua resultam em teclados **com desempenhos inferiores** aqueles obtidos a partir da metodologia evolutiva dos teclados assistivos proposta nesta tese. Pode-se observar na Figura 6.23 que a diferença entre as médias de esforço dos dois métodos é considerável.

Uma conclusão importante até este momento é a antagonia entre o esforço de varredura e a taxa de ambiguidade. Nos mais de dois mil teclados desenvolvidos pôde-se observar que o crescimento dessas duas variáveis é inversamente proporcional. Assim, quanto menor é a ambiguidade maior é o esforço de varredura e quanto maior é a taxa de ambiguidade menor é o esforço de varredura.

No método tradicional os pesquisadores utilizam um algoritmo de otimização para minimizar o número de colisões entre as palavras do teclado virtual. Porém, observou-se que em teclados virtuais que usam o método de varredura os teclados que tiveram o maior esforço de digitação possuíam também o menor número de colisões. Assim, é mais indicado que o teclado virtual aprimore o esforço de varredura em detrimento do número de colisões.

6.5 Experimento Evolutivo de Efetividade

Neste capítulo foi realizado um experimento para mostrar a eficiência da metodologia evolutiva de teclados assistivos em relação ao método tradicional de otimização. A metodologia proposta neste trabalho possibilita o usuário otimizar o teclado virtual a qualquer momento de acordo com a sua necessidade. Essa liberdade de escolha é importante para garantir a adaptação

do sistema de CAA ao usuário. Entretanto, não foi definido qual seria o ponto ótimo, ou pelo menos parcialmente ótimo, para efetuar o método de otimização.

Assim, foi desenvolvido um novo experimento com a finalidade de identificar qual a quantidade de palavras que o usuário deve inserir no sistema para realizar a otimização do teclado virtual e obter bons resultados. Para realizar esse experimento foi necessário coletar diversos textos de um mesmo autor e submetê-los a metodologia de evolução de teclados virtuais. A partir dos resultados obtidos, com a otimização desses textos, foi elaborada uma análise para identificar os melhores pontos de otimização.

6.5.1 Seleção dos sujeitos

Assim como no anterior, este experimento utiliza pessoas com SE inicialmente é inviável devido ao tempo disponível para a finalização desta tese. Para selecionar os sujeitos optou-se por identificar autores contemporaneos que escrevem sobre diversos assuntos periodicamente no formato digital.

Os critérios de seleção dos autores foram: (1) disponibilidade dos textos em formato digital e gratuito; e (2) quantidade de textos maior ou igual a 91. O número de textos foi definido como 91 para simular um usuário que utilize a metodologia evolutiva durante três meses, escrevendo todos os dias e acionando o mecanismo evolutivo diariamente. Assim, é possível definir quando os usuários possuem as melhores taxas de otimização.

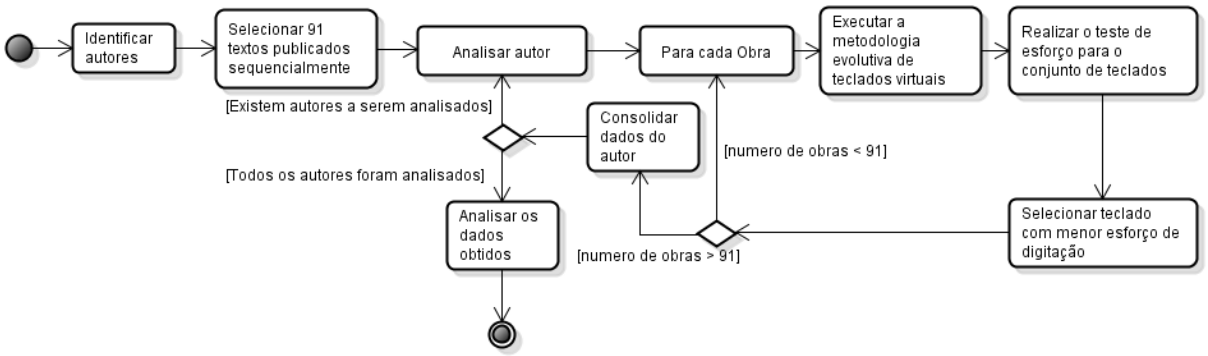
Os sujeitos escolhidos foram autores que escrevem para a Folha de São Paulo. 10 escritores foram selecionados: Xico Sá, Inácio Araujo, Frederico Vasconcelos, Pedro Diniz, Leandro Colon, Rodolfo Lucena, Alexandre Schwartsman, Antonio Prata, Márcia Dessen e Luiz Felipeponde. O site Folha de São Paulo foi escolhido por possuir conteúdo diponibilizado gratuitamente e pela periodicidade com que seus escritores publicam. Os escritores foram selecionados utilizando os critérios de seleção e a diversidade de seus assuntos.

6.6 Relato Operacional do Experimento

Este experimento foi executado a partir de 01 de outubro de 2015 e foi finalizado no dia 02 de novembro de 2015. A Figura 6.24 mostra o processo desse experimento.

O experimento iniciou com a **identificação dos autores**. Esses escritores foram escolhidos de acordo com os critérios apresentados na seção 6.5.1 deste capítulo. Após identificá-los foram **selecionados 91 textos pertencentes a eles**. Esses textos estavam em formato digital, disponíveis gratuitamente e seguiam uma ordem cronológica de publicação.

Figura 6.24 – Processo de execução do experimento proposto para identificar o ponto de otimização do teclado virtual assistivo.



Logo depois de selecionar esses textos, foi iniciado o processo para **extraí-los**. Assim, foi desenvolvido um programa que pesquisava os textos dos autores e os inseria em uma base de dados separada. Finalmente, foi realizada a fase de **análise dos autores**.

Na análise dos autores, a metodologia evolutiva de teclados virtuais foi aplicada ao conjunto de obras de cada autor. Cada texto servia de fomento para a construção de um vocabulário contendo as palavras utilizadas pelo escritor e suas frequências. Após construir esse vocabulário, o algoritmo genético era executado gerando novas composições de teclados virtuais. Os teclados gerados eram submetidos ao teste de esforço de digitação que analisava a sua performance para digitar todos os textos publicados pelo autor. A cada novo texto as palavras e suas frequências dentro do vocabulário eram atualizadas.

As informações geradas por cada otimização apresentavam: o esforço de digitação, o número de palavras, o número de palavras distintas. O esforço de digitação determina a quantidade de interações necessárias entre o usuário e o *software* para digitar um texto. O número de palavras representa a quantidade de palavras que o texto possui. O número de palavras distintas define a quantidade de palavras diferentes que o texto possui.

Se o número de textos fosse maior do que 91, os dados do autor eram consolidados e um novo autor era submetido a análise. Assim, que todos os escritores foram analisados foi iniciada a **análise dos resultados**.

6.7 Análise dos Resultados

Com as informações obtidas do experimento foi construída uma planilha eletrônica para cada autor. Essa planilha contém o nome do autor, o teclado virtual gerado, o esforço de digitação, a quantidade de palavras e a quantidade de palavras distintas dos textos.

Para cada planilha desenvolvida foi construído um gráfico. Nesse gráfico, o lado es-

querdo do eixo y representa o esforço necessário para digitar o texto em porcentagem e o lado direito o número de palavras utilizadas para otimizar o teclado. O eixo x mostra a cronologia crescente dos textos de 1 a 91.

Para facilitar a análise do gráfico, optou-se por apresentar na integra os resultados de otimização dos primeiros 30 textos. Após esses resultados as otimizações são apresentadas em intervalos de 10 textos. Além disso, o lado direito do eixo y foi limitado a um valor que possibilitasse o detalhamento dos valores do gráfico nas primeiras 30 evoluções.

As Figuras 6.25 a 6.34 mostram os gráficos de evolução por autor.

Figura 6.25 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Alexandre Schwartsman.

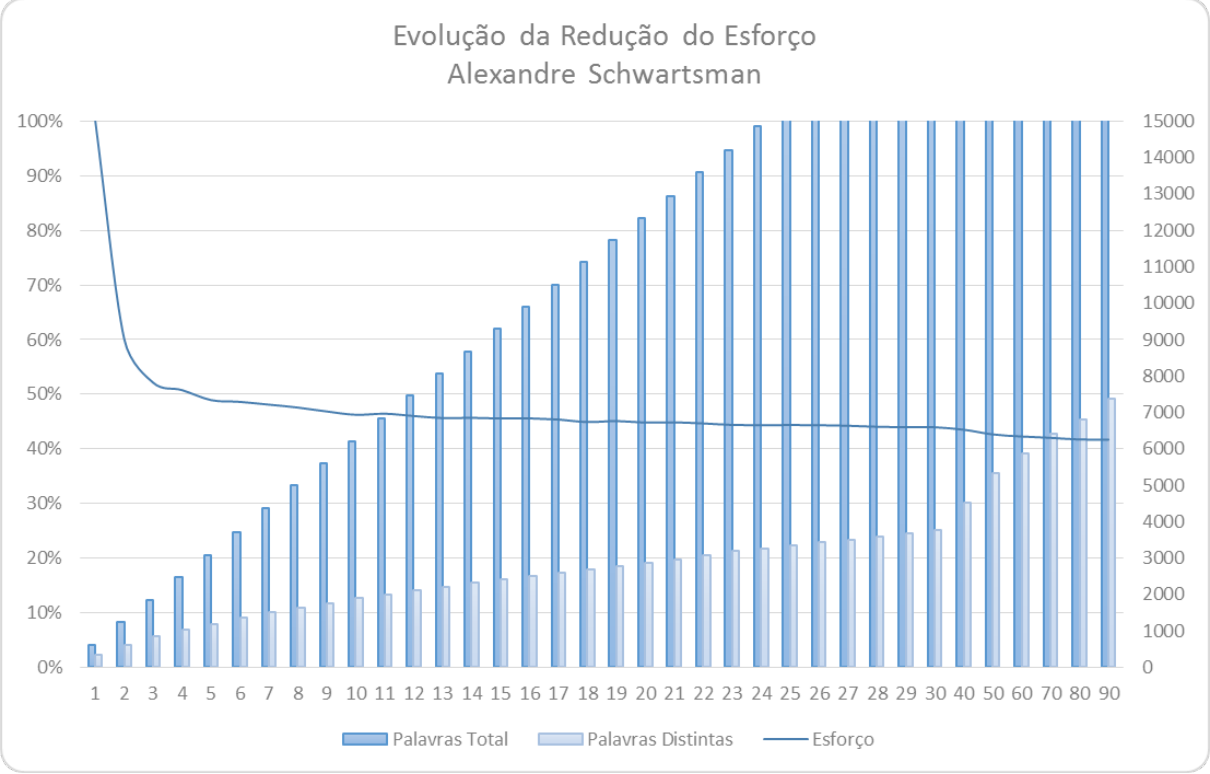


Figura 6.26 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Antonio Prata.

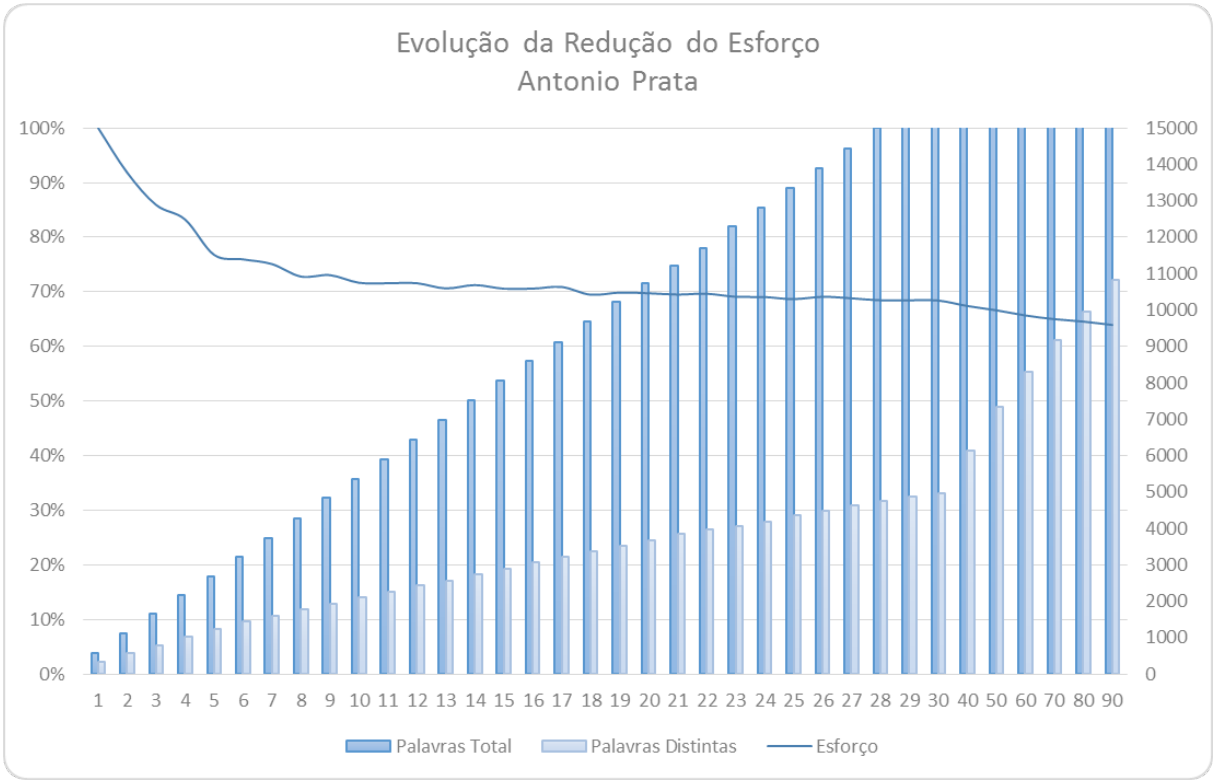


Figura 6.27 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Frederico Vasconcelos.

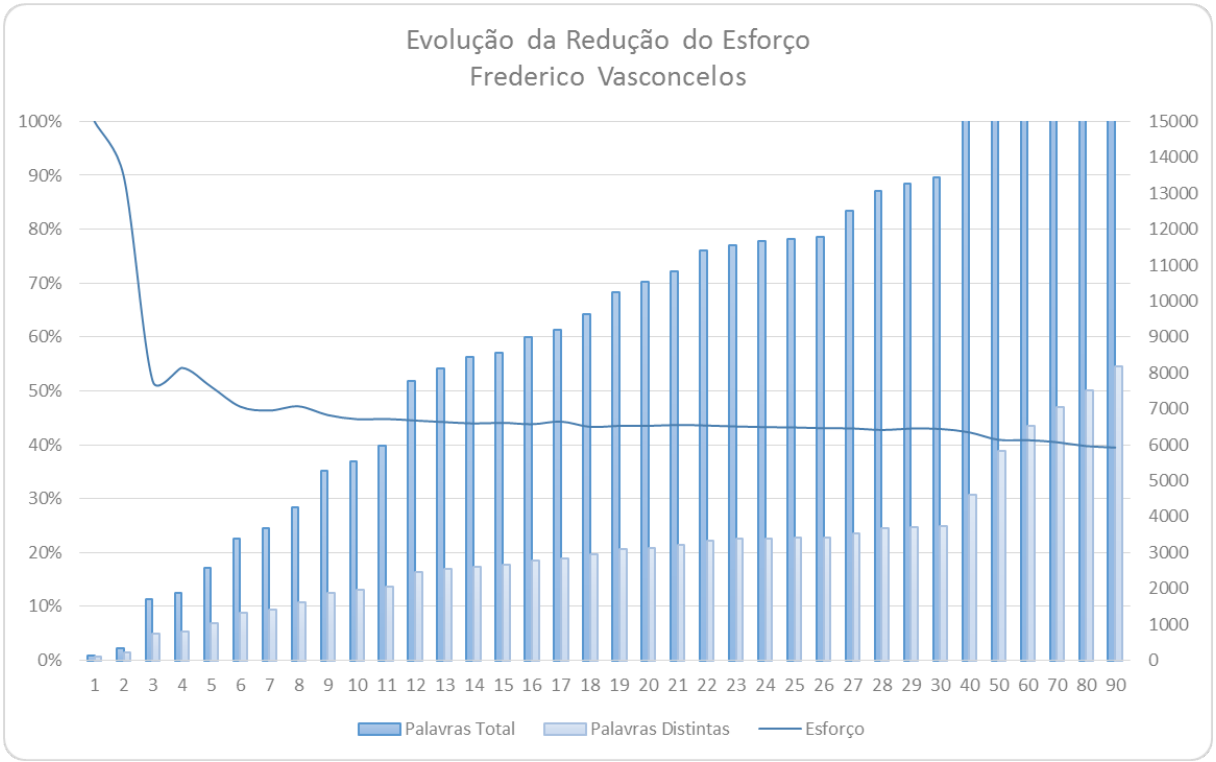


Figura 6.28 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Inácio Araujo.

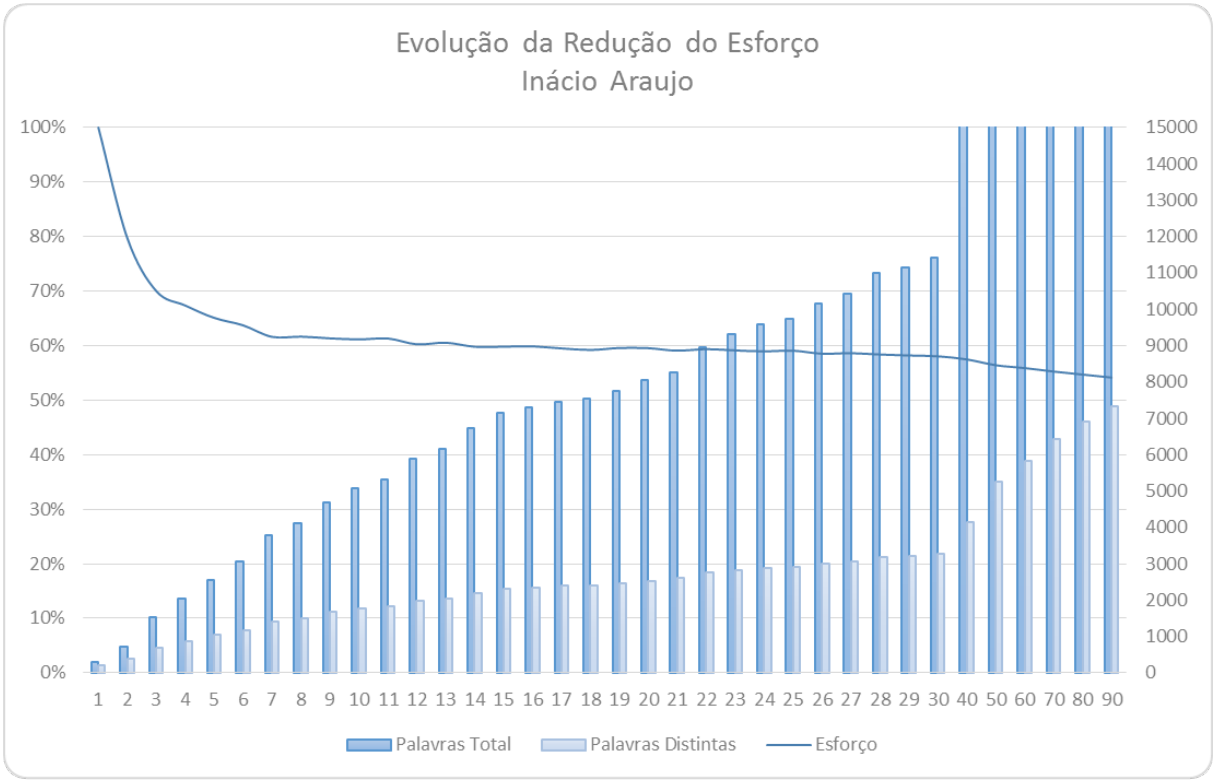


Figura 6.29 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Leandro Colon.

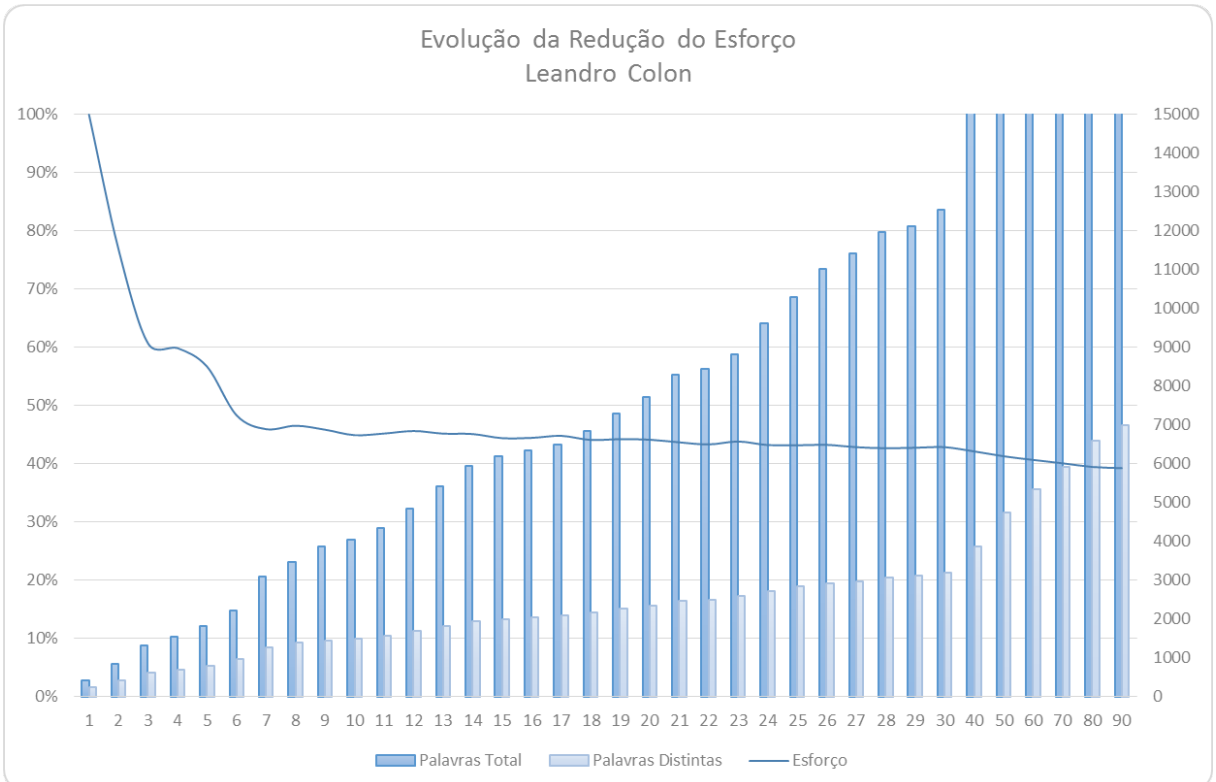


Figura 6.30 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Felipe Araujo.

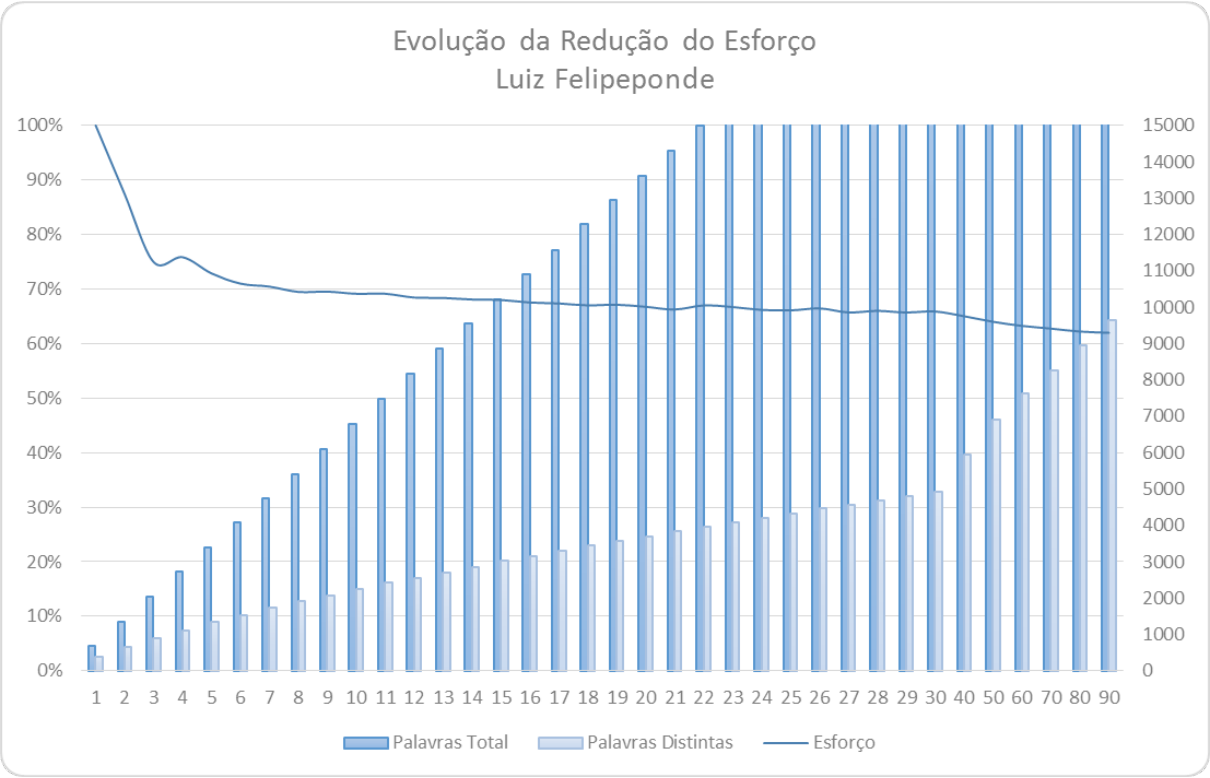


Figura 6.31 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Marcia Dessen

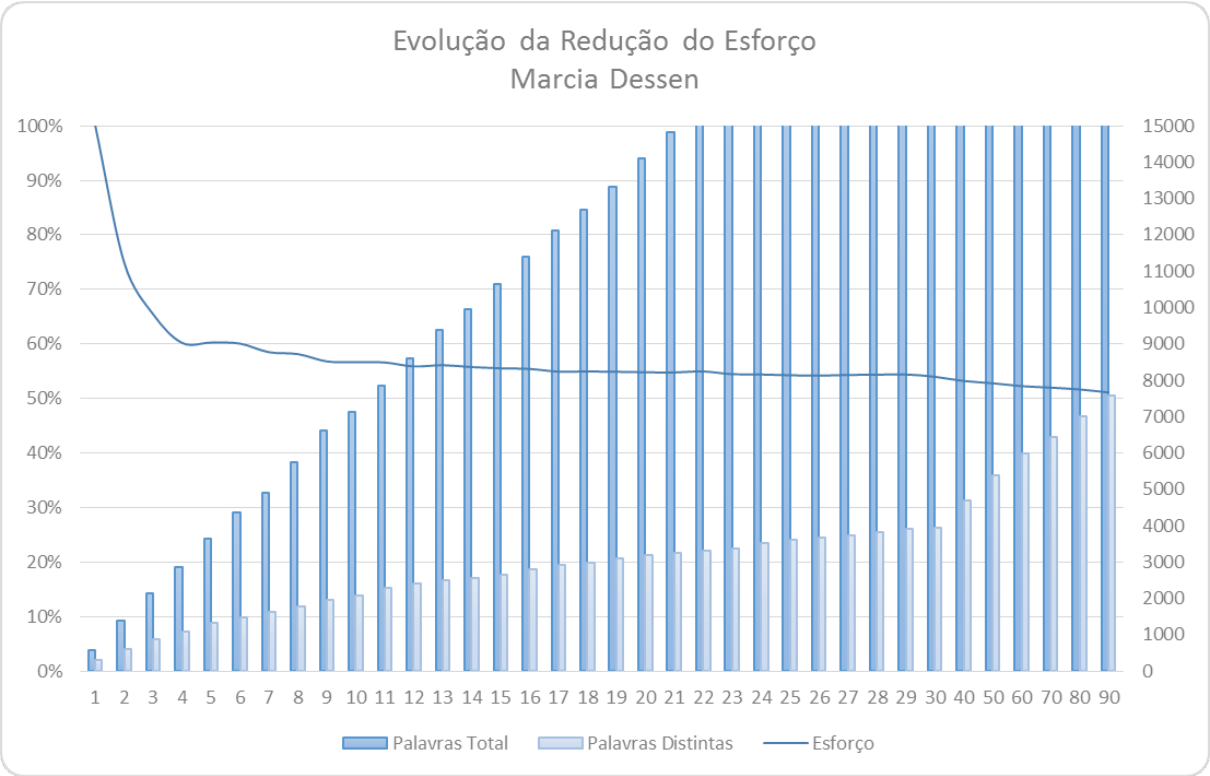


Figura 6.32 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Pedro Diniz.

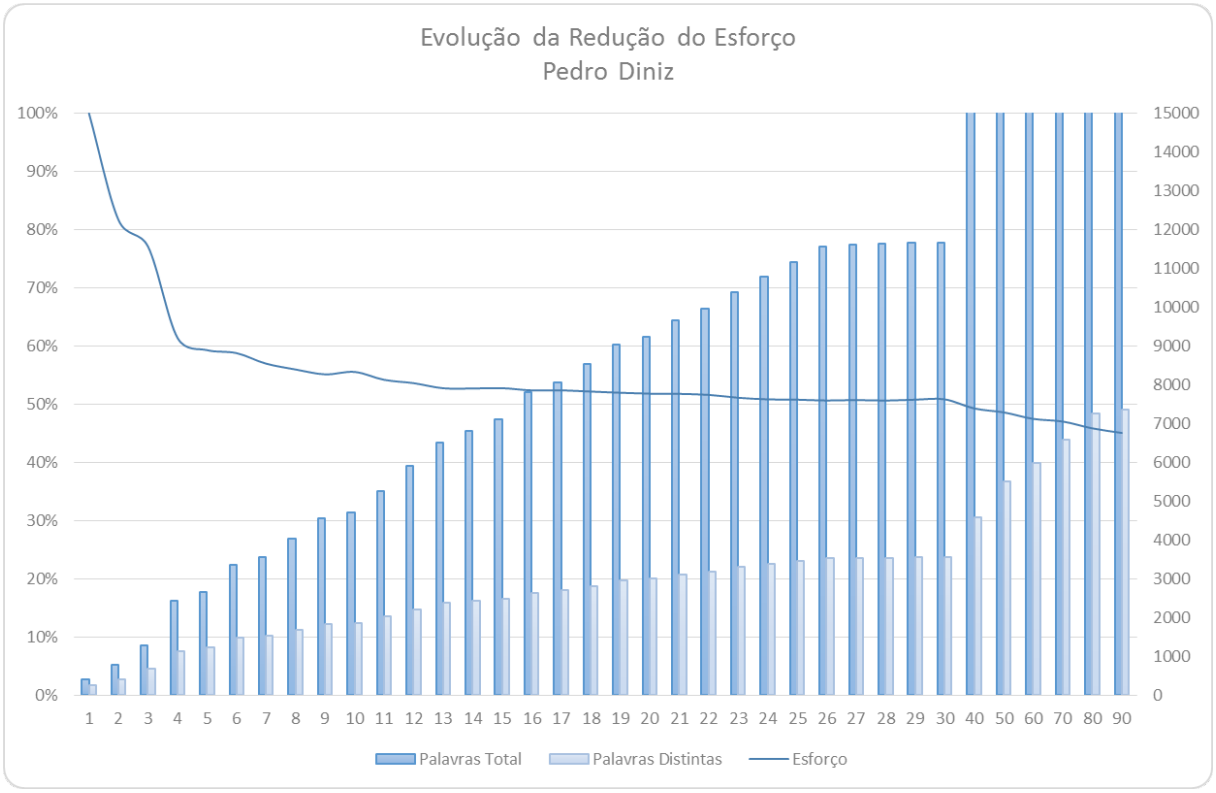


Figura 6.33 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Rodolfo Lucena.

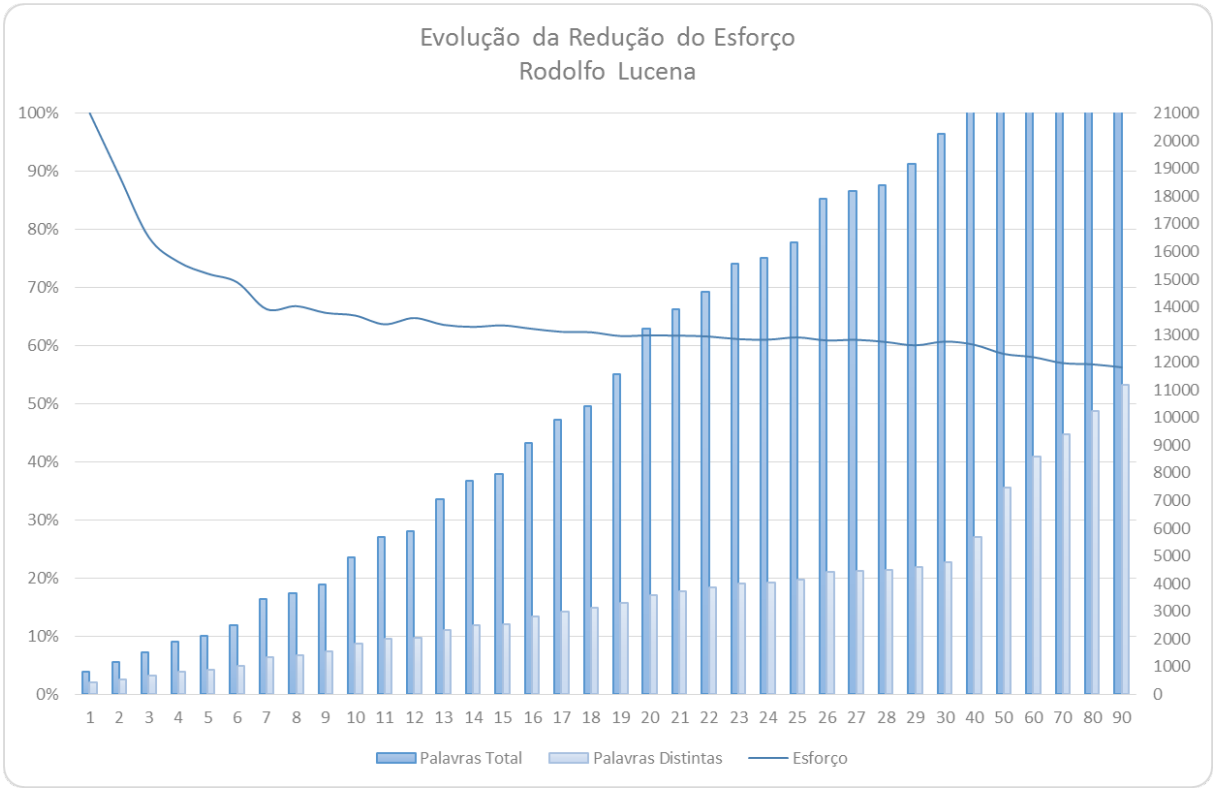
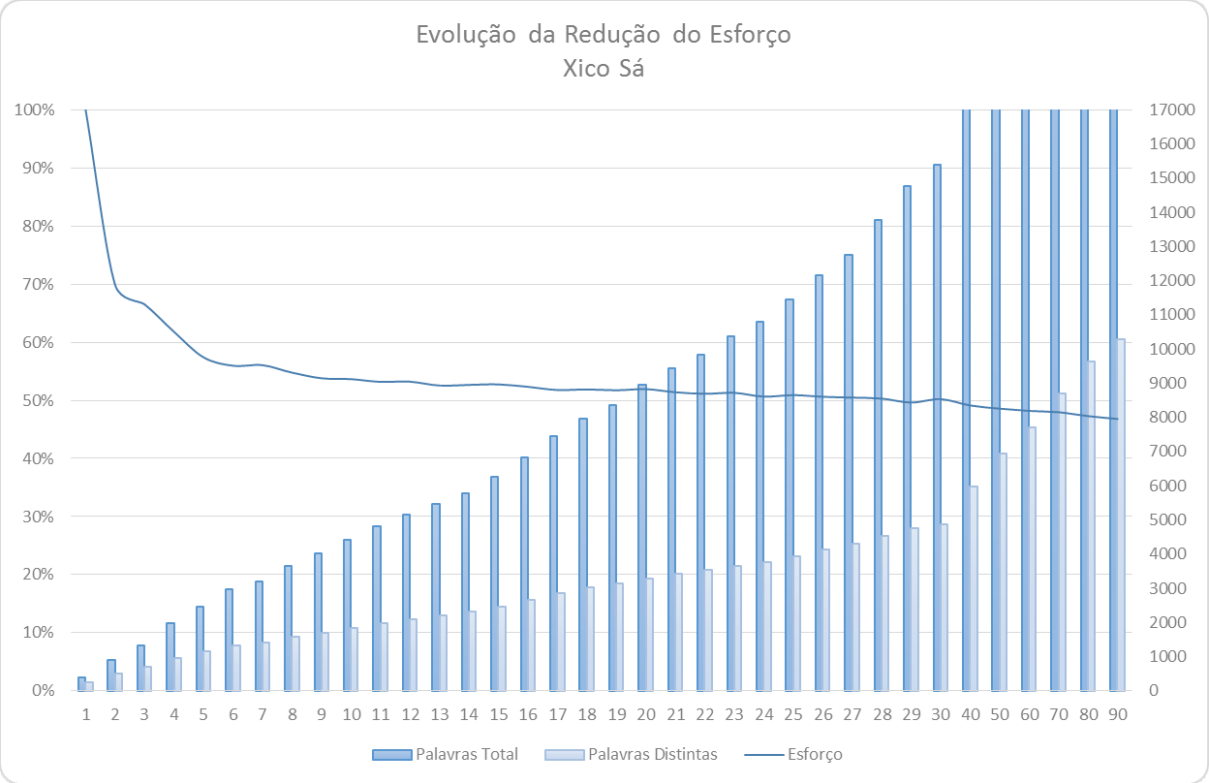


Figura 6.34 – Gráfico da evolução da redução do esforço de digitação durante a utilização da metodologia proposta aplicada aos textos de Xico Sá



Analisando os gráficos, das Figuras de 6.25 a 6.34, pode-se observar que as melhores otimizações foram entre os dez primeiros textos diminuindo o esforço de digitação em até 40%. Além disso, foi verificado que essa otimização ocorreu quando o usuário inseriu em média de 2800 a 3800 palavras dentre as quais 900 a 1500 eram distintas. Ao final das dez primeiras otimizações, a metodologia evolutiva de teclados assistivos estabizou. A partir desse ponto, o método proposto possibilitou uma diminuição do esforço de digitação de 1% a 5% em média a cada três mil a sete mil novas palavras inseridas no sistema.

Conclui-se que a princípio o usuário pode utilizar o método de otimização a cada 200 a 600 novas palavras inseridas. Esses números representam a média de palavras dos textos de cada autor. O paciente deve solicitar a otimização do teclado quando adicionar ao sistema uma média de três mil a sete mil palavras.

6.8 Conclusões

Neste capítulo foram realizados dois experimentos. O objetivo do primeiro era comprovar que a metodologia de evolução dos teclados virtuais é mais eficiente do que o método de otimização tradicional. O segundo experimento foi executado para determinar a quantidade de palavras necessárias para obter uma otimização efetiva.

Incialmente foram apresentadas as características do primeiro experimento, definiu-se as hipóteses, as variáveis dependentes e independentes, os sujeitos, a instrumentação, o projeto do experimento, as métricas e o planejamento da análise dos dados. Posteriormente, foi descrito o processo de execução do experimento e sua realização. Finalmente, foram analisados os dados coletados do experimento primeiro experimento. Posteriormente, foi descrito segundo experimento, seus objetivos, a seleção dos sujeitos escolhidos para esse experimento, o relato de execução do experimento e a análise dos dados objetidos.

Pode-se concluir que o método proposto nesta tese apresenta melhores resultados do que o método tradicional. Isso quer dizer que ao utilizar a metodologia proposta é possível que o usuário tenha uma redução considerável do esforço de digitação. Assim, o paciente que utilizar o teclado virtual desenvolvido neste trabalho pode digitar por mais tempo sem se fadigar. Além disso, outros estudos podem ser desenvolvidos para aprimorar a técnica. Durante os testes, esses estudos podem utilizar os métodos de predição ou até mesmo realizar um experimento a longo prazo com pessoas reais.

Outra conclusão importante do experimento foi a constatação da possibilidade da antagonia entre as funções objetivos de esforço de varredura e número de colisões. Os resultados dessas funções mostraram valores inversamente proporcionais. Pode-se supor que essa tendência é devido ao acumulo de letras mais utilizadas em uma mesma tecla. Para ter certeza dessa hipótese é necessário realizar novos experimentos com o objetivo de comprová-la.

A última conclusão do primeiro experimento afirma que é mais indicado que o teclado virtual aprimore o esforço de varredura em detrimento do número de colisões. Essa afirmação comprova a teoria proposta em (LESHER; MOULTON, 2000). Na qual o autor afirma que a otimização do esforço de varredura é uma característica importante para teclados virtuais que utilizam esse método para selecionar as teclas. O objetivo do segundo experimento foi identificar o número de palavras necessário para obter uma otimização utilizando a metodologia proposta. Pode-se concluir que inicialmente é vantajoso otimizar o teclado usando em média 200 a 600 palavras até que o sistema possua de três a quatro mil palavras inseridas. Após essa quantidade de palavras, a otimização é efetiva quando é inserida em média de três mil a sete mil palavras.

O próximo capítulo apresenta as conclusões e as contribuições desta tese e os trabalhos futuros.

CAPÍTULO 7

CONCLUSÕES E TRABALHOS FUTUROS

7.1 Introdução

Este capítulo apresenta as conclusões e os trabalhos futuros que poderão ser desenvolvidos a partir desta tese.

7.2 Conclusões

Este trabalho apresentou a modelagem e o desenvolvimento de um teclado virtual assistivo, compacto, otimizado e evolutivo para pessoas com SE. Para obter esse objetivo foram realizadas duas revisões sistemáticas e uma revisão de literatura.

O objetivo da primeira revisão sistemática foi pesquisar o estado da arte da Tecnologia Assistiva e da Comunicação Alternativa e Aumentativa, assim como, identificar e definir sistemas de CAA. A segunda revisão pesquisou sobre as características, os métodos de otimização e as métricas aplicadas a teclados virtuais. Finalmente, a revisão de literatura identificou os métodos existentes utilizados para otimizar a distribuição das letras entre as teclas do teclado virtual.

Após pesquisar os trabalhos literários, foi apresentada a proposta de um novo teclado virtual assistivo, compacto, otimizado e evolutivo. Identificou-se as características mais adequadas para serem utilizadas no novo teclado de acordo com a literatura. Além de modelar o teclado virtual assistivo para adequar-se a pessoas com SE, foram propostas novas técnicas de otimização do teclado virtual.

Inicialmente foi apresentado um novo método de desambiguidade parcial de letras. Esse método possibilita diminuir o número de colisões entre as palavras aumentando com o menor valor possível o número de interações entre o teclado e o usuário. Em seguida, foi descrito a metodologia evolutiva de teclados assistivos. Essa metodologia, diferente do método tradicional, considera o vocabulário e o modo de escrita do usuário para otimizar os *layouts* dos teclados. Outra vantagem da técnica proposta é a continuidade da evolução do *layout* e a possibilidade

de migração suavizada pelo método de sugestão de dissemelhança e testes de performance.

Finalmente, foram realizados dois experimentos. O objetivo do primeiro era mostrar a eficiência da métodologia de evolução dos teclados virtuais assistivos comparada com o método tradicional de otimização. Com esse experimento, pode-se concluir que o esforço de digitação utilizando os teclados gerados pelo método proposto foi de até 40% menor do que o método tradicional.

A finalidade do segundo experimento foi identificar o número de palavras necessários para obter uma otimização utilizando a metodologia proposta. Pode-se concluir que inicialmente é vantajoso otimizar o teclado usando em média 200 a 600 palavras até que o sistema possua de três a quatro mil palavras inseridas. Após essa quantidade de palavras, a otimização é efetiva quando é inserida em média de três mil a sete mil palavras.

7.3 Trabalhos Futuros

Para testar a metodologia de teclados assistivos foram realizados experimentos virtuais. É importante que esses experimentos sejam efetuados utilizando pacientes com SE.

Outro ponto importante é aprimorar os métodos de predição de texto usados pelo teclado virtual. Apesar desse teclado implementar diversos métodos, não foi implementado o método de predição utilizando gramática. Esse tipo de predição pode ser incluido ao teclado virtual permitindo que o sistema se torne mais robusto.

Além do teclado virtual assistivo e da métodologia de teclados virtuais proposta nesta tese foi desenvolvido um método de desambiguidade parcial. Para comprovar a efetividade deste método é importante que seja realizado um experimento científico. Esse experimento deve verificar o quanto o método proposto neste trabalho pode diminuir o esforço de digitação e aumentar a performance de entrada de dados.

Outro trabalho importante é desenvolver o restante do módulo de comunicação da arquitetura do sistema de CAA apresentada no capítulo 2. Para completar esse módulo é necessário que seja desenvolvida uma planilha de comunicação que integre o teclado virtual assistivo desenvolvido nesta tese.

7.4 Considerações finais

Este capítulo apresentou as conclusões deste trabalho e os trabalhos futuros que poderão ser desenvolvidos a partir desta tese.

REFERÊNCIAS

AL-ABDULLATIF, A. et al. Mind-controlled augmentative and alternative communication for people with severe motor disabilities. In: *Innovations in Information Technology (IIT), 2013 9th International Conference on*. [S.l.: s.n.], 2013. p. 107–112.

ANN, O. C.; THENG, L. B. A study on the effectiveness of biometrics based alternative communication tool. In: *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*. [S.l.: s.n.], 2011. p. 1–4.

ARBOLEDA, C. et al. P300-based brain computer interface experimental setup. In: *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. [S.l.: s.n.], 2009. p. 598–601. ISSN 1557-170X.

BAAS, M. et al. Système de saisie de texte visant à réduire l'effort des utilisateurs à handicap moteur. In: ACM. *Proceedings of the Ergonomie et Informatique Avancée Conference*. [S.l.], 2010. p. 19–26.

BAKER, B. R. Using images to generate speech. *BYTE*, McGraw-Hill, Inc., Hightstown, NJ, USA, v. 11, n. 3, p. 160–168, mar. 1986. ISSN 0360-5280. Disponível em: <http://dl.acm.org/citation.cfm?id=5874.5882>.

BALJKO, M.; TAM, A. Indirect text entry using one or two keys. *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility - Assets '06*, ACM Press, New York, New York, USA, p. 18, 2006. Disponível em: <http://portal.acm.org/citation.cfm?doid=1168987.1168992>.

BASILI, V. R.; ROMBACH, H. D. The tame project: Towards improvement-oriented software environments. *Software Engineering, IEEE Transactions on*, IEEE, v. 14, n. 6, p. 758–773, 1988.

BELATAR, M.; POIRIER, F. Text entry for mobile devices and users with severe motor impairments: Handiglyph, a primitive shapes based onscreen keyboard. In: *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*. New York, NY, USA: ACM, 2008. (Assets '08), p. 209–216. ISBN 978-1-59593-976-0. Disponível em: <http://doi.acm.org/10.1145/1414471.1414510>.

BERSCH, R. Introdução à tecnologia assistiva. *Porto Alegre: CEDI*, 2008.

BESIO, W.; KAY, S.; LIU, X. An optimal spatial filtering electrode for brain computer interface. In: *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. [S.l.: s.n.], 2009. p. 3138–3141. ISSN 1557-170X.

BHATTACHARYA, S.; BASU, A.; SAMANTA, D. Computational modeling of user errors for the design of virtual scanning keyboards. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, v. 16, n. 4, p. 400–9, ago. 2008. ISSN 1558-0210. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/18701386>.

BHATTACHARYA, S.; LAHA, S. Bengali text input interface design for mobile devices. *Universal Access in the Information Society*, v. 12, n. 4, p. 441–451, ago. 2012. ISSN 1615-5289. Disponível em: <http://link.springer.com/10.1007/s10209-012-0280-1>.

BISWAS, P.; SAMANTA, D. Friend: A communication aid for persons with disabilities. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, v. 16, n. 2, p. 205–209, 2008. ISSN 1534-4320.

BLAIN, S.; MIHAILIDIS, A.; CHAU, T. Assessing the potential of electrodermal activity as an alternative access pathway. *Medical Engineering & Physics*, v. 30, n. 4, p. 498–505, 2008.

BROUILLETTE, A.; SHARMA, U.; KALITA, J. Multi-objective optimization for efficient brahmic keyboards. In: *24th International Conference on Computational Linguistics*. [S.l.: s.n.], 2012. p. 29.

CALTENCO, H. A. et al. Understanding computer users with tetraplegia: survey of assistive technology users. *International Journal of Human-Computer Interaction*, Taylor & Francis, v. 28, n. 4, p. 258–268, 2012.

CARDWELL, M. et al. Locked-in syndrome. *Texas medicine*, v. 109, n. 2, p. e1–e1, 2012.

CHIN, S.; LEE, C. yeon; LEE, J. Facial expression image mapping for brain-computer interface using el type classification. In: *Information Sciences and Interaction Sciences (ICIS), 2010 3rd International Conference on*. [S.l.: s.n.], 2010. p. 465–469.

CIPRESSO, P. et al. The combined use of brain computer interface and eye-tracking technology for cognitive assessment in amyotrophic lateral sclerosis. In: *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th International Conference on*. [S.l.: s.n.], 2011. p. 320–324.

CLARKSON, P.; ROBINSON, A. Language model adaptation using mixtures and an exponentially decaying cache. In: *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*. [S.l.: s.n.], 1997. v. 2, p. 799–802 vol.2. ISSN 1520-6149.

COLAS, S. et al. Artificial ants for the optimization of virtual keyboard arrangement for disabled people. In: MONMARCHé, N. et al. (Ed.). *Artificial Evolution*. Springer Berlin Heidelberg, 2008, (Lecture Notes in Computer Science, v. 4926). p. 87–99. ISBN 978-3-540-79304-5. Disponível em: <http://dx.doi.org/10.1007/978-3-540-79305-2_8>.

COLKER, R. Americans with disabilities act: A windfall for defendants, the. *Harv. Cr-ClL Rev.*, HeinOnline, v. 34, p. 99, 1999.

COOK, A. M.; HUSSEY, S. Assistive technologies: Principles and practice. Mosby, 2001.

CROES, G. A. A method for solving traveling-salesman problems. *Operations Research*, INFORMS, v. 6, n. 6, p. 791–812, 1958.

DAWKINS, R. *A escalada do monte improvável: uma defesa da teoria da evolução*. [S.l.]: Editora Companhia das Letras, 1996.

DEB, K.; JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *Evolutionary Computation, IEEE Transactions on*, v. 18, n. 4, p. 577–601, Aug 2014. ISSN 1089-778X.

DEEPA, V.; THANGARAJ, P.; CHITRA, S. Investigating principal component analysis for classification of eeg data. In: *Networking and Information Technology (ICNIT), 2010 International Conference on*. [S.l.: s.n.], 2010. p. 461–464.

DICIONARIO-ABERTO.NET. *Dicionário Aberto*. 2015. Disponível em: <http://www.dicionario-aberto.net/>.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, IEEE, v. 1, n. 1, p. 53–66, 1997.

DOVAL, F. G.; CARBALLO, J. P.; JEREMIAS, J. V. Tictac: Information and communication technologies for augmentative communication boards. In: *Education Engineering (EDUCON), 2010 IEEE*. [S.l.: s.n.], 2010. p. 1783–1787.

DREWES, H. Eye gaze tracking for human computer interaction. 2010. Disponível em: <http://nbn-resolving.de/urn:nbn:de:bvb:19-115914>.

EGGERS, J. et al. Optimization of the keyboard arrangement problem using an ant colony algorithm. *European Journal of Operational Research*, Elsevier, v. 148, n. 3, p. 672–686, 2003.

EMILIANI, P. et al. A study of two-inputs scanning methods to enhance the communication rate. *Assistive Technology from Adapted Equipment to Inclusive Environments: AAATE 2009*, IOS Press, v. 25, p. 132, 2009.

ENGELBRECHT, A. P. *Computational intelligence: an introduction*. [S.l.]: John Wiley & Sons, 2007.

FARAJ, K. A.; MOJAHID, M.; VIGOUROUX, N. 3DKey : An Accordion-Folding Based Virtual Keyboard for Small Screen. p. 634–644, 2009.

FARAJ, K. A.; MOJAHID, M.; VIGOUROUX, N. BigKey : A Virtual Keyboard for Mobile Devices. n. 60, p. 3–10, 2009.

FARAJ, K. A.; VIGOUROUX, N. SmartKey : A Multi Purpose Target Expansion Based Virtual Keyboard. p. 251–252, 2009.

FAZLY, A. *The Use of Syntax in Word Completion Utilities*. 2002.

FELZER, T.; RINDERKNECHT, S. 3dScan : An Environment Control System Supporting Persons With Severe Motor Impairments. p. 213–214, 2009.

FITTS, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, v. 47, n. 6, p. 381–391, 1954.

FRANCIS, G.; JOHNSON, E. Speed – accuracy tradeoffs in specialized keyboards. *Journal of Human Computer Studies*, Elsevier, v. 69, n. 7-8, p. 526–538, 2011. ISSN 1071-5819. Disponível em: <http://dx.doi.org/10.1016/j.ijhcs.2011.04.002>.

FRANCIS, G.; OXTOBY, C. Building and testing optimized keyboards for specific text entry. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, SAGE Publications, v. 48, n. 2, p. 279–287, 2006.

FU, Y.-F.; HO, C.-S. A fast text-based communication system for handicapped aphasiacs. In: *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference on*. [S.l.: s.n.], 2009. p. 583–594.

GARAY-VITORIA, N.; ABASCAL, J. Text prediction systems: a survey. *Universal Access in the Information Society*, v. 4, n. 3, p. 188–203, dez. 2005. ISSN 1615-5289. Disponível em: <http://link.springer.com/10.1007/s10209-005-0005-9>.

GARCÍA, J. C. D.; FILHO, T. A. G. Pesquisa nacional de tecnologia assistiva. *São Paulo: ITS Brasil/MCTI-Secis*, 2012.

GELORMINI, D.; BISHOP, B. Optimizing the android virtual keyboard: A study of user experience. In: IEEE. *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*. [S.l.], 2013. p. 1–4.

GHOSH, S. Designing an Efficient Virtual Keyboard for Text Composition in Bengali. n. c, p. 84–87, 2011.

GHOSH, S. et al. Effective Virtual Keyboard Design with Size and Space Adaptation. n. April, p. 262–267, 2010.

GIVEN, L. M. *The Sage encyclopedia of qualitative research methods*. [S.l.]: Sage Publications, 2008.

GLOVER, F.; KOCHENBERGER, G. A. *Handbook of metaheuristics*. [S.l.]: Springer Science & Business Media, 2003.

GOETTL, J. S.; BRUGH, A. W.; JULSTROM, B. A. Call me e-mail: arranging the keyboard with a permutation-coded genetic algorithm. In: ACM. *Proceedings of the 2005 ACM symposium on Applied computing*. [S.l.], 2005. p. 947–951.

GOLDBERG, D. E. *Genetic algorithms in search, optimization, and machine learning*. New York: Addison-Wesley, 1989.

GOODMAN, J. et al. Language modeling for soft keyboards. *Proceedings of the 7th international conference on Intelligent user interfaces - IUI '02*, ACM Press, New York, New York, USA, p. 194, 2002. Disponível em: <http://portal.acm.org/citation.cfm?doid=502716.502753>.

GRANGE, A. L'interface du clavier virtuel chewing word. In: ACM. *Conference Internationale Francophone sur I'Interaction Homme-Machine*. [S.l.], 2010. p. 237–240.

GROVER, D.; KING, M.; KUSHLER, C. *Reduced keyboard disambiguating computer*. Google Patents, 1998. US Patent 5,818,437. Disponível em: <http://www.google.com/patents/US5818437>.

GUERRIER, Y. et al. Comparative Study between AZERTY-type and K- Hermes virtual Keyboards dedicated to Users with Cerebral Palsy. 2011.

HADKA, D. *MOEA Framework, a Java library for multiobjective evolutionary algorithms*. 2015. Disponível em: <http://www.moeaframework.org/>.

HANSON, E. K. et al. The impact of alphabet supplementation and word prediction on sentence intelligiblity of electronically distorted speech. *Speech Communication*, v. 52, n. 2, p. 99–105, 2010.

HARBUSCH, K.; KÜHN, M. Towards an adaptive communication aid with text input from ambiguous keyboards. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2*. [S.l.], 2003. p. 207–210.

HERRERO, D.; MONTEIRO, C. B. M. Verificação das habilidades funcionais e necessidades de auxílio do cuidador em crianças com paralisia cerebral nos primeiros meses de vida. *Revista brasileira de crescimento e desenvolvimento humano*, scielopepsic, v. 18, p. 163 – 169, 08 2008. ISSN 0104-1282.

HICK, W. E. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, Taylor & Francis, v. 4, n. 1, p. 11–26, 1952.

HILL, K. *Advances in Augmentative and Alternative Communication as Quality-of-Life Technology*. 2010. 43–58 p.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. [S.l.]: The University of Michigan Press, 1975.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* [S.l.]: U Michigan Press, 1975.

HOSTE, L.; SIGNER, B. SpeeG2 : A Speech- and Gesture-based Interface for Efficient Controller-free Text Entry. 2013.

HYMAN, R. Stimulus information as a determinant of reaction time. *Journal of experimental psychology*, American Psychological Association, v. 45, n. 3, p. 188, 1953.

IBM. *IBM SPSS Software - Brasil*. 2015. Disponível em: <http://www-01.ibm.com/software/br/analytics/spss/>.

INSTITUTO BRASILéIRO DE GEOGRAFIA E ESTATíSTICA. Censo demográfico 2010. Rio de Janeiro, RJ, 2010. Disponível em: <http://www.ibge.gov.br/home/presidencia/noticias/imprensa/ppts/00000008473104122012315727483985.pdf>. Acesso em: 01 out. 2013.

JAIN, S.; BHATTACHARYA, S. Virtual Keyboard Layout Optimization. n. April, p. 312–317, 2010.

JANPINIJRUT, S.; NATTEE, C.; KAYASITH, P. An approach for improving thai text entry on touch screen mobile phones based on distance and statistical language model. Springer, p. 44–55, 2011.

JOHNSON, J. M. et al. Perspectives of speech language pathologists regarding success versus abandonment of aac. *Augmentative and Alternative Communication*, Informa UK Ltd UK, v. 22, n. 2, p. 85–99, 2006.

JONG, K. A. D. *Evolutionary computation: a unified approach*. [S.l.]: MIT press, 2006.

JOSHI, A. et al. Design and evaluation of Devanagari virtual keyboards for touch screen mobile phones. *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11*, ACM Press, New York, New York, USA, p. 323, 2011. Disponível em: <http://dl.acm.org/citation.cfm?doid=2037373.2037422>.

KEEGAN, J.; BURKE, E.; CONDRON, J. An electrooculogram-based binary saccade sequence classification (bssc) technique for augmentative communication and control. In: *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. [S.l.: s.n.], 2009. p. 2604–2607. ISSN 1557-170X.

KENNEDY, J. Particle swarm optimization. In: *Encyclopedia of Machine Learning*. [S.l.]: Springer, 2010. p. 760–766.

KIM, J. H. et al. Differences in typing forces, muscle activity, comfort, and typing performance among virtual, notebook, and desktop keyboards. *Applied ergonomics*, Elsevier Ltd, v. 45, n. 6, p. 1406–13, nov. 2014. ISSN 1872-9126. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/24856862>.

KIRKPATRICK, S.; VECCHI, M. et al. Optimization by simmulated annealing. *science*, Washington, v. 220, n. 4598, p. 671–680, 1983.

KNUTH, D. E. Dynamic huffman coding. *Journal of algorithms*, Elsevier, v. 6, n. 2, p. 163–180, 1985.

KNUTH, D. E. *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1998. ISBN 0-201-89685-0.

KRASKOWSKY, L. H.; FINLAYSON, M. Factors affecting older adults' use of adaptive equipment: Review of the literature. *The American Journal of Occupational Therapy*, The American Occupational Association, Inc., v. 55, n. 3, p. 303–310, 2001.

KROLAK, A.; STRUMILLO, P. Vision-based eye blink monitoring system for human-computer interfacing. In: *Human System Interactions, 2008 Conference on*. [S.l.: s.n.], 2008. p. 994–998.

KRUG, P. D. P. B. C.; AMARAL, K. M. Esclerose lateral amiotrófica. novembro 2002. Disponível em: <http://www.saudedireta.com.br/docsupload/1332009057ELA.pdf>.

KUBLER, A. et al. Bci meeting 2005-workshop on clinical issues and applications. *IEEE Transactions on neural systems and rehabilitation engineering*, IEEE, v. 14, n. 2, p. 131, 2006.

KWON, S.; LEE, D.; CHUNG, M. K. Effect of key size and activation area on the performance of a regional error correction method in a touch-screen QWERTY keyboard. *International Journal of Industrial Ergonomics*, Elsevier Ltd, v. 39, n. 5, p. 888–893, set. 2009. ISSN 01698141. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0169814109000481>.

LADANY, S. P. A model for optimal design of keyboards. *Computers Operations Research*, v. 2, n. 1, p. 55 – 59, 1975. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/0305054875900271>.

LESHER, G.; MOULTON, B.; HIGGINBOTHAM, D. J. Techniques for augmenting scanning communication. *Augmentative and Alternative Communication*, Informa UK Ltd UK, v. 14, n. 2, p. 81–101, 1998.

LESHER, G. W.; MOULTON, B. J. A method for optimizing single-finger keyboards. In: ERIC. *Proceedings of the RESNA 2000 Annual Conference*. [S.l.], 2000. p. 91–93.

LESHER, G. W.; MOULTON, B. J.; HIGGINBOTHAM, D. J. Optimal character arrangements for ambiguous keyboards. *Rehabilitation Engineering, IEEE Transactions on*, IEEE, v. 6, n. 4, p. 415–423, 1998.

LEVINE, S. H.; TREPAGNIER, C. Customised text entry devices for motor-impaired users. *Applied ergonomics*, Elsevier, v. 21, n. 1, p. 55–62, 1990.

LIGHT, L.; ANDERSON, P. Designing better keyboards via simulated annealing. Miller Freeman Publishers, 1993.

LOJA, L. F. B. et al. A concept-environment for computer-based augmentative and alternative communication founded on a systematic review. *Research on Biomedical Engineering*, 2015.

LOPES, M. C. S. *Mineração de dados textuais utilizando técnicas de clustering para o idioma português*. Tese (Doutorado) — UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2004.

MACKENZIE, I. S. Kspc (keystrokes per character) as a characteristic of text entry techniques. In: *Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*. London, UK, UK: Springer-Verlag, 2002. (Mobile HCI '02), p. 195–210. ISBN 3-540-44189-1. Disponível em: <http://dl.acm.org/citation.cfm?id=645739.666587>.

MACKENZIE, I. S. Chapter 4 - evaluation of text entry techniques. In: MACKENZIE, I. S.; TANAKA-ISHII, K. (Ed.). *Text Entry Systems*. Burlington: Morgan Kaufmann, 2007. p. 75 – 101. ISBN 978-0-12-373591-1. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780123735911500048>.

MACKENZIE, I. S. et al. Letterwise: prefix-based disambiguation for mobile text input. In: *UIST'01*. [S.l.: s.n.], 2001. p. 111–120.

MACKENZIE, I. S.; ZHANG, S. X. The design and evaluation of a high-performance soft keyboard. *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*, ACM Press, New York, New York, USA, n. May, p. 25–31, 1999. Disponível em: <http://portal.acm.org/citation.cfm?doid=302979.302983>.

MACKENZIE, I. S.; ZHANG, S. X.; SOUKOREFF, R. W. Text entry using soft keyboards. *Behaviour & information technology*, Taylor & Francis, v. 18, n. 4, p. 235–244, 1999.

MAJARANTA, P. et al. Auditory and visual feedback during eye typing. In: *CHI '03 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2003. (CHI EA '03), p. 766–767. ISBN 1-58113-637-4. Disponível em: <http://doi.acm.org/10.1145/765891.765979>.

MAK, J.; WOLPAW, J. Clinical applications of brain-computer interfaces: Current state and future prospects. *Biomedical Engineering, IEEE Reviews in*, v. 2, p. 187–199, 2009.

MAK, J. N.; WOLPAW, J. R. Clinical applications of brain-computer interfaces: current state and future prospects. *Biomedical Engineering, IEEE Reviews in*, IEEE, v. 2, p. 187–199, 2009.

MASON, C.; CHINN, K. Augmentative alternative communication. *International Encyclopedia of Education*, p. 544–549, 2010.

MASSEY, F. J. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, American Statistical Association, v. 46, n. 253, p. 68–78, 1951. Disponível em: <http://www.jstor.org/stable/2280095>.

MCCOY, K. F. et al. *Speech and Language processing as assistive technologies*. 2013. 1143–1146 p.

MELANIE, M. An introduction to genetic algorithms. *Cambridge, Massachusetts London, England, Fifth printing*, v. 3, 1999.

MERINO, M. et al. Assessment of biosignals for managing a virtual keyboard. In: MIESENBERGER, K. et al. (Ed.). *Computers Helping People with Special Needs*. Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7383). p. 331–337. ISBN 978-3-642-31533-6. Disponível em: <http://dx.doi.org/10.1007/978-3-642-31534-3_50>.

METROPOLIS, N. et al. Equation of state calculations by fast computing machines. *The journal of chemical physics*, AIP Publishing, v. 21, n. 6, p. 1087–1092, 1953.

MICHAELIS.UOL.COM.BR. *Dicionário Português Online: Moderno Dicionário da Língua Portuguesa - Michaelis - UOL*. 2015. Disponível em: <http://michaelis.uol.com.br/moderno/portugues/index.php>.

MICHALEWICZ, Z. *Genetic algorithms+ data structures= evolution programs*. [S.l.]: Springer Science & Business Media, 1996.

MILLET, B.; ASFOUR, S.; LEWIS, J. R. Selection-based virtual keyboard prototypes and data collection application. *Behavior research methods*, v. 41, n. 3, p. 951–6, ago. 2009. ISSN 1554-351X. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/19587212>.

MIRó, J.; BERNABEU, P. Text entry system based on a minimal scan matrix for severely physically handicapped people. In: MIESENBERGER, K. et al. (Ed.). *Computers Helping People with Special Needs*. Springer Berlin Heidelberg, 2008, (Lecture Notes in Computer Science, v. 5105). p. 1216–1219. ISBN 978-3-540-70539-0. Disponível em: <http://dx.doi.org/10.1007/978-3-540-70540-6_183>.

MIRÓ-BORRÁS, J. et al. Ambiguous keyboards and scanning: The relevance of the cell selection phase. In: *Human-Computer Interaction–INTERACT 2009*. [S.l.]: Springer, 2009. p. 1–4.

MIRó-BORRáS, J. et al. Text Entry in the E-Commerce Age : Two Proposals for the Severely Handicapped. v. 4, n. 1, p. 101–112, 2009.

MOLINA, A. et al. Evaluation of unambiguous virtual keyboards with character prediction. *Assistive Technology from Adapted Equipment to Inclusive Environments: AAATE 2009*, IOS Press, v. 25, p. 144, 2009.

MOLINA, a. J.; RIVERA, O.; GóMEZ, I. Measuring Performance of Virtual Keyboards Based on Cyclic Scanning. *2009 Fifth International Conference on Autonomic and Autonomous Systems*, Ieee, p. 174–178, 2009. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4976600>.

NICOLAU, H. et al. Mobile text-entry and visual demands: reusing and optimizing current solutions. *Universal Access in the Information Society*, v. 13, n. 3, p. 291–301, set. 2013. ISSN 1615-5289. Disponível em: <http://link.springer.com/10.1007/s10209-013-0319-y>.

NORMAN, D.; RUMELHART, D. Studies of typing from the lnr research group. In: COOPER, W. (Ed.). *Cognitive Aspects of Skilled Typewriting*. Springer New York, 1983. p. 45–65. ISBN 978-1-4612-5472-0. Disponível em: <http://dx.doi.org/10.1007/978-1-4612-5470-6_3>.

NUANCE.COM. *Nuance*. 2015. Disponível em: <http://www.nuance.com/news/pressreleases/2007/20070824_tegic.asp>.

NUANCE.COM. *XT9 | XT9 Smart Input is a multi-modal text input solution that supports a variety of device form factors and input methods. | Nuance*. 2015. Disponível em: <http://www.nuance.com/for-business/by-product/xt9/index.htm>.

OKASAKA, S.; HOSHINO, Y. Development of estimation method about activity states for nirs-based bci system. In: *Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint 6th International Conference on*. [S.l.: s.n.], 2012. p. 1144–1149.

OOMMEN, B. J.; VALIVETI, R. S.; ZGIERSKI, J. A fast learning automaton solution to the keyboard optimization problem. In: *Proceedings of the 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems - Volume 2*. New York, NY, USA: ACM, 1990. (IEA/AIE '90), p. 981–990. ISBN 0-89791-372-8. Disponível em: <http://doi.acm.org/10.1145/98894.99108>.

ORENGO, V.; HUYCK, C. A stemming algorithmm for the portuguese language. In: IEEE. *spire*. [S.l.], 2001. p. 0186.

ORHAN, U. et al. Rsvp keyboard: An eeg based typing interface. In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. [S.l.: s.n.], 2012. p. 645–648. ISSN 1520-6149.

PANWAR, P.; SARCAR, S.; SAMANTA, D. Eyeboard: A fast and accurate eye gaze-based text entry system. *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, Ieee, p. 1–8, dez. 2012. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6481793>.

PARDALOS, P. M.; RENDL, F.; WOLKOWICZ, H. The quadratic assignment problem: A survey and recent developments. In: *In Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, volume 16 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. [S.l.: s.n.], 1994.

PARK, S.-W. et al. Augmentative and alternative communication training using eye blink switch for locked-in syndrome patient. *Annals of Rehabilitation Medicine*, v. 36, n. 2, p. 268–272, 2012.

POLácEK, O.; MíKOVEC, Z.; SLAVíK, P. Predictive scanning keyboard operated by hissing. In: *Proceedings of the 2nd IASTED International Conference Assistive Technologies*. [S.l.]: IASTED, 2012. (AT 2012), p. 862–869.

POUPLIN, S. et al. Effect of a dynamic keyboard and word prediction systems on text input speed in patients with functional tetraplegia. *Journal of Rehabilitation Research and Development*, v. 51, n. 3, p. 467–480, 2014.

PRABHU, V.; PRASAD, G. Designing a virtual keyboard with multi-modal access for people with disabilities. *2011 World Congress on Information and Communication Technologies*, Ieee, p. 1133–1138, dez. 2011. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6141407>.

PRZEDBORSKI, S.; VILA, M.; JACKSON-LEWIS, V. Series introduction: Neurodegeneration: What is it and where are we? *The Journal of Clinical Investigation*, The American Society for Clinical Investigation, v. 111, n. 1, p. 3–10, 1 2003. Disponível em: <http://www.jci.org/articles/view/17522>.

RAYNAL, M.; VIGOUROUX, N. Genetic algorithm to generate optimized soft keyboard. In: ACM. *CHI'05 extended abstracts on Human factors in computing systems*. [S.l.], 2005. p. 1729–1732.

SADLER, T.; LANGMAN, J. *Langman. Embriología médica: con orientación clínica*. Editorial Medica Panamericana Sa de, 2007. ISBN 9789500600774. Disponível em: <http://books.google.com.br/books?id=KEw1RjJQ6hEC>.

SARCAR, S. et al. Virtual Keyboard Design : State of the Arts and Research Issues. n. April, p. 289–299, 2010.

SARDINHA, T.; FILHO, J. M.; ALAMBERT, E. 2014. Disponível em: <http://corpusbrasileiro.pucsp.br/>.

SASSAKI, R. K. Nada sobre nós sem nós: da integração à inclusão. *Rev. Nac. Reabil*, v. 10, n. 57, p. 8–16, 2007.

SCHADLE, I. Sibyl: Aac system using nlp techniques. In: MIESENBERGER, K. et al. (Ed.). *Computers Helping People with Special Needs*. Springer Berlin Heidelberg, 2004, (Lecture Notes in Computer Science, v. 3118). p. 1009–1015. ISBN 978-3-540-22334-4. Disponível em: <http://dx.doi.org/10.1007/978-3-540-27817-7_149>.

SCHALK, G. et al. Brain computer interfaces (bcis): Detection instead of classification. *Journal of Neuroscience Methods*, v. 167, n. 1, p. 51–62, 2008.

SCHULZ, A. G. Fuzzy rule-based expert systems and genetic machine learning. *Physica-Verlag*, 1997.

SCHWARTZMAN, J. S. Paralisia cerebral. *Temas sobre Desenvolvimento Edição Especial*, p. 13, 1993.

SHANG, H.; MERRETTAL, T. Tries for approximate string matching. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 8, n. 4, p. 540–547, 1996.

SHANNON, C. E. Prediction and entropy of printed english. *Bell system technical journal*, Wiley Online Library, v. 30, n. 1, p. 50–64, 1951.

SHARMA, M. K. et al. Parameters Effecting the Predictive Virtual Keyboard. n. April, p. 268–275, 2010.

SHARMA, M. K. et al. Visual clue: An approach to predict and highlight next character. *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, Ieee, p. 1–7, dez. 2012. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6481820>.

SIK-LÁNYI, C.; MOLNÁR-LÁNYI, Á. Psychological and pedagogic testing of handicapped children with locomotion disorder using multimedia programs. In: *3rd International Conference on Disability, Virtual Reality and Associated Technologies, Alghero, Sardinia, Italy*. [S.l.: s.n.], 2000. p. 99–106.

SILFVERBERG, M.; MACKENZIE, I. S.; KORHONEN, P. Predicting text entry speed on mobile phones. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2000. (CHI '00), p. 9–16. ISBN 1-58113-216-6. Disponível em: <http://doi.acm.org/10.1145/332040.332044>.

SILVA, F.; PEREIRA, F. Communication between people with motion and speech disabilities. In: *Information Systems and Technologies (CISTI), 2011 6th Iberian Conference on*. [S.l.: s.n.], 2011. p. 1–4.

SIMATHAMANAND, J.; PIROMSOPA, K. Performance optimization of Thai virtual keyboard for social networking. *2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Ieee, p. 210–213, maio 2011. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5930122>.

SIMPSON, R. Making better decisions [modeling the assistive technology assessment]. *Engineering in Medicine and Biology Magazine, IEEE*, v. 27, n. 2, p. 23–28, 2008. ISSN 0739-5175.

SMITH, E. Locked-in syndrome. 2004. Disponível em: <http://www.bmj.com/content/330/7488/406?variant=full-text>.

SÖRENSEN, K. Multi-objective optimization of mobile phone keymaps for typing messages using a word list. *European Journal of Operational Research*, Elsevier, v. 179, n. 3, p. 838–846, 2007.

SORGER, B. et al. *Another kind of BOLD Response: answering multiple-choice questions via online decoded single-trial brain signals*. [S.l.: s.n.], 2009. v. 177. 275–292 p.

SOUKOREFF, R. W.; MACKENZIE, I. S. Measuring Errors in Text Entry Tasks: An Application of the Levenshtein String Distance Statistic. In: *CHI '01 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2001. (CHI EA '01), p. 319–320. ISBN 1-58113-340-5. Disponível em: <http://dx.doi.org/10.1145/634067.634256>.

SUN, G.; HU, J.; WU, G. A novel frequency band selection method for common spatial pattern in motor imagery based brain computer interface. In: *Neural Networks (IJCNN), The 2010 International Joint Conference on*. [S.l.: s.n.], 2010. p. 1–6.

TANAKA-ISHII, K.; INUTSUKA, Y.; TAKEICHI, M. Entering text with a four-button device. *Proceedings of the 19th international conference on Computational linguistics* -, Association for Computational Linguistics, Morristown, NJ, USA, v. 1, n. 1, p. 1–7, 2002. Disponível em: <http://portal.acm.org/citation.cfm?doid=1072228.1072377>.

TANAKA-ISHII, K.; INUTSUKA, Y.; TAKEICHI, M. Entering text with a four-button device. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. [S.l.], 2002. p. 1–7.

TéCNICAS, B. C. de A. dezembro 2007. Disponível em: <http://www.mj.gov.br/corde/arquivos/doc/Ata_VII_Reuni~ao_do_Comite_de_Ajudas_Técnicas.doc>.

THOMAS, K. et al. An adaptive filter bank for motor imagery based brain computer interface. In: *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. [S.l.: s.n.], 2008. p. 1104–1107. ISSN 1557-170X.

TOPAL, C.; BENLIGIRAY, B. On The Efficiency Issues of Virtual Keyboard Design. n. 111, 2012.

TRUONG, C. T. Collaborative Smart Virtual Keyboard. p. 513–522, 2013.

USAKLI, A. B.; GURKAN, S. Design of a Novel Efficient Human – Computer Interface : An Electrooculagram Based Virtual Keyboard. p. 1–10, 2009.

USAKLI, A. B. et al. A hybrid platform based on eog and eeg signals to restore communication for patients afflicted with progressive motor neuron diseases. In: *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. [S.l.: s.n.], 2009. p. 543–546.

VARCHOLIK, P. D.; LAVIOLA, J. J.; HUGHES, C. E. Establishing a baseline for text entry for a multi-touch virtual keyboard. *International Journal of Human-Computer Studies*, Elsevier, v. 70, n. 10, p. 657–672, out. 2012. ISSN 10715819. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1071581912000961>.

WANDMACHER, T. et al. Sibylle, an assistive communication system adapting to the context and its user. *ACM Trans. Access. Comput.*, ACM, New York, NY, USA, v. 1, n. 1, p. 6:1–6:30, maio 2008. ISSN 1936-7228. Disponível em: <http://doi.acm.org/10.1145/1361203.1361209>.

WARD, D. J.; BLACKWELL, A. F.; MACKAY, D. J. Dasher—a data entry interface using continuous gestures and language models. In: ACM. *Proceedings of the 13th annual ACM symposium on User interface software and technology*. [S.l.], 2000. p. 129–137.

WILKINSON, K. M.; HENNIG, S. The state of research and practice in augmentative and alternative communication for children with developmental/intellectual disabilities. *Mental Retardation and Developmental Disabilities Research Reviews*, Wiley Online Library, v. 13, n. 1, p. 58–69, 2007.

WOBBROCK, J. O. Chapter 3 - measures of text entry performance. In: MACKENZIE, I. S.; TANAKA-ISHII, K. (Ed.). *Text Entry Systems*. Burlington: Morgan Kaufmann, 2007. p. 47 – 74. ISBN 978-0-12-373591-1. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780123735911500036>.

WOHLIN, C. et al. *Experimentation in software engineering*. [S.l.]: Springer Science & Business Media, 2012.

WU, F.-G.; HUANG, Y.-C.; WU, M.-L. New chording text entry methods combining physical and virtual buttons on a mobile phone. *Applied ergonomics*, Elsevier Ltd, v. 45, n. 4, p. 825–32, jul. 2014. ISSN 1872-9126. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/24263072>.

XPERTKEYBOARD.COM. *XPeRT Keyboarding - Touch Typing Speeds without Training*. 2015. Disponível em: <http://www.xpertkeyboard.com/>.

YAMADA, H. *A Historical Study of Typewriters and Typing Methods, from the Position of Planning Japanese Parallels*. Journal of Information Processing, 1980. Disponível em: <http://books.google.com.br/books?id=KtWuGwAACAAJ>.

YIN, P.-Y.; SU, E.-P. Cyber swarm optimization for general keyboard arrangement problem. *International Journal of Industrial Ergonomics*, Elsevier, v. 41, n. 1, p. 43–52, 2011.

ZHAI, S.; HUNTER, M.; SMITH, B. A. The metropolis keyboard-an exploration of quantitative techniques for virtual keyboard design. In: ACM. *Proceedings of the 13th annual ACM symposium on User interface software and technology*. [S.l.], 2000. p. 119–128.

ZHAI, S.; HUNTER, M.; SMITH, B. a. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. *Proceedings of the 13th annual ACM symposium on User interface software and technology - UIST '00*, ACM Press, New York, New York, USA, n. UIST, p. 119–128, 2000. Disponível em: <http://portal.acm.org/citation.cfm?doid=354401.354424>.

ZHAI, S.; SMITH, B. A. Alphabetically biased virtual keyboards are easier to use: layout does matter. In: ACM. *CHI'01 extended abstracts on Human factors in computing systems*. [S.l.], 2001. p. 321–322.