CIÊNCIAS DA VIDA

ATHENEUS
coleção acadêmica

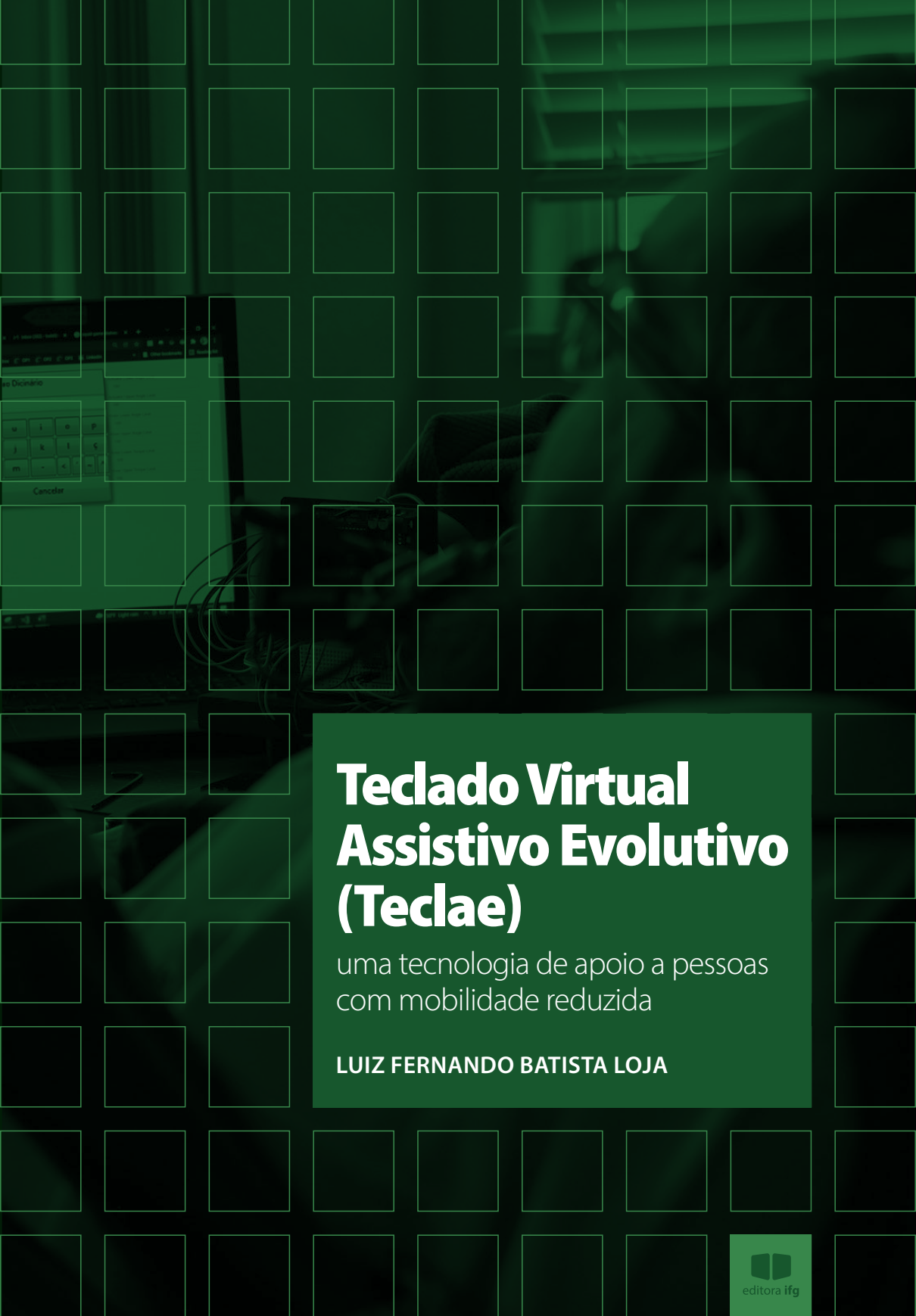
Teclado Virtual Assistivo Evolutivo (Teclae)

uma tecnologia de apoio a pessoas com mobilidade reduzida

LUIZ FERNANDO BATISTA LOJA

editora ifg

A Tecnologia Assistiva proporciona formas, meios, recursos e métodos que visam melhorar a qualidade de vida das pessoas com deficiência ou mobilidade reduzida. 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). As pessoas com SE ficam totalmente restritas e em fases mais avançadas não são capazes de se comunicar 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. Neste livro, o/a leitor/a conhecerá a proposta de 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.
Boa leitura!



Teclado Virtual Assistivo Evolutivo (Teclae)

uma tecnologia de apoio a pessoas com mobilidade reduzida

LUIZ FERNANDO BATISTA LOJA

editora ifg

ATHENEUS
coleção acadêmica

CIÊNCIAS DA VIDA

Teclado Virtual Assistivo Evolutivo (Teclae)

uma tecnologia de apoio a pessoas com mobilidade reduzida

LUIZ FERNANDO BATISTA LOJA

editora ifg

ISBN 978-85-67022-80-2

L835t	Loja, Luiz Fernando Batista Teclado Virtual Assistivo Evolutivo (Teclae): uma tecnologia de apoio a pessoas com mobilidade reduzida/ Luiz Fernando Batista Loja. -- Goiânia: Editora IFG, 2024. 200 p.: il. – (Coleção Atheneus) ISBN 978-85-67022-80-2 (impresso) ISBN 978-85-67022-70-3 (digital) 1. Tecnologia assistiva. 2. Comunicação aumentativa e alternativa. 3. Teclado virtual assistivo. I. Loja, Luiz Fernando Batista. II. Título. III. Coleção. CDD 004.81

Catalogação na Publicação:
Maria Aparecida Andrade de Oliveira Tsu – Bibliotecária-documentalista – CRB-1/1604

Instituto Federal de Educação, Ciência e Tecnologia de Goiás

Editora IFG

Avenida C-198, Qd. 500. Jardim América.
Goiânia/GO | CEP. 74270–040.
(62) 3612-2251
editora@ifg.edu.br

Sumário

Prefácio

Inicio o texto dizendo da minha honra e satisfação em prefaciar a obra de Luiz Fernando Batista Loja, que resulta de uma sólida trajetória formativa.

Em 2012 dei início ao meu processo de formação em nível de pós-doutoramento na Faculdade de Engenharia Elétrica, da Universidade Federal de Uberlândia (UFU), sob supervisão da Profa. Edna Lúcia Flôres. Em parceria com meu amigo Francisco Ramos de Melo, Chico, também em formação no pós-doutorado, iniciamos um trabalho sobre aplicação de Inteligência Artificial (IA) no campo das Tecnologias Assistivas (TA). Para tanto, reunimos um grupo de pessoas interessadas em aprofundar sobre essa perspectiva de estudo. Foi nesse momento que conheci o Luiz Loja, um daqueles gênios da área de computação, cheio de energia e vontade de fazer a diferença.

Havíamos, até então, na perspectiva das TA, desenvolvido uma planilha de comunicação tendo como referência uma experiência vivenciada com uma pessoa com Esclerose Lateral Amiotrófica (ELA), uma doença severa que afeta o sistema nervoso e, de forma degenerativa e progressiva, acarreta paralisia de forma irreversível, causando tetraplegia e limitação da fala.

Muito embora a planilha tenha apresentado resultado prático, possibilitando comunicação efetiva, ela carecia de aperfeiçoamentos, além de uma migração para um formato digital e integrado a um sistema computacional. No coletivo, projetamos a possibilidade de transformar a planilha, até então em formato físico, em um sistema inteligente, dotado de IA, dentro da modalidade de TA denominada de Comunicação Aumentativa e Alternativa (CAA).

Nesse desafio, buscamos compreender o que era mais urgente, essencial, e percebemos que o aprofundamento sobre o teclado da planilha se apresentava como ação principal a ser desenvolvida. Para uma pessoa que se comunica apenas pelos olhos, como é o caso de pessoas com ELA,

em nossa avaliação o teclado se caracterizava como principal forma de entrada de dados.

A partir de tal desafio, sob a orientação da Profa. Lúcia Flôres, Luiz Loja ingressou em seu processo de doutoramento na UFU, já em 2013, com o propósito de desenvolver um teclado virtual assistivo, com uso de metodologia evolutiva, de forma que o teclado a ser desenvolvido pudesse se adaptar ao vocabulário e ao modo de escrita de cada usuário.

A ideia do trabalho proposto era interessante, inovadora, com aplicação imediata e resultado social prático. Mas, devagar com o andor que o santo é de barro, ou, como se diz aqui em Goiás, não se pode botar o carro na frente dos bois. No campo da ciência, o método é base de todo o processo.

Para o desenvolvimento do teclado proposto, era necessário, antes, realizar duas etapas do processo investigativo: o levantamento de toda produção bibliográfica sobre o TA e CAA – processo denominado de estado da arte – e a realização de uma rigorosa revisão sistemática da literatura, que consistia na busca por trabalhos acadêmico-científicos desenvolvidos sobre o tema, com exposição de técnicas e modelos adotados, características, os métodos de otimização e as métricas aplicadas a teclados virtuais. A primeira tinha como foco o aprofundamento teórico-metodológico do estudo, enquanto a segunda, mais complexa e (in)tensa, possibilitaria avaliar a viabilidade do estudo, a partir do levantamento do ponto de tese, bem como permitiria, na sequência, compreender qual o grau e nível de inovação a proposta de trabalho apresentava.

Depois de um intenso trabalho de organização da pesquisa, em banca de qualificação, dentre os diferentes estudos correlatos analisados, a sua proposta se apresentou como sólida, consistente e inovadora. Fatores que lhe deram condições de avançar para a etapa seguinte do trabalho: desenvolvimento do sistema de otimização do teclado. Em essência, a ideia, de forma simplificada, consistia no desenvolvimento de um teclado – que a posteriori seria acessado por meio de um dispositivo de entrada (uma "câmera inteligente") para captura de sinais voluntários/intencionais (sinais bióticos) – que, ao ser acionado pelo usuário, reconhecesse palavras, a partir de uma

antecipação máxima (predição), minimizando o número de interações necessárias para a escrita de uma palavra, fator que reduziria o difícil esforço de digitação realizado pelo usuário, ampliando a sua performance.

E eis que, em 2015, Luiz Loja, o menino prodígio, chegou lá! Depois de um conjunto de publicações sólidas e reconhecimento da comunidade científica do seu feito, ele defendeu sua tese de doutorado. O garoto que outrora arquitetava softwares de jogos e fazia bicos desenvolvendo plataformas online e pequenos projetos para empresas se formou doutor e tornou-se professor de uma importante instituição pública de educação superior de Goiás, o Instituto de Educação, Ciência e Tecnologia de Goiás. Tudo fruto da sua inteligência, dedicação e humildade, mas sem se esquecer de sua querida e importante orientadora e seus colaboradores. Além de ser disposto, dinâmico, alegre, Luiz é também uma pessoa parceira e que sabe trabalhar em equipe.

O livro em apresentação é uma síntese dessa trajetória, uma oportunidade de o público mais amplo (re)conhecer o fruto de todo esse esforço individual e trabalho coletivo produzido. Segue o convite para a leitura desta obra, resultado de uma densa pesquisa, que conduz uma necessária aproximação da academia e da ciência com a sociedade, possibilitando a todos, todas e todes o acesso a importantes e sólidas contribuições científicas e encaminhamentos para o avanço da inclusão social.

Ricardo Antonio Gonçalves Teixeira
PROFESSOR DO PROGRAMA DE PÓS-GRADUAÇÃO EM EDUCAÇÃO, DA FACULDADE DE EDUCAÇÃO DA UNIVERSIDADE FEDERAL DE GOIÁS.

Introdução

Até o ano 2000, mais de 10% de toda a 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 — têm algum tipo de deficiência. Dessas pessoas, 7% tinham alguma restrição motora (IBGE, 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, 2009). Além de aumentar sua independência, reduz seu isolamento social e minimiza 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 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 cerebrais 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 (FU; HO, 2009; GARCIA DOVAL;POUSADA CARBALLO; VEZ JEREMIAS, 2010; 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 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;

SARCAR; SAMANTA, 2011). Entretanto, em comparação ao seu correlato físico sabe-se que esse tipo de solução possui uma baixa performance de digitação (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 a 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, segundo 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 (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.

Neste livro, apresentamos o desenvolvimento de 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.

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 à 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 trinta palavras por minuto, se operados por digitadores experientes (SIMATHAMANAND; PIROMSOPA, 2011; VARCHOLIK; LAVIOLA; HUGHES, 2012; HOSTE; SIGNER, 2013). Os teclados virtuais assistivos inserem em média de quatro a sete palavras por minuto, quando utilizados por uma pessoa com deficiência motora (MIRÓ-BORRÁS; BERNABEU-SOLER, 2009). Logo, 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 com a utilização de um sistema que se adapte às suas necessidades, foi a principal motivação da pesquisa de que resultou este livro.

1 Tecnologia Assistiva e Comunicação Aumentativa e Alternativa

Atualmente a sociedade vem pesquisando tecnologias e técnicas com a finalidade de auxiliar a inclusão social de indivíduos com deficiência (GALVÃO FILHO; GARCÍA, 2012). Essa tendência originou uma nova área de conhecimento denominada Tecnologia Assistiva (TA). Cook, Polgar e Encarnação (2014) definem TA mencionando o conceito elaborado por Colker (1999). Essa definição determina que TA é uma ampla gama de equipamentos, serviços, estratégias e práticas concebidas e aplicadas para minorar os problemas funcionais encontrados por indivíduos com deficiências.

Tecnologia Assistiva e Comunicação Aumentativa e Alternativa

Uma subárea 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 agudos o paciente é privado de todos os seus movimentos e da capacidade de fala, caracterizando a síndrome do encarceramento (SE).

O objetivo dos ambientes de comunicação de CAA é permitir a comunicação das pessoas portadoras de SE com os seus semelhantes e com

o mundo externo. Esses ambientes são compostos por métodos e *softwares* que possibilitam essa comunicação.

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 bengala improvisada, caracteriza o uso de um recurso de TA (GALVÃO FILHO; GARCÍA, 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 constrói 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, podem também causar tetraplegia e restringir 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 1998 como *American with Disabilities Act* (ADA). A tradução de *assistive technology* para a expressão tecnologia assistiva foi sugerida por Sassaki (1996). Essa tradução serviu para substituir diferentes denominações adotadas até então e, além disso, unificou a terminologia referente aos mais diversos suportes que as pessoas com deficiência necessitam. O autor definiu tecnologia assistiva como:

> A tecnologia destinada a fornecer suporte (mecânico, elétrico, eletrônico, computadorizado etc.) às 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, 1996, p. 2).

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 à participação de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social. (BRASIL, 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:

- Tratamento médico pessoal;
- Treinamento de habilidades;
- Órteses e próteses;
- Proteção e cuidados pessoais;
- Mobilidade pessoal;
- Cuidados com o lar;
- Mobiliário e adaptações para residências e outras edificações;
- Comunicação e informação;
- Manuseio de objetos e equipamentos;
- Melhorias ambientais, ferramentas e máquinas; e
- 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.

Outra definição para as subáreas de TA é realizada pela *Horizontal European Activities in Rehabilitation Technology* (Heart). A Heart divide TA em seis 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 (GALVÃO FILHO; GARCÍA, 2012). O Quadro 1 mostra essa classificação.

Quadro 1 – Classificação HEART

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

Fonte: Galvão Filho e García (2012).

Diferente das classificações apresentadas, José Tonolli e Rita Bersch descreveram 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. O Quadro 2 apresenta as subáreas de TA, de acordo com a Classificação Nacional de Tecnologia Assistiva do Departamento de Educação dos Estados Unidos.

Devido à interdisciplinaridade e à 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 livro é utilizada a classificação proposta por José Tonolli e Rita Bersh (BERSCH, 2008). Essa definição é voltada à área acadêmica e abrange o conceito de Comunicação Aumentativa e Alternativa (CAA).

Quadro 2 – Subáreas da TA

Subárea	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.

Fonte: Galvão Filho e García (2012).

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). Segundo Park *et al.* (2012), 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.

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 (MAK; WOLPAW, 2009; SORGER *et al.*, 2009). 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, 2013).

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 mau funcionamento das células do sistema nervoso de forma excessiva e progressiva de determinadas áreas do cérebro. Portanto, pacientes com 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, Piconand 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, há casos de sobrevida longa. Sua incidência é estimada em 1 a 2,5 indivíduos portadores a cada 100 mil habitantes por ano, com uma prevalência de 2,5 a 8,5 por 100 mil 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, 2004). 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 e 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), propõe-se 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 *software* são *componentes secundários*. Os *componentes terciários* mostram a relação entre a adequação e a expectativa do usuário ao utilizar o sistema de comunicação.

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 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 1 mostra esse modo de representação da linguagem.

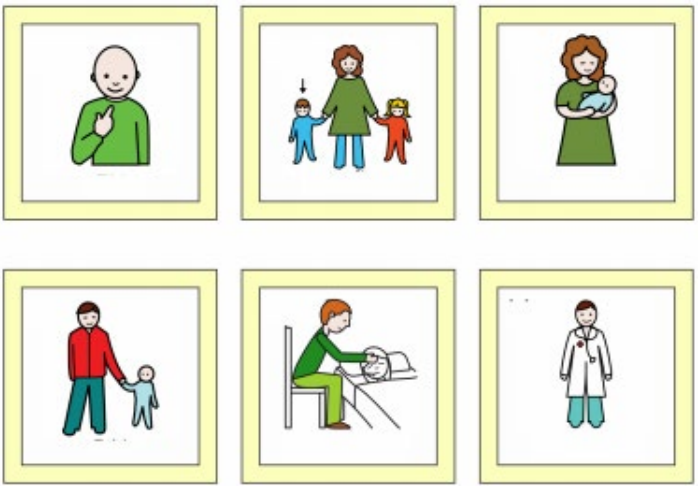


Figura 1 – Modo de representação da linguagem usando apenas figuras
Fonte: Elaborada pelo autor.

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 ilustra esse tipo de comunicação representada por um teclado virtual comum.

Figura 2 – Modo de representação da linguagem utilizando apenas o alfabeto, números e caracteres especiais

Fonte: Elaborada pelo autor.

A representação por compactação semântica ou ícones de multisignificados foi proposta 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 de sistemas com este tipo de representação não precisam ser alfabetizados.

Nos trabalhos de Arboleda *et al.* (2009) e Usakli *et al.* (2009), os autores adotaram uma estratégia híbrida, utilizando tanto imagens quanto caracteres para substituir a linguagem "natural". As figuras 3 e 4 mostram as propostas de comunicação de Usakli *et al.* (2009) e Arboleda *et al.* (2009), respectivamente.

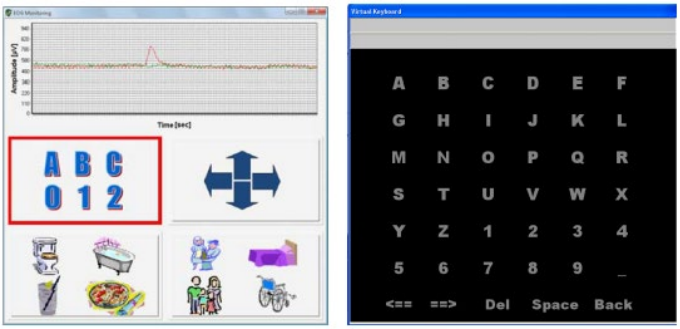


Figura 3 – Modo híbrido de representar a linguagem proposto por Usakli *et al.* (2009) com símbolos, números e letras

Fonte: Usakli *et al.* (2009).

Na planilha de comunicação desenvolvida por Usakli *et al.* (2009), as setas podem ser utilizadas para mover o mouse, ao passo que as figuras permitem ao usuário uma comunicação mais rápida e as letras proporcionam uma comunicação mais detalhada. O usuário pode escolher as ações entre as várias imagens e escrever textos mais complexos utilizando o teclado virtual. Essa abordagem também permite controlar o *mouse* por meio das setas, que representam a direção do movimento.

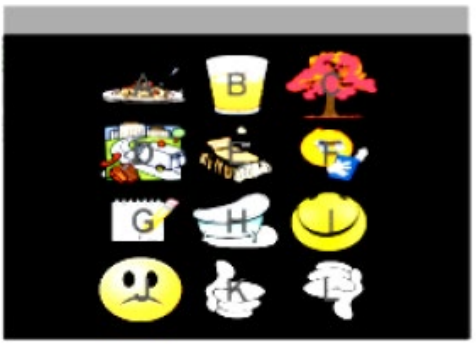


Figura 4 – Interface proposta por Arboleda *et al.* (2009) para representar a linguagem com símbolos e letras

Fonte: Arboleda *et al.* (2009).

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.

Por seu turno, Garcia Doval, Pousada Carballo e Vez Jeremias (2010), Prabhu e Prasad (2011) e Park *et al.* (2012) usaram somente o alfabeto para representar a linguagem "natural".

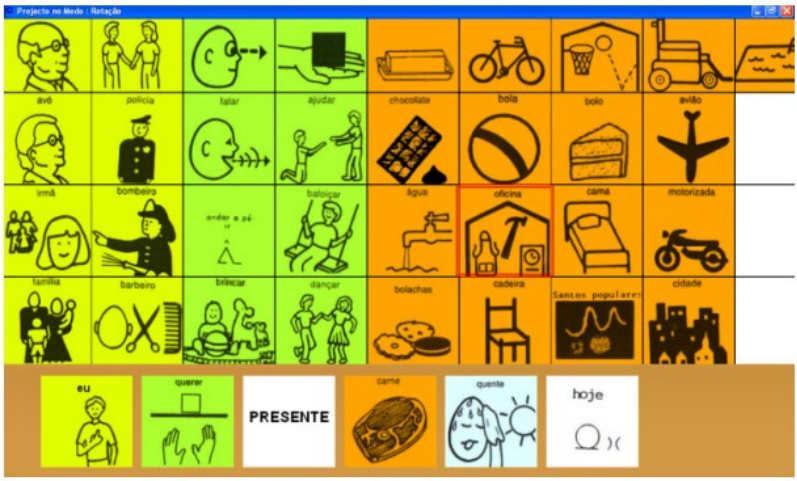


Figura 5 – Interface com compactação semântica proposta por Silva e Pereira (2011)
Fonte: Silva e Pereira (2011).

A Figura 5 mostra a interface proposta por Silva e Pereira (2011), cujo modo de representação permite construir sentenção usando apenas figuras. De acordo com os autores, a linguagem "natural" foi representada por compactação semântica. Essas imagens são organizadas em uma matriz de símbolos e descrições. Os símbolos representam alguma informação ou comando que o usuário pode selecionar de modo a definir o que deseja transmitir para o receptor, escolhendo um elemento dessa matriz e compondo as palavras e/ou a frase.

Componentes secundários

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

O dispositivo de entrada é o equipamento usado para capturar qualquer tipo de intenção voluntária do paciente. As pessoas portadoras de SE para

comunicarem-se por meio desses dispositivos estão aptas a emitirem cinco tipos de sinais biológicos que podem ser convertidos em comandos computacionais. Os 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 eletro-oculograma (EOG) (USAKLI *et al.*, 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 *et al.* (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 Lau (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 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 dispositivo de comunicação que interpreta uma fala degradada por processador de voz e gera como saída uma expressão sintetizada.

Ann e Lau (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; DELARGY, 2005). 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 minicâmera acoplada a um óculos para capturar o piscar dos olhos dos pacientes portadores de SE. Para interagir com a interface proposta por Park *et al.* (2012), é necessário uma calibração do *software* de reconhecimento de piscadas. A Figura 6 ilustra o dispositivo usado no trabalho desses autores, ressaltando que o usuário pode interagir com o sistema por meio de suas piscadas.

Figura 6 – Dispositivo de entrada adaptado a um óculos proposto por Park *et al.* (2012)
Fonte: Park *et al.* (2012).

O trabalho de Panwar, Sarcar e Samanta (2012) tentou solucionar os problemas relacionados à 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 à tela do computador que possibilitam o *mouse* se movimentar pelo olhar. A Figura 7 ilustra o uso do dispositivo de entrada.

Figura 7 – Dispositivo de entrada proposto por Panwar, Sarcar e Samanta (2012) com o *software* ITU Gazetracker

Fonte: Panwar, Sarcar e Samanta (2012).

Usakli *et al.* (2009) propuseram uma abordagem híbrida que utiliza o eletro-oculograma (EOG) e o eletroencefalograma (EEG) para facilitar a comunicação de pacientes com SE. A finalidade do EOG é capturar os potenciais entre a córnea e a retina. Esse potencial é utilizado como estímulo 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 aos 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, 2009).

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, 2009). Durante a pesquisa de que resultou este livro, foram encontrados trabalhos relacionados às atividades de operação do BCI (SCHALK *et al.*, 2008; THOMAS *et al.*, 2008; BESIO; KAY; LIU, 2009; CHIN; YEON LEE; LEE, 2010; DEEPA; THANGARAJ; CHITRA, 2010; SUN; HU; WU, 2010; OKASAKA; HOSHINO, 2012), bem como aos 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 Usakli *et al.* (2009) e Cipresso *et al.* (2011) foram usados o movimento dos olhos e o BCI. Todos esses dispositivos transmitem sinais que devem ser decodificados e posteriormente transformados em informações. 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; GARCIA DOVAL; POUSADA CARBALLO; VEZ 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.

Garcia Doval, Pousada Carballo e Vez 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.

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 (KÜBLER *et al.*, 2006). Para realizar essas etapas é necessário 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 outro. Assim, os sistemas de CAA devem possuir interfaces simples, intuitivas e que possam se adaptar a cada usuário.

Proposta de ambiente para a Comunicação Aumentativa e Alternativa

Os artigos e documentos pesquisados apresentaram uma ou mais contribuições na área de Comunicação Aumentativa e Alternativa. Alguns trabalhos mostraram melhorias nos componentes primários, outros apresentaram aperfeiçoamentos nos componentes secundários e poucos se destacaram nos componentes terciários.

Um ambiente de comunicação adaptável, robusto e de baixo custo 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 8 ilustra a interação do usuário com o ambiente de comunicação proposto e as suas funcionalidades.

Com o objetivo de aumentar a velocidade de comunicação do paciente com o meio externo, o *módulo de comunicação* mostrado na Figura 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. Objetiva-se, assim, analisar, modelar, codificar e validar esse teclado, que será acoplado ao módulo de comunicação desse ambiente.

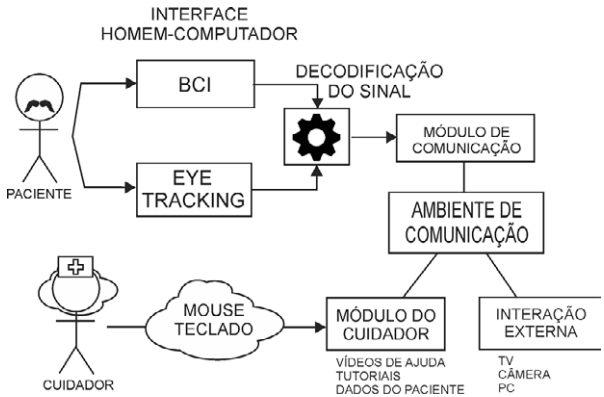


Figura 8 – Arquitetura do sistema de CAA proposto por Loja *et al.* (2015)
Fonte: Loja *et al.* (2015).

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 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

Teclados virtuais

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. As respostas a essas três perguntas são essenciais para a modelagem e a construção do Teclado Virtual Assistisvo Evolutivo (Teclae), apresentado neste livro, 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 esforço de digitação alto e 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. Em seguida, a finalidade da terceira pergunta é identificar quais são as métricas que avaliam a performance de comunicação dos teclados.

Neste capítulo, além da resposta às três perguntas fundamentais para a pesquisa de que resultou este livro, há a descrição das principais características de um teclado virtual e a exposição dos métodos e técnicas de otimização da performance de digitação, bem como as métricas utilizadas para avaliar essa performance.

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 característica do teclado virtual. Assim, determinou-se o conceito de *característica do teclado virtual* como: qualquer atributo pertencente a um teclado virtual essencial para 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 9 mostra essas características e os valores de cada uma delas.

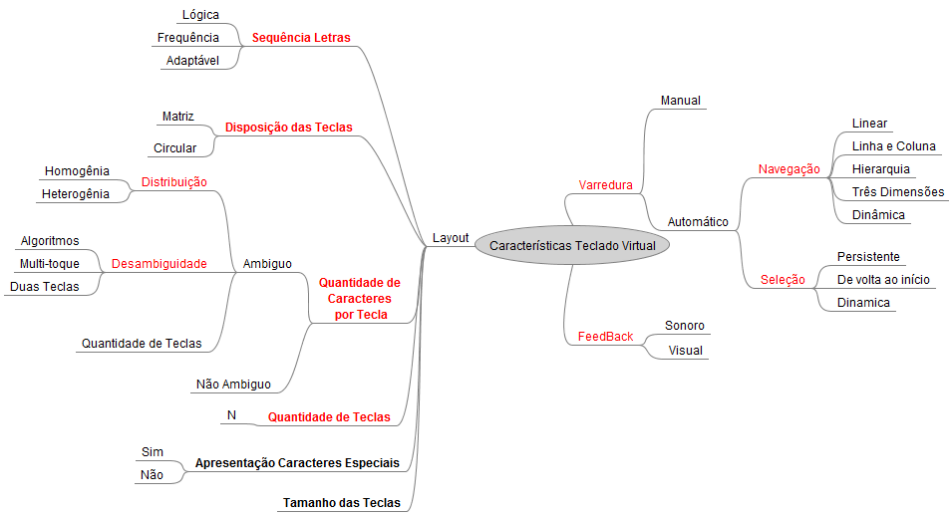


Figura 9 – Mapa mental das características do teclado virtual identificadas na revisão sistemática

Fonte: Elaborada pelo autor.

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, dispondo-as em linhas e colunas. Poucas pesquisas posicionam as teclas de forma circular (PRABHU; PRASAD, 2011; TOPAL; BENLIGIRAY; AKINLAR, 2012). As figuras 10 e 11 ilustram essas duas disposições de teclas, respectivamente.

Figura 10 – Teclado virtual em forma de matriz
Fonte: Elaborada pelo autor.

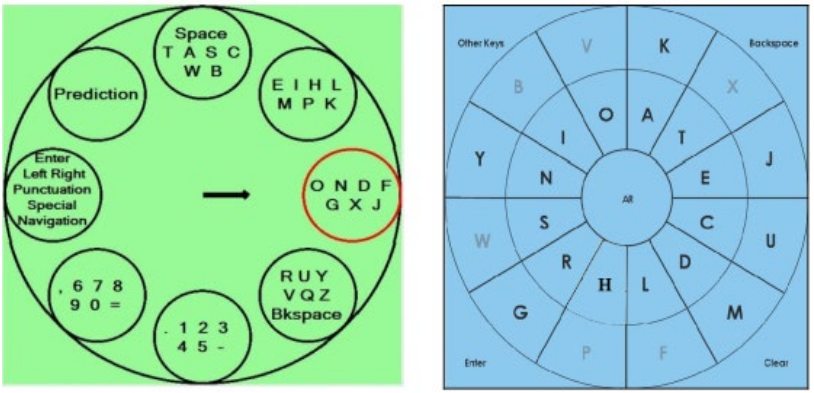


Figura 11 – Teclados virtuais em forma circular
Fonte: Prabhu e Prasad (2011) e Topal, Benligiray e Akinlar (2012).

A forma de distribuição matricial se assemelha mais à 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. A forma circular, por sua vez, é totalmente diferente da forma de apresentação das teclas dos teclados convencionais. Pesquisadores como Topal, Benligiray e Akinlar (2012) defendem que o posicionamento circular possibilita que as letras estejam mais próximas umas das outras, aumentando a performance de digitação.

O tamanho das teclas

Além da geometria de apresentação do teclado virtual, vários trabalhos consideram relevante o tamanho das teclas (AL FARAJ; MOJAHID; VIGOUROUX, 2009b; AL FARAJ; VIGOUROUX; MOJAHID, 2009; GELORMINI; BISHOP, 2013; GHOSH *et al.*, 2010; NICOLAU *et al.*, 2013; SARCAR *et al.*, 2010; VARCHOLIK; LAVIOLA; HUGHES, 2012; WU; HUANG; WU, 2014). O teclado virtual pode possuir diversas teclas de mesmo tamanho ou de tamanhos variados. Trabalhos como os de Al Faraj, Mojahid e Vigouroux (2009a, 2009b) 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, diminuindo a possibilidade de que o usuário pressione uma tecla indesejada. Essa estratégia também reduz a distância entre as teclas e o tempo de procura pela letra desejada, o que aumenta a performance de digitação.

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 e 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 determinado *corpus* do idioma adotado pelo teclado virtual (JOSHI *et al.*, 2011). O *corpus* de uma linguagem é a 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 à 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.

Quantidade de caracteres por tecla

A *quantidade de caracteres por tecla* é característica relevante na construção dos teclados virtuais. Teclados que possuem uma letra por tecla são classificados como teclados não ambíguos, e os teclados que possuem mais de um caractere por tecla são categorizados como teclados ambíguos (MOLINA; RIVERA; GÓMEZ, 2009). Analisando os artigos e documentos encontrados na pesquisa de que resultou este livro 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 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 o de Tanaka-Ishii, Inutsuka e Takeichi (2002) e o de Miró-Borrás e Bernabeu-Soler (2009), 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.

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, entre eles o sonoro, visual e pseudotátil.

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* pseudotátil 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.

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 (POLÁČEK; MÍKOVEC; SLAVÍK, 2012). Por esse motivo, essas duas técnicas são consideradas essenciais para a utilização do teclado virtual assistivo.

Protocolo de interação

O protocolo de interação permite desenvolver comandos específicos para determinados tipos 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 os padrões de piscada: se o usuário piscasse brevemente os olhos, o sistema poderia efetuar uma ação e realizaria outra, se o paciente fechasse os olhos durante maior tempo. 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 *enes* comandos. Com 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 organiza os comandos conforme uma sequência de piscadas (KNUTH, 1985), usando uma combinação binária de no máximo n elementos para obter a representação. Por exemplo, a representação alfabética do código de Morse, que usa um código de até cinco elementos para representar uma letra, um número ou um caractere especial.

Utilizando a codificação de Huffman é possível estruturar os comandos do sistema conforme uma sequência de piscadas. Uma sequência de piscadas curtas e longas pode 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 de 12 a 14 ilustram as possíveis opções de árvores conforme o tamanho do código a ser interpretado. Nessas figuras a letra *P* significa uma piscada curta, enquanto a letra *L* representa uma piscada longa. Pode-se observar que nelas a possibilidade de opções aumenta exponencialmente em relação ao tamanho do código.

Além disso, quanto maior é o número de elementos do código, mais tempo é necessário para acionar um comando.

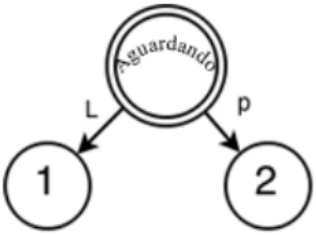


Figura 12 – Possíveis opções para as palavras de tamanho um com a codificação de Huffman
Fonte: Elaborada pelo autor.

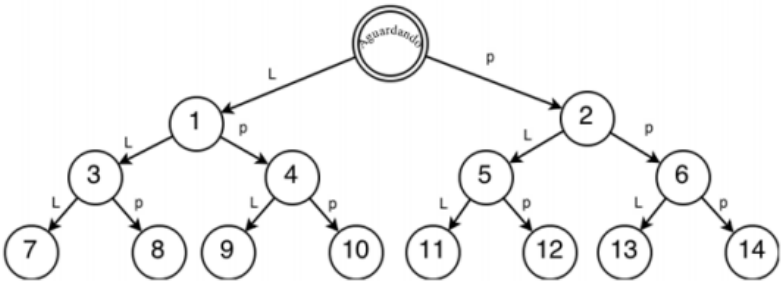


Figura 13 – Possíveis opções para as palavras de tamanho dois com a codificação de Huffman
Fonte: Elaborada pelo autor

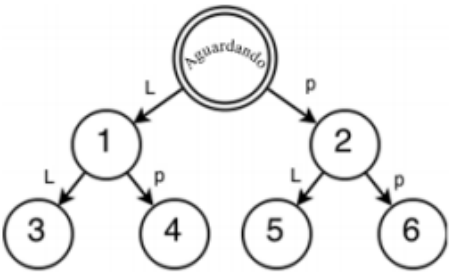


Figura 14 – Possíveis opções para as palavras de tamanho três com a codificação de Huffman
Fonte: Elaborada pelo autor.

Para acionar o comando de número oito ilustrado na Figura 14 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 lon-

ga 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. A Figura 15 mostra o tempo de execução dos comandos 4, 5 e 13 da Figura 14. O tempo total para a execução dessa sequência de comandos é de no mínimo 3.800 ms.

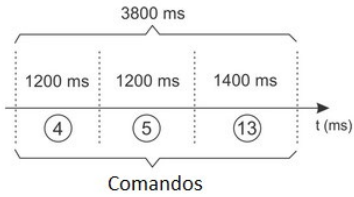


Figura 15 – Tempo de execução dos comandos 4, 5 e 13 da Figura 14
Fonte: Elaborada pelo autor.

Na sequência, são apresentados os métodos de varredura que podem ser utilizados como 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.

Métodos de varredura

Foram identificados quatro *métodos de varredura* (POLÁČEK; 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 determinado período de tempo. O usuário seleciona o elemento emitindo um sinal, assim que a tecla desejada ficar destacada das demais teclas (SCHADLE, 2004).

Na Figura 16, que ilustra a varredura linear, 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. Esse método passa pelas teclas sequencialmente até que seja selecionada uma delas.

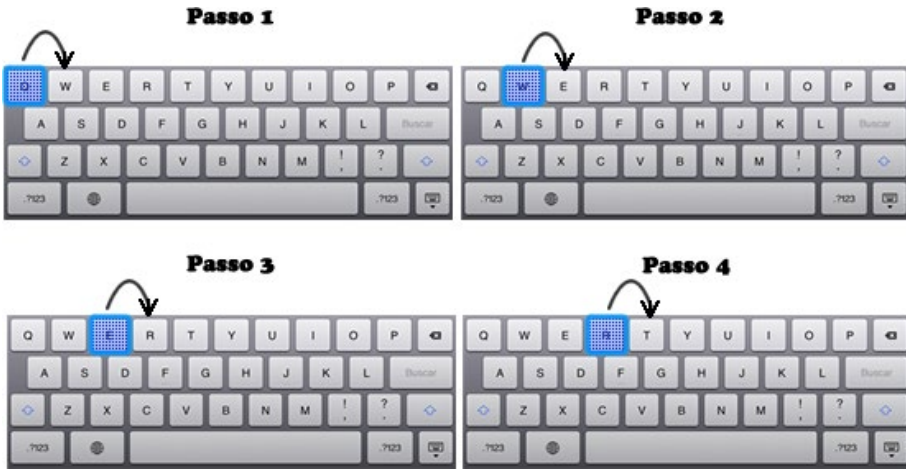


Figura 16 – Método de varredura linear em quatro passos

Fonte: Elaborada pelo autor.

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 17 mostra esse método.

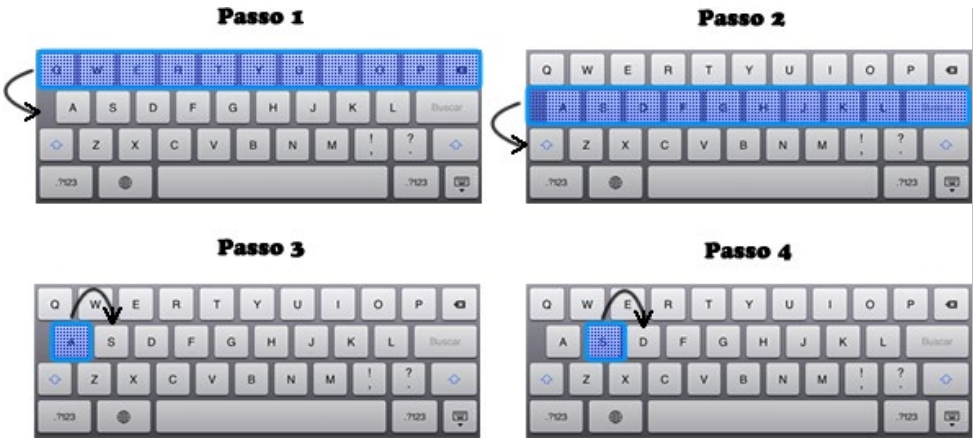


Figura 17 – Método de varredura linha e coluna em quatro passos

Fonte: Elaborada pelo autor.

Como é possível observar na figura, os grupos de linhas são destacados até que o usuário escolha a linha na qual a letra está. Após a primeira seleção, é executado o método de seleção linear. Nos passos um e dois o foco do cursor está se movendo entre as linhas do teclado. No passo dois o usuário emite 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 seções distintas (FELZER; RINDERKNECHT, 2013). 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 18 mostra a varredura em três dimensões, na qual são destacados blocos de caracteres. Assim que o usuário seleciona um bloco, o sistema executa o método de varredura linha e coluna.

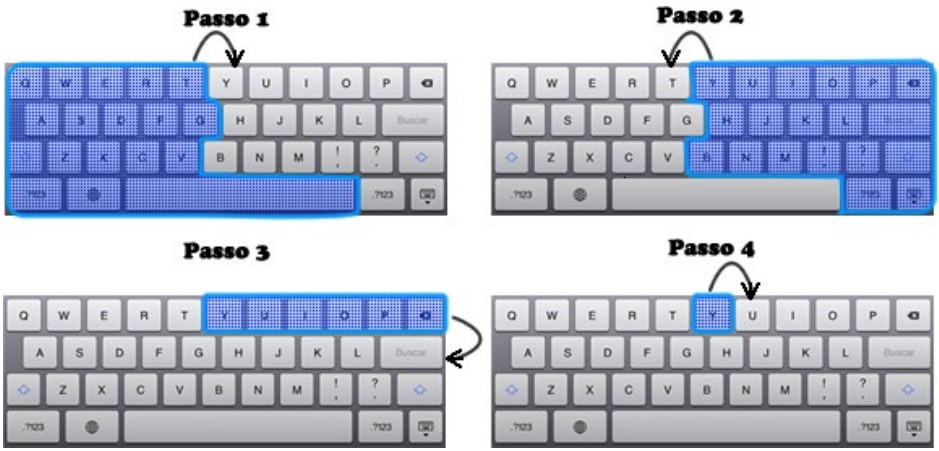


Figura 18 – Método de varredura três dimensões em quatro passos
Fonte: Elaborada pelo autor.

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 das 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. A Figura 19 ilustra o grafo de navegação balanceado utilizando o unigram, ao passo que a Figura 20 representa a varredura em hierarquia com esse grafo, de modo que o método é iniciado destacando o grupo de caracteres. Assim que um grupo é selecionado, o sistema inicia o método de varredura linear. .

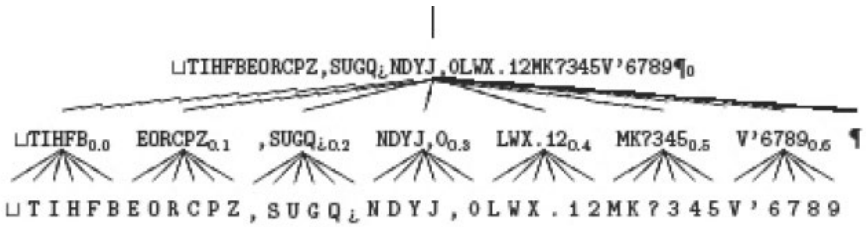


Figura 19 – Grafo do método de varredura *containment hierarchy* em forma de árvore
Fonte: Baljko e Tam (2006).

Figura 20 – Método de varredura *containment hierarchy* em quatro passos
Fonte: Elaborada pelo autor.

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, os autores ajustaram os caracteres em ordem alfabética e posicionaram as teclas em uma matriz de 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 21 mostra um exemplo desse método de varredura. É realizada uma varredura binária entre os caracteres. Assim, o sistema destaca o bloco desejado e o usuário seleciona o bloco no qual está o caractere até que o sistema destaque o caractere desejado.

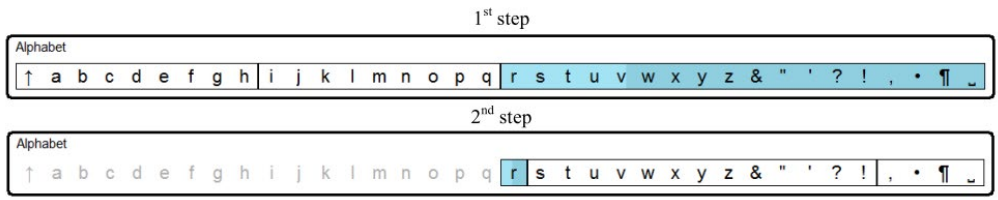


Figura 21 – Método de varredura proposto por Poláček, Míkovec e Slavík (2012)
Fonte: Poláček, Míkovec e Slavík (2012).

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.

Outra característica variável do método de varredura é a ação após a seleção de 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 a 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 à performance de digitação dos teclados que utilizam o método de varredura (MIRÓ-BORRÁS; BERNABEU-SOLER, 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).

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.

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.

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 do termo e/ou 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, 2006).

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, 2006).

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 22 ilustra os tipos de predição encontrados na literatura, que se dividem em dois grupos: predição estatística e predição sintática – grupos que podem ser combinados.

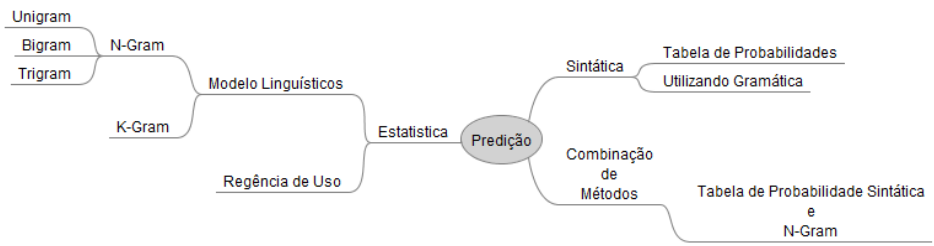


Figura 22 – Tipos de predição identificados durante a revisão sistemática
Fonte: Elaborada pelo autor.

Predição de letras

Os *algoritmos de predição de letras* auxiliam o usuário sugerindo os caracteres mais prováveis de ocorrerem após a escolha de 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.

Modelo linguístico n-gram

A probabilidade de determinado termo 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 determinada sequência de termos baseando-se nos 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, 1998b). 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 1 mostra o cálculo realizado no n-gram.

$$P(c_i) = P(c_i | c_{i-n}, \ldots, c_{i-1}) \qquad \text{Equação 1}$$

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 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 determinada sequência de termos.

Tabela 1 – Probabilidades do método n-gram

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

Fonte: Elaborada pelo autor.

Considerando a Tabela 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.

Modelo linguístico k-gram

Outro modelo linguístico utilizado para predição de texto é o k-gram. Segundo Miró-Borrás e Bernabeu-Soler (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 à outra palavra.

Além da previsão do caractere, o k-gram pode ser usado para a previsão das palavras (LESHER; MOULTON; HIGGINBOTHAM, 1998b). 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, 1998b).

Predição de palavras

A predição de palavras pode ser utilizada para completar palavras (TRUONG *et al.*, 2013) ou para sugerir uma palavra durante a escrita/digitação. O método de predição é o sistema que busca antecipar o próximo bloco de caracteres, letras, sílabas, palavras, sentenças, que o usuário deseja expressar (GARAY-VITORIA; ABASCAL, 2006).

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 aos *softwares* de comunicação com o objetivo de aumentar a performance de digitação (GARAY-VITORIA; ABASCAL, 2006). Nas pesquisas de que provém este livro 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.

Frequência de ocorrência de uma única palavra

O método de freqüência de determinada palavra calcula o número de vezes que essa aparece no *corpus* da linguagem (GARAY-VITORIA; ABASCAL, 2006). 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 23 ilustra o processo de sugestão de palavras utilizando a frequência única.

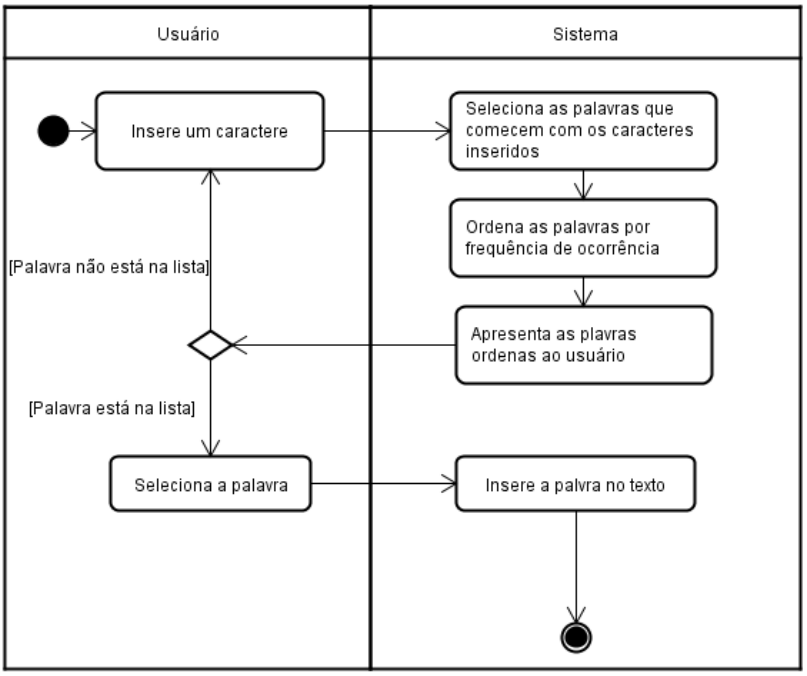


Figura 23 – Processo de predição de uma única palavra
Fonte: Elaborada pelo autor.

Assim que o usuário inicia o processo de digitação, o sistema seleciona quais palavras que começam com os caracteres inseridos e 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.

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, 2006). Portanto, o método de regência de uso especifica o quão recentemente uma palavra foi usada. Se determinado termo foi utilizado recentemente, essa palavra tem prioridade sobre as outras (SHARMA *et al.*, 2012). 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 alguma palavra completa ou seleciona uma palavra a partir da lista de sugestões, a variável relativa à regência de uso da palavra inserida é acrescida de um. Dependendo da posição do elemento no texto, essa variável é ponderada por um fator de deterioração (CLARKSON; ROBINSON, 1997).

O conceito do fator de deterioração determina que a probabilidade de recorrência de determinada 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 quinze a vinte palavras depois do termo escrito. Após esse número de palavras a probabilidade de reutilização tende a cair.

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, 2006). Para n maior do que um, esse método pode ser calculado como um n-gram. Assim que o usuário insere alguma 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, 2006). Os bigrams consideram a palavra

anteriormente escrita para predizer a próxima palavra. A Equação 2 calcula a probabilidade do bigram.

$$P(w_i) = P(w_i|w_{i-1})$$ Equação 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 mostra o cálculo do trigram.

$$P(w_i) = P(w_i|w_{i-1}, w_{i-2})$$ Equação 3

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

O trigram é dos métodos de predição estatística mais utilizados (SHARMA *et al.*, 2010). Wandmacher *et al.* (2008) afirmaram que, quanto maior o número de termos considerados anteriormente à palavra que está sendo escrita, maior é a exatidão do método de predição. Porém, Garay-vitoria e Abascal (2006) e Sharma *et al.* (2010) ressaltaram que o aumento do número de palavras pode ocasionar em acréscimo considerável no uso de processamento e memória computacional.

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. Sendo o primeiro elemento a probabilidade de ocorrência de cada palavra, e o segundo termo da tabela a probabilidade relativa de ocorrência de todas as categorias sintáticas após cada categoria sintática (GARAY-VITORIA; ABASCAL, 2006). Quando o usuário insere determinada 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 4 calcula a probabilidade de prever a próxima palavra baseada nas duas últimas categorias sintáticas.

$$P(w_i|t_{i-1}, t_{i-2})$$ Equação 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 5 apresenta o cálculo dessa probabilidade.

$$P(w_i) = P\left(w_i|w_{i-1,}t_{i-1,}t_{i-2,}\right)$$ Equação 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 à 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 ultilizou o modelo de linguagem bigramo, usando

essas duas predições balanceadas por meio de uma variável que define o peso de cada probabilidade. A Equação 6 mostra esse cálculo.

$$P(w_i) = \alpha \times P(w_i|w_{i-1}) + (1-\alpha) \times [P(w_i|t_{cw}) \times P(t_{cw}|t_{pw}, t_{ppw})] \quad \text{Equação 6}$$

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

Predição sintática utilizando gramática

De acordo com Garay e Abascal (2006), 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 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, 2006).

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 às palavras. Algumas características desse componente são: organização, tamanho, número de dicionários e adaptação do dicionário.

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 (GARAY-VITORIA; ABASCAL, 2006; SHARMA *et al.*, 2010).

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, 2006). 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 reestruturações inteiras no dicionário (GARAY-VITORIA; ABASCAL, 2006).

Tamanho do dicionário

O tamanho do dicionário é outra característica que deve ser considerada ao construir esse componente, pois ela 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, 2006). 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 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, 2006).

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, 2006).

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.

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, à medida que esse 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 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, 2006).

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; SARCAR; SAMANTA, 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.

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 da 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 7 representa a Lei de Fitts.

$$TM = a + b\,log_2(\frac{D_{ij}}{L_j + 1}) \qquad \text{Equação 7}$$

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 24 ilustra as variáveis D_{ij} e L_j.

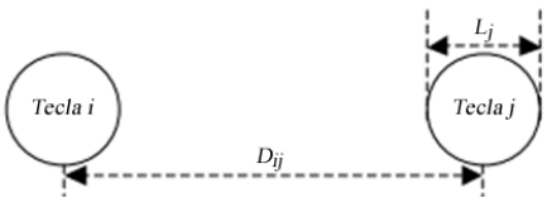


Figura 24 – Ilustração da Lei de Fitts

Fonte: Elaborada pelo autor.
Nota: 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 Al faraj, Mojahid e Vigouroux (2009b) usa a Lei de Fitts para otimizar o teclado de dispositivos móveis. A Figura 25 mostra o teclado proposto por eles.

Figura 25 – Teclado proposto por Al Faraj, Mojahid e Vigouroux (2009b)
Fonte: Al Faraj, Mojahid e Vigouroux (2009b).
Nota: As teclas são aumentadas de tamanho para facilitar sua seleção.

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 (AL FARAJ; MOJAHID; VIGOUROUX, 2009b).

Al Faraj, Mojahid e Vigouroux (2009a) propuseram outro *layout* de teclado virtual adicionando uma terceira dimensão a ele. Novamente apresentaram, utilizando a Lei de Fitts, um *layout* que expande e contrai. Dois fatores realizam a expansão e a contração desse *layout*, sendo eles: o movimento de seleção do usuário e a predição dos próximos caracteres mais prováveis. A Figura 26 ilustra o *layout* proposto em seus estudos.

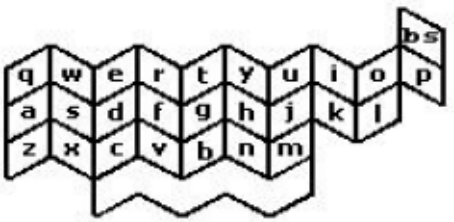


Figura 26 – Teclado proposto por Al Faraj, Mojahid e Vigouroux (2009a)
Fonte: Al Faraj, Mojahid e Vigouroux (2009a).

Nota: O teclado possui a forma de uma sanfona. Assim que o usuário inicia a seleção, as letras e as teclas se expandem.

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

Frequência de um único caractere

O método de frequência de um único caractere calcula a probabilidade de ocorrência de caractere no *corpus* de determinada 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 de ela ser a primeira letra da palavra para distribuir as letras entre as teclas.

Topal, Benligiray e Akinlar (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 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 27 foi construído para minimizar o movimento dos olhos.

Figura 27 – Teclado proposto por Panwar, Sarcar e Samanta (2012)

Fonte: Panwar, Sarcar e Samanta (2012).

Nota: Esse teclado é similar ao teclado de Fitally, no qual as letras mais utilizadas ficam mais próximas da tecla de espaço enquanto as demais teclas ficam na parte mais exterior do teclado.

Frequência de dígrafos

O método de otimização de *layouts* que utiliza dígrafos calcula a probabilidade de ocorrência de determinado dígrafo a partir do *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 e 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 28 mostra um exemplo de teclado virtual construído pela ferramenta.

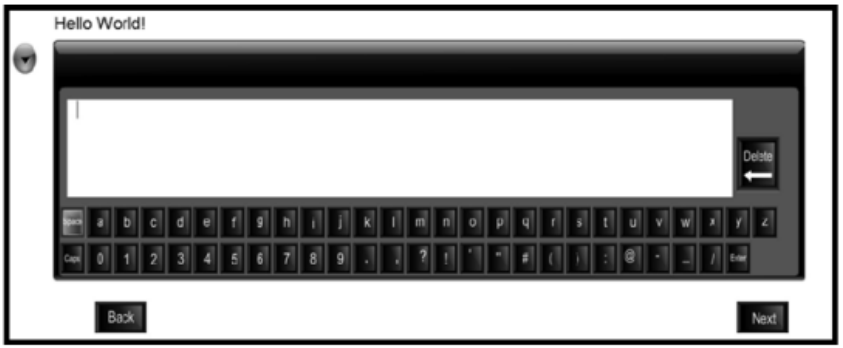


Figura 28 – *Software* proposto por Millet, Asfour e Lewis (2009) para construção de teclados virtuais

Fonte: Millet, Asfour e Lewis (2009).

Nota: O objetivo dos autores foi realizar testes de performance e usabilidade.

Fitts' Digraph

O modelo *Fitts' 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 é convertida 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 construir esse *layout*. Após quatro horas de treino um usuário que usa o teclado OPTI atinge 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' Digraph*, bigram e trigram. Esse novo *layout* é dividido em três partes distintas: caracteres, flexões e pontuações.

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 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 29.

Figura 29 – Teclado proposto por Sarcar *et al.* (2010) para a linguagem bengali
Fonte: Sarcar *et al.* (2010)).

Algoritmos de otimização

Ainda não existe uma solução definitiva que resolva o problema da baixa performance de escrita em teclados virtuais utilizados por pessoas com deficiência (POLÁČEK; MÍKOVEC; SLAVÍK, 2012). Portanto, para otimizar os *layouts* dos teclados virtuais alguns pesquisadores utilizam algoritmos de otimização. A finalidade desses algoritmos é identificar alguma solução ótima para o problema, minimizando a 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. Além disso, é preciso determinar qual o método será usado para otimizar o arranjo das letras.

Como o processo de otimização de *layouts* utilizando algoritmos é complexo, todo o Capítulo 4 é dedicado ao detalhamento desse problema.

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 multitoque, duas teclas e algoritmos de desambiguidade.

Multitoque

No método multitoque 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 o 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 30 ilustra o método de desambiguidade multitoque. Nessa figura o usuário digita a palavra "cd". O círculo em cima da tecla significa que ela foi pressionada.

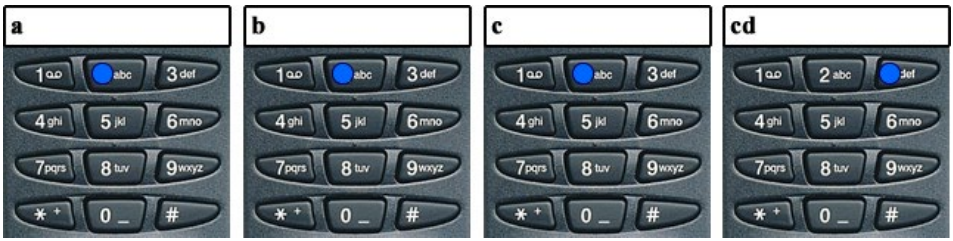


Figura 30 – Desambiguidade utilizando o método multitoque

Fonte: Elaborada pelo autor.

Nota: O processo apresentado pela figura mostra a escrita da palavra "cd".

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 deseja selecionar a letra "e", considerando o teclado mostrado na Figura 30, 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 duas 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).

Algoritmos de desambiguidade

Os algoritmos de desambiguidade são mais complexos. Essa abordagem utiliza um dicionário de termos para encontrar as palavras candidatas que correspondem à 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 nove teclas que possua a sequência de letras em ordem alfabética, como mostrado na Figura 31. 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 31 – Teclado ambíguo com nove teclas
Fonte: Elaborada pelo autor.
Nota: As letras de "a" a "z" são distribuídas entre as nove teclas seguindo a ordem alfabética.

LetterWise

Mackenzie *et al.* (2001) apresentaram um novo método denominado *LetterWise* para realizar o processo de desambiguidade. Esse método utiliza uma tabela de prefixos para executar esse processo, sendo o prefixo 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 alguma 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 de a 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.

Algoritmo T9

O método T9 (texto em nove teclas) de desambiguidade (GROVER; KING; KUSHLER, 1998) é usado para resolver a ambiguidade das palavras nos telefones com nove teclas. Esse método de desambiguidade possui melhor performance de digitação se comparado ao método multitoque (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 à 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 à 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, na próxima vez que o usuário entrar com essa palavra o algoritmo a apresentará na lista de sugestões.

Em 2007, a empresa Nuance Communications incorporou-se à Tegic Communications e adquiriu os direitos sobre o T9 (NUANCE, 2015a). Logo, foram adicionadas novas funcionalidades a ele, como a predição de texto, a correção de erros por região, a correção da pronúncia, completar as palavras, os termos de atalho e a adição de pontuação automática (NUANCE, 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.

A correção de erros por região visa 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 à 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.

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 nove 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 *et al.* (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.

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 à tecnologia (MCCOY *et al.*, 2013). Consequentemente, esses usuários podem parar de utilizar o 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 de o 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.

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. Essa personalizaçãoé efetuada explicitamente, ou seja, o usuário configura o *software* da maneira que 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 do 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 permaneçam 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 fator essencial para o usuário desse teclado (GARAY-VITORIA; ABASCAL, 2006). 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.

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 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 de maneira rápida provavelmente preferem um tempo de espera menor do que os usuários que demoram para pressioná-las.

Bhattacharya e Laha (2012) propuseram 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 à grande quantidade de símbolos, esses pesquisadores dividiram o *layout* em dois conjuntos de símbolos em alternância, com o objetivo de apresentar os caracteres mais prováveis de serem digitados durante a maior parte do tempo. Essa alternância entre os conjuntos de símbolos é realizada considerando o método de paginação atualmente menos utilizado, com uso para gerenciar a memória do computador.

Métricas para avaliação da 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 deles é mais lento do que o anterior provavelmente não será mais 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 relação direta. Assim, não é vantagem um método ter 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).

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 apresentaram a performance de digitação utilizando essa 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 8 mostra o cálculo da WPM.

$$WPM = \frac{|T| - 1}{S} \times 60 \times \frac{1}{5}$$ Equação 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 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 um caractere, pois a contagem do tempo começa a partir da entrada do primeiro caractere. Um exemplo de medição utilizando o WPM é descrito a seguir.

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 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 cinquenta segundos, o valor WPM do método de entrada será igual a 10,8 palavras por minuto. A Figura 32 mostra a utilização da Equação 8.

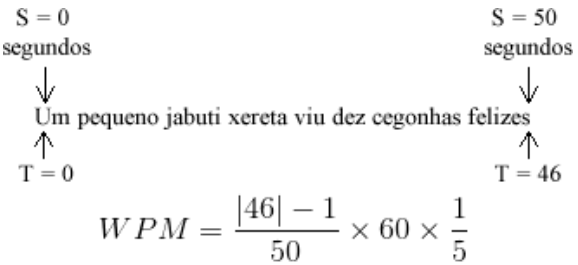


Figura 32 – Exemplo de utilização da medição WPM

Fonte: Elaborada pelo autor.

Caracteres por minuto

A medida de caracteres por minuto afere o número de caracteres digitados por minuto. Wobbrock (2007) e Bhattacharya e Laha (2012) 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 9, T e S são os mesmos números utilizados na Equação 8.

$$CPM = \frac{|T| - 1}{S} \times 60$$ Equação 9

As métricas WPM e CPM não consideram os erros inseridos pelos usuários durante a entrada de texto (WOBBROCK, 2007). Bhattacharya, Basu e Samanta (2008) mostraram a importância de considerar esses erros para aferir a performance de digitação de um teclado virtual. Isso porque muitos softwares de entrada de dados podem possuir boa performance de comunicação, porém esses mesmos sistemas podem apresentar 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.

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).

$$KSPC = \frac{|IS| - 1}{T}$$ Equação 10

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

Na Equação 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 33 mostra a utilização da métrica KSPC. Nessa figura, o usuário digita o pangrama em cinquenta 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.

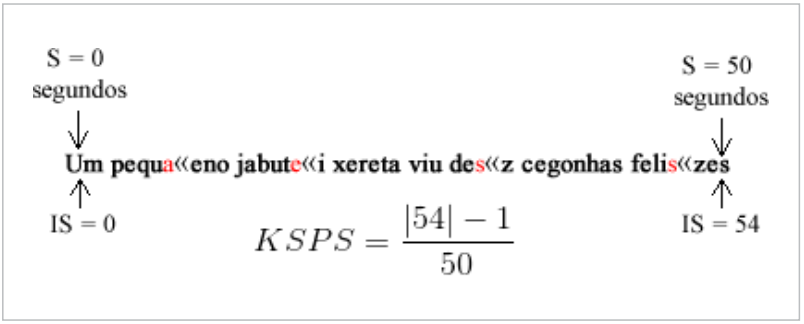


Figura 33 – Exemplo da utilização da métrica KSPS
Fonte: Elaborada pelo autor.

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 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 gesto para o sistema. Por exemplo, 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 11 mostra o cálculo da GPC.

$$GPC = \frac{|IS\emptyset| - 1}{T} \qquad \text{Equação 11}$$

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

Taxa de erros

A taxa de erros do usuário é aferida como mostrado na Equação 12 por Bhattacharya e Laha (2012).

$$Total\ Error\ Rate = \frac{INF + IF}{C + INF + IF} \times 100$$ Equação 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.

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 e Mackenzie (2001).

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 34 ilustra esse teclado.

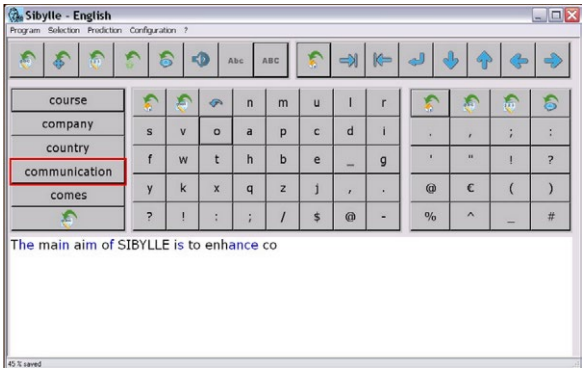


Figura 34 – Teclado Sibylle proposto por Wandmacher *et al.* (2008)
Fonte: Wandmacher *et al.* (2008).
Nota: Esse teclado utiliza diversas técnicas de predição e contempla caracteres especiais.

O FITALY (Figura 35), um teclado virtual construído para aumentar a performance de digitação utilizando *stylus* (MACKENZIE; ZHANG; SOUKOREFF, 2010), possui duas barras de espaço. Seu *layout* posiciona as letras mais comuns perto desses teclas, tendo sido desenvolvido com o *corpus* e os dígrafos da língua inglesa. Trabalhos como os de Panwar, Sarcar e Samanta (2012) utilizaram uma variação desse teclado em conjunto com a captura do movimento dos olhos para auxiliar a digitação.

Z	V	C	H	W	K
F	I	T	A	L	Y
Espaço		N	E	Espaço	
G	D	O	R	S	B
Q	J	U	M	P	X

Figura 35 – Teclado FITALY
Fonte: Jain e Bhattacharya (2010).
Nota: Este teclado possui duas barras de espaço e posiciona as letras mais utilizadas da língua inglesa perto dessas teclas.

O teclado virtual OPTI (Figura 36), que apresenta uma proposta de similar à do FITALY (GUERRIER *et al.*, 2011), utiliza a Lei de Fitts ponderada pela frequência de dígrafos (*Fitts' Digraph*) para posicionar as teclas e as letras (MACKENZIE; ZHANG, 1999). O objetivo desse teclado é aumentar a velocidade de digitação e minimizar a quantidade de erros de entrada de dados.

Figura 36 – Teclado OPTI

Fonte: Mackenzie e Zhang (1999).
Nota: Este teclado ossui quatro barras de espaço e utiliza a Lei de Fitts, bem como a frequência entre os dígrafos, para posicionar as letras entre as teclas.

O Dvorak (Figura 37) é um teclado virtual que reorganiza as letras do *layout* QWERTY (MACKENZIE; ZHANG; SOUKOREFF, 2010). Esse *layout* foi desenvolvido a fim de aumentar a performance de digitação dos teclados que podem ser operados com as duas mãos. Esse teclado distribui as letras de acordo com a frequência dos dígrafos, os quais são posicionados de maneira que suas letras fiquem alternadas entre a mão esquerda e a direita. Seu *layout* possibilita reduzir a movimentação dos dedos e, assim, o esforço durante a entrada de texto (GUERRIER *et al.*, 2011).

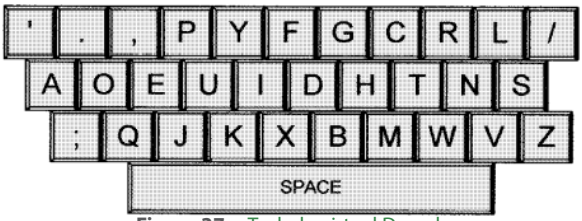


Figura 37 – Teclado virtual Dvorak

Fonte: Mackenzie, Zhang e Soukoreff (2010).
Nota: Este teclado osiciona as letras entre as teclas fundamentado na frequência de ocorrência dos dígrafos da língua inglesa.

O teclado virtual K-Hermes (Figura 38), um teclado ambíguo que utiliza o método de multitoque para realizar a desambiguidade das teclas (GUERRIER *et al.*, 2011), apresenta 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.

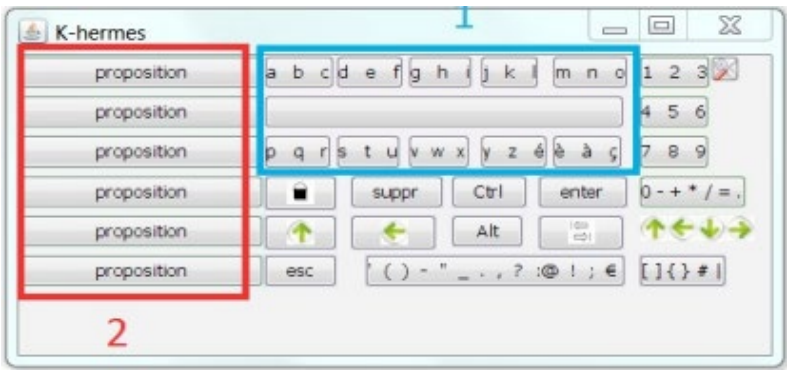


Figura 38 – Teclado virtual K-Hermes
Fonte: Guerrier *et al.* (2011).

Nota: Trata-se de um teclado ambíguo que utiliza letras, caracteres especiais e números, possui sistema de predição e dispõe as letras entre as teclas em ordem alfabética. A seleção 1 mostra a disposição das teclas e a seleção 2 apresenta a lista de palavras sugeridas.

O Dasher (Figura 39) é uma interface de entrada de texto orientada por gestos de apontamentos contínuos e naturais (WARD; BLACKWELL; MACKAY, 2000).

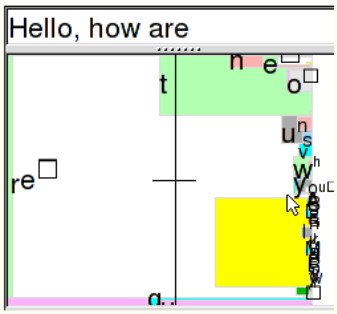


Figura 39 – Teclado virtual Dasher
Fonte: Ward, Blackwell e Mackay (2000).

Nota: Este teclado é uma forma de comunicação diferenciada. As letras passam no aplicativo de acordo com sua probabilidade de ocorrência.

A abordagem desse teclado 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, o que pode ser empregado quando não é possível utilizar teclados virtuais de tamanho convencional (WARD; BLACKWELL; MACKAY, 2000).

O Chewing Word (Figura 40) 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 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.

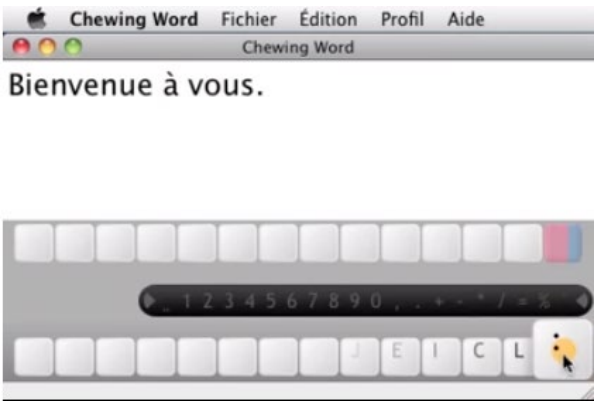


Figura 40 – Teclado virtual Chewing Word

Fonte: Grange (2010).

Nota: Este teclado foi criado a princípio para a língua francesa, mas atualmente ele suporta diversas línguas. Ele possui um sistema de predição de letras que auxilia o usuário a selecionar as letras mais prováveis de ocorrerem.

O UKO-II (Figura 41), um teclado virtual ambíguo com apenas quatro teclas (HARBUSCH; KÜHN, 2003), utiliza um algoritmo de desambiguidade e permite que o usuário configure a sequência dos símbolos entre as teclas. Seu principal objetivo é facilitar a entrada de textos dos usuários com paralisia cerebral.

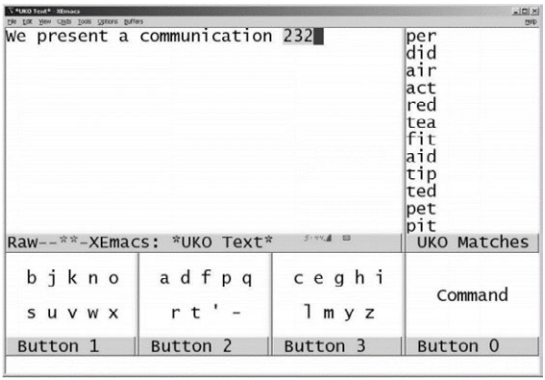


Figura 41 – Teclado virtual UKO-II

Fonte: Harbusch e Kühn (2003).

Nota: Este teclado é ambíguo e possui um sistema de predição de palavras além de um algoritmo de desambiguidade.

O K-Thot (Figura 42), um teclado virtual desenvolvido para pessoas com deficiência motora, tenta reduzir o número de movimentos necessários para a entrada de texto, aproximando as letras que possuem as maiores frequências de ocorrência em francês.

Figura 42 – Teclado virtual K-Thot

Fonte: Baas *et al.* (2010).

Nota: Este teclado tenta diminuir o número de movimentos necessários para digitar uma palavra aproximando as letras com maior frequência de ocorrência.

O KeyGlass (Figura 43), um teclado que obedece o *layout* francês de teclas e letras, tem o objetivo de diminuir o tempo e o esforço de digitação. Para atingir esse objetivo, esse teclado apresenta as letras mais prováveis de ocorrerem ao redor da última letra selecionada.

Figura 43 – Teclado virtual KeyGlass

Fonte: Colas *et al.* (2008).

Nota: Este teclado apresenta as letras mais prováveis de ocorrerem perto da letra que foi selecionada.

O teclado virtual Metrópolis (Figura 44) 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 (2000) é aumentar a performance de digitação em teclados para dispositivos móveis.

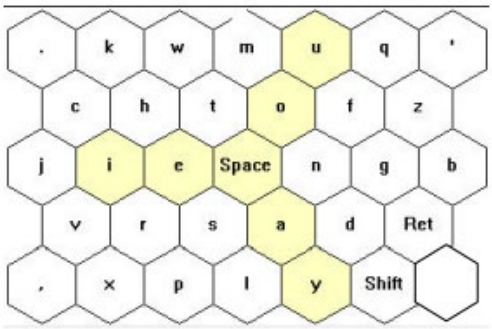


Figura 44 – Teclado virtual Metrópolis

Fonte: Zhai, Hunter e Smith (2000).

Nota: Zhai, Hunter e Smith (2000) utilizaram o algoritmo de otimização Metrópolis para distribuir as letras entre as teclas do teclado virtual.

O teclado virtual HandiGyph (Figura 45), proposto por Belatar e Poirier (2008), 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.

Figura 45 – Teclado virtual HandiGyph

Fonte: Belatar e Poirier (2008).

Nota: Proposto por Belatar e Poirier (2008), trata-se de um teclado ambíguo com quatro teclas que utiliza um algoritmo de desambiguidade.

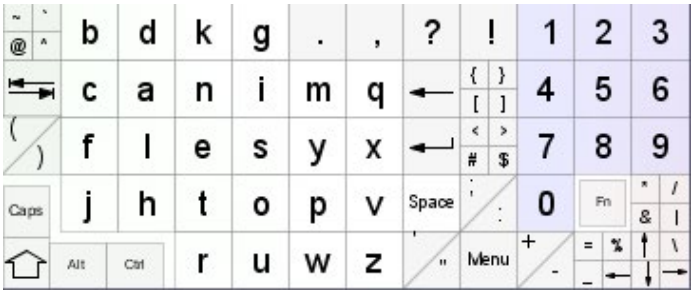


Figura 46 – Teclado vitural Atomik

Fonte: Zhai e Kristensson (2007).

Nota: Este teclado utiliza o algoritmo de otimização Metrópolis para distribuir as letras entre as teclas. Ele disponibiliza números e caracteres especiais e de navegação.

O teclado virtual Atomik (Figura 46) é um acrônimo 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.

Com a finalidade de diminuir a movimentação dos dedos durante a digitação, o teclado virtual Xpert (Figura 47) aproxima as letras que compõem os bigrams da língua inglesa.

Figura 47 – Teclado virtual Xpert
Fonte: XpertKeyboard (2015).
Nota: Este teclado distribui as letras entre as teclas utilizando os bigrams da língua inglesa.

3

Otimização de layouts

A distribuição de letras entre as teclas de um teclado influencia 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, 1993). Portanto, a solução do arranjo de teclas não pode ser realizada computacionalmente em tempo polinomial.

Apesar de não possuir solução determinística que possa ser encontrada em 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 meta-heurística dos algoritmos genéticos utilizada na pesquisa apresentada neste livro.

Problema de otimização do teclado

O problema de otimização do arranjo do teclado resume em distribuir uma quantidade de caracteres em um conjunto de 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 à desambiguidade e à predição de texto (YIN; SU, 2011).

Os critérios ergonômicos estão associados à redução do esforço de digitação relacionado à 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 13 (LESHER; MOULTON; HIGGINBOTHAM, 1998b).

$$A_{total} = \frac{M!}{\prod_{i=1}^{N=1}(k_i!) \times \prod_{n=1}^{n=k_{max}}(c_n!)}$$ Equação13

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, 1998b). Ele analisou um *layout* que tinha oito teclas cada uma com três caracteres e uma tecla com dois. Esse arranjo permitiu gerar 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 próxima seção.

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 de as funções objetivo e das configurações de solução serem estáticas para cada problema, o método de solução pode variar.

A meta-heurística é um conjunto de métodos que coordenam os procedimentos de busca com estratégias de 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 (GLOVER; KOCHENBERGER, 2003). Portanto, a meta-heurística pode obter boas soluções para problemas que não possuem um algoritmo de complexidade polinomial. Algumas meta-heurísticas utilizadas para resolver o problema de otimização do teclado são: o recozimento simulado (KIRKPATRICK; VECCHI; GELATT, 1983), os algoritmos genéticos (HOLLAND, 1975), a nuvem de partículas (KENNEDY, 2011), 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 meta-heurística foi proposta por Kirkpatrick e colaboradores.

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 . 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 Holland (1975). 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 para o problema que está sendo tratado. Para cada indivíduo pertencente a 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 e Kennedy em 1995 (KENNEDY, 2011). Esse algoritmo é inspirado no comportamento e na dinâmica social dos pássaros, que durante o voo 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 alguma 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 à 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, em que 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 a 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 à otimização do arranjo do teclado encontrados 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 (1958). O método original é chamado de 2-opt e foi aplicado primeiramente para resolver o problema clássico do caixeiro viajante. Esse algoritmo realiza 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 a uma certa probabilidade. Os autômatos de aprendizado são os que aprendem quais são as melhores ações para minimizar as penalidades (OOMMEN; VALIVETI; ZGIERSKI, 1990).

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 (GOETTL; BRUGH; JULSTROM, 2005; LIGHT; ANDERSON, 1993; SÖRENSEN, 2007). 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 (GOETTL; BRUGH; JULSTROM, 2005; YIN; SU, 2011). 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 (EGGERS *et al.*, 2003; LEVINE; TREPAGNIER, 1990; LIGHT; ANDERSON, 1993). 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. Esses autores empregaram 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 convencionais, o índice de eficiência na desambiguidade das letras foi de 86% usando dez teclas e 62% ao utilizar quatro teclas.

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. Para isso, utilizaram a técnica de aprendizagem estocástica por autômatos. O autômato proposto por Oommen e colaboradores distribuíram 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 e 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 produzido 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 (1998b) apresentaram um trabalho de otimização utilizando o método *n-optimization*. Para calcular a eficiência dos teclados construídos usando esse método, 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 foi sugerido antes do caractere quando 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 obtiveram resultados melhores do que o seu correlato (LEVINE; TREPAGNIER, 1990).

Zhai e Smith (2001) avançaram com as pesquisas de teclados não ambíguo utilizando o algoritmo Metropolis para construir um novo *layout*. O objetivo 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 meta-heurística 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 desse 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 (GOETTL; BRUGH; JULSTROM, 2005; RAYNAL; VIGOUROUX, 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 estes autores foi codificado em função objetivo, essas funções são

minimizadas utilizando um algoritmo genético. Goettl, Brugh e Julstrom (2005) conseguiram construir teclados 40% a 20% mais eficientes do que os teclados QWERTY e Dvorak, respectivamente.

Sorensen (2007) desenvolveu seu trabalho com o objetivo de aprimorar a digitação das mensagens pelo celular. Para otimizar o teclado ambíguo de nove teclas foi utilizado um método de busca randômica. O autor 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 que 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 e Johnson (2011) 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 mantêm a sequência de grupos como a de números, caracteres de pontuação e letras. O trabalho desses autores 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-se dos outros, pois considerou quatro funções objetivo para a otimização. As propriedades otimizadas por Yin e Su (2011) foram: acessibilidade da tecla, conforto da postura de digitação, número de teclas pressionadas e colisões entre as palavras. Assim, conseguiram bons resultados e compararam o seu trabalho com outras abordagens como teclado alfabético, a abordagem de Levine e a distribuição de frequência.

Finalmente, Brouillette, Sharma e Kalita (2012) aumentaram a performance de digitação dos teclados não ambíguos utilizando algoritmos genéticos.

O trabalho deles também considerou o tempo de aprendizado para os novos usuários, com um teclado que possuía o dobro da performance de digitação do teclado QWERTY.

Análise

Os trabalhos de Yin e Su (2011), Lesher, Moulton e Higginbotham (1998b) e Levine e Trepagnier (1990) otimizaram os esforços necessários para digitar uma palavra. Porém, esses trabalhos consideraram apenas o acesso direto a tecla, sendo que 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; BERNABEU-SOLER, 2009). Apenas o trabalho de Francis e Johnson (2011) considerou o acesso indireto as teclas.

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

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, sendo a primeira o fato de que a maneira de escrita e o vocabulário do usuário não são considerados. A segunda é que os apanhados de textos nem sempre são equivalentes entre as pesquisas, e 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 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 (1998a). 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 à 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, cultural, sócioeconômico 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 há pouco tempo atrás podem tornar-se rapidamente obsoletas.

Fundamentado nas pesquisas de Francis e Oxtoby (2006) e de 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, que possibilita aumentar a sua capacidade de entrada de dados e principalmente diminuir o esforço de digitação.

Algoritmos genéticos

Como citado na seção "Soluções para problemas de otimização", os algoritmos genéticos simulam a teoria da evolução das espécies de Darwin. Essa teoria mostra a seleção natural, onde 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 definição rigorosa aceita por toda a comunidade de computação evolutiva 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 (MITCHELL, 1999). Decompondo esses elementos, pode-se reorganizá-los em componentes e processo.

As subseções "Componentes" e "Processo evolutivo" apresentam respectivamente, o detalhamento dos componentes e do processo padrão dos AGs.

Componentes

Nesta seção são apresentados os principais componentes presentes nos AGs.

Indivíduo

Os indivíduos representam as possíveis soluções para o problema a ser resolvido. Cada indivíduo, também chamado de cromossomo, é composto por genes, termo que foi 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, sendo o valor possível de um gene denominado como alelo. O alelo ocupa uma determinada posição em um cromossomo, que é definida como *lócus* (SADLER, 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, 1975).

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, sendo 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).

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 "Processo evolutivo".

O dimensionamento da população é uma característica importante para o AG. Algumas questões relevantes em relação a esse dimensionamento são (DE 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 a 1.000 indivíduos (DE JONG, 2006).

Algumas características importantes da população são:

- *Geração*: é a representação do número de vezes que a população passou pelo processo evolutivo;
- *Adaptação*: é a média dos resultados do valor de aptidão de cada indivíduo;
- *Grau de convergência*: representa o quão próxima a média de adaptação da geração atual está em relação às gerações anteriores;
- *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
- *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.

Função objetivo

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

É importante que as funções objetivo sejam codificadas para não terem 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 seu tempo de execução muito elevado, o algoritmo genético pode ficar lento a ponto de se tornar inviável.

Processo evolutivo

A Figura 48 ilustra o processo evolutivo de um AG. Nas seções a seguir são explicadas cada etapa desse processo (GOLDBERG, 1989; HOLLAND, 1975; MICHALEWICZ, 1996).

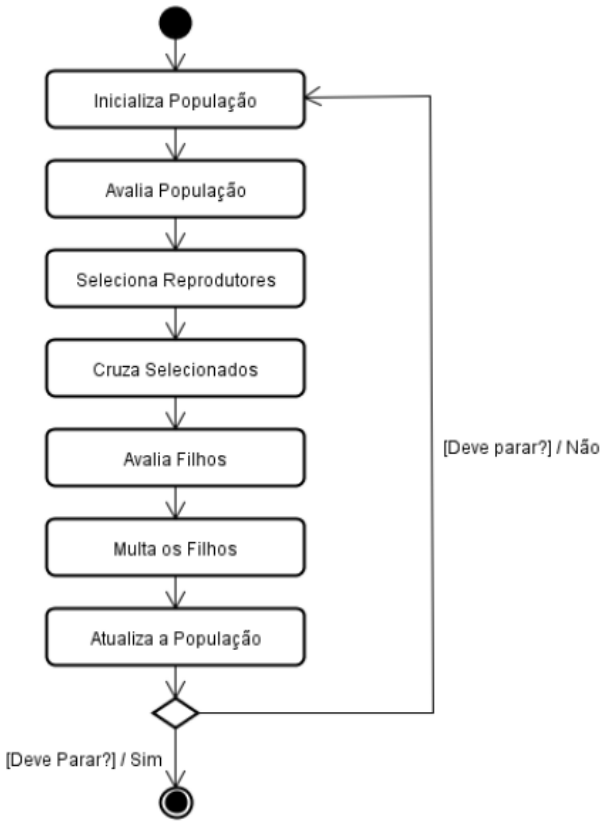


Figura 48 – Processo de execução do algoritmo genético
Fonte: Elaborada pelo autor.

Inicialização

A primeira etapa de um AG é a inicialização, que 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 melhor desempenho (GOLDBERG, 1989). Os operadores mais tradicionais de inicialização, segundo Goldberg (1989), são:

- *Inicialização aleatória uniforme*: esse operador atribui um valor do conjunto de alelos para cada gene do indivíduo que é sorteado de forma aleatória e uniforme;
- *Inicialização aleatória não uniforme*: neste operador, a escolha do valor a ser atribuído ao gene depende da frequência dele em relação aos demais valores;
- *Inicialização aleatória com dope*: indivíduos otimizados são inseridos na população. A presença desses indivíduos pode guiar o AG a uma convergência prematura; e
- *Inicialização parcialmente enumerativa*: os indivíduos são inseridos de tal forma que a população começa o processo evolutivo possuindo todos os formatos possíveis de determinada ordem.

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).

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 (DAWKINS, 1996; DE JONG, 2006; GEYER-SCHULZ, 1997; GOLDBERG, 1989).

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 (GEYER-SCHULZ, 1997; GOLDBERG, 1989).

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

Finalmente, a forma uniforme define que todos os indivíduos têm a mesma probabilidade de serem selecionados. Apesar de essa forma possibilitar maior pesquisa do espaço de soluções, esse tipo de seleção possui uma chance menor de realizar uma melhora durante o cruzamento, de acordo com Goldberg (1989) e Geyer-Schutz (1997).

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 autofertilização.

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 autofertilização o indivíduo é cruzado com ele mesmo.

O cruzamento nem sempre ocorre com os elementos que foram eleitos para reproduzirem, sendo que 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 algum 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 dez posições, o número sorteado deve estar entre o intervalo de zero a dez.

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 49 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.

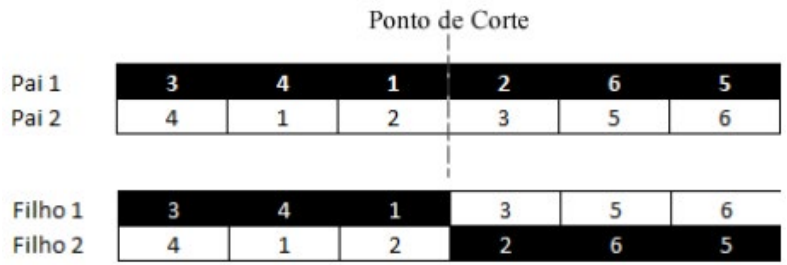


Figura 49 – Cruzamento com o operador 1PX de indivíduos compostos por um vetor de seis posições de números que variam de um a seis

Fonte: Elaborada pelo autor.

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 50 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.

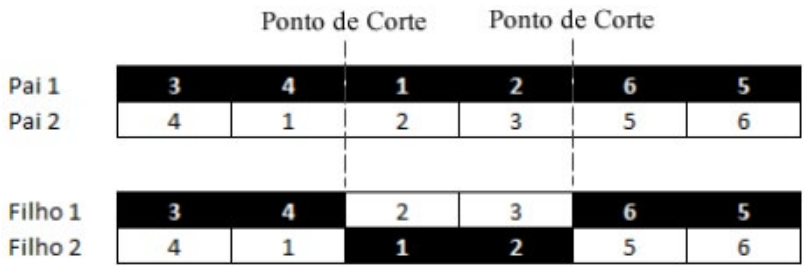


Figura 50 – Cruzamento com o operador MPX de indivíduos compostos por um vetor de seis posições de números que variam de um a seis

Fonte: Elaborada pelo autor.

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.

Mutação

A fase de mutação inicia-se 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 a 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* (GEYER-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 pares de genes, e cada par sorteado é permutado entre si. Finalmente, na mutação *creep* um valor aleatório é somado ou subtraído do valor do gene (DAWKINS, 1996; DE JONG, 2006; GEYER-SCHULZ, 1997; GOLDBERG, 1989).

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 (GEYER-SCHULZ, 1997; GOLDBERG, 1989).

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, sendo que 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 (DE JONG, 2006; GEYER-SCHULZ, 1997; GOLDBERG, 1989).

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 (GOLDBERG, 1989; GEYER-SCHULZ, 1997; DE JONG, 2006).

Finalização

Na fase final do AG não é executada nenhuma operação genética, 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

Teclado Virtual Assistivo Evolutivo (Teclae)

A primeira revisão sistemática realizada para desenvolver a pesquisa de que provém este livro possuía como tema central a Tecnologia Assistiva (TA) 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 dessa pesquisa foi elaborar o módulo responsável pela comunicação, mais especificamente o teclado virtual assistivo.

Para elucidar o estado da arte e esclarecer os vários requisitos do teclado assistivo, foi elaborada uma nova revisão sistemática. O ponto central dessa revisão foram 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 de a revisão sistemática sobre teclados assistivos ter identificado diversos métodos de otimizar esse teclado, não foi encontrada uma maneira de otimizar a distribuição das letras entre as teclas do teclado virtual, tampouco 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 apresentada neste livro.

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.

Características do Teclado Virtual Assistivo Evolutivo (Teclae)

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 51 mostra as características destacadas no Capítulo 2 e o valor de cada característica selecionada para compor o teclado virtual assistivo.

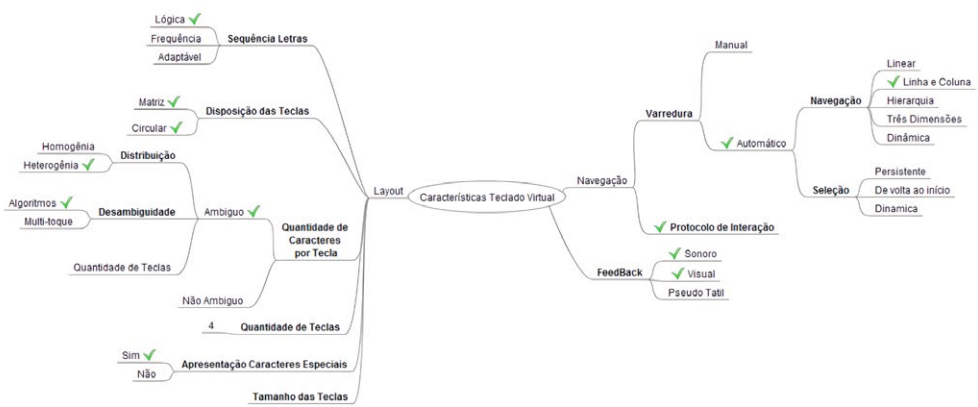


Figura 51 – Características selecionadas para o teclado virtual assistivo
Fonte: Elaborada pelo autor.

Nota: Os valores das características selecionadas estão marcados com o sinal de visto (✓)

Em negrito na Figura 51 estão destacadas as características do teclado virtual, os valores que possuem um sinal de visto (✓) 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, e a explicação de cada escolha é realizada neste capítulo.

Quantidade de caracteres por tecla

Teclados virtuais ambíguos possuem performance de digitação melhor do que os teclados não ambíguos (MOLINA *et al.*, 2009; TOPAL; BENLIGIRAY; AKINLAR, 2012). 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 à quantidade de caracteres por tecla.

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, 2002). 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 determinado 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 tecla com sete caracteres e a terceira e a quarta com seis caracteres.

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. Entretanto, incluir esses caracteres ao teclado pode aumentar o número de teclas, 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 seção "Protocolo de interação". 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 52 mostra a disposição desses componentes.

Figura 52 – Conjunto de teclas de escrita e conjunto de teclas de edição
Fonte: Elaborada pelo autor.

A tecla “Apagar” permite ao usuário remover o caractere anterior à letra. Essa tecla foi posicionada estrategicamente para permitir ao usuário apagar sequencialmente os caracteres. A tecla “Parágrafo” 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 53 ilustra essa matriz. Para permitir ao usuário selecionar esses caracteres é utilizado o método de varredura linha e coluna.

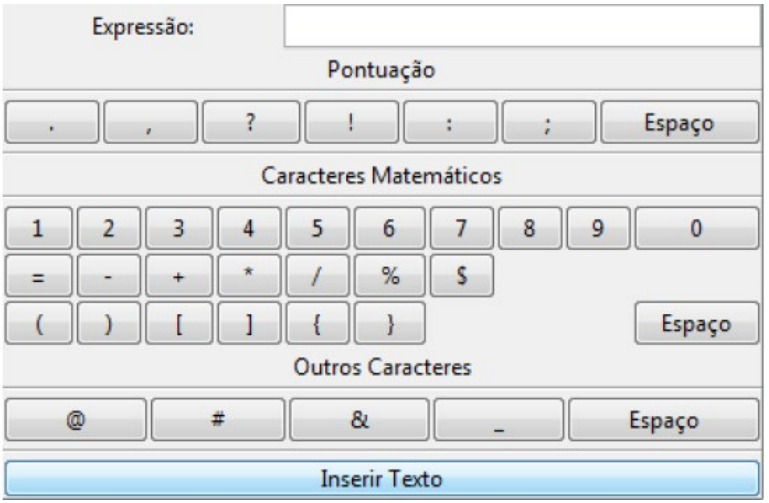


Figura 53 – Matriz de caracteres especiais disponibilizada pelo sistema para inserção desse tipo de caractere

Fonte: Elaborada pelo autor.

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 ultiliza a varredura linha e coluna e por essa funcionalidade o paciente pode inserir novas palavras. A Figura 54 mostra o teclado virtual não ambíguo.

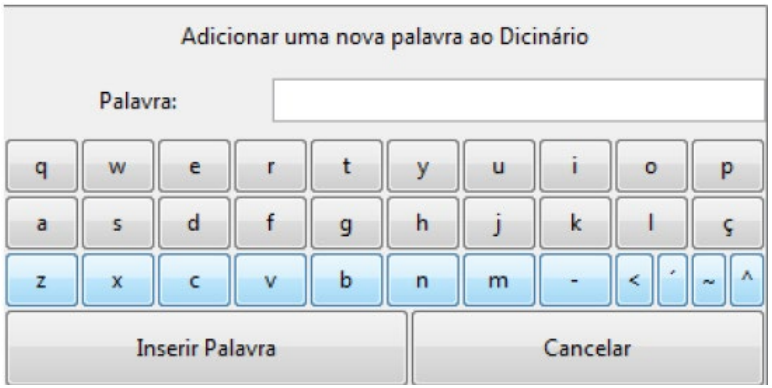


Figura 54 – Interface do teclado virtual disponibilizada para inserção de novas palavras pelo usuário

Fonte: Elaborada pelo autor.

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

Sequência das letras

A distribuição de letras utilizando suas frequências é uma técnica promissora para aumentar a performance de digitação (ZHAI; HUNTER; SMITH, 2000). Porém, a distribuição das letras em forma alfabética é mais usada para usuários deficientes, mesmo que esse tipo de distribuição limite a performance de digitação (TOPAL; BENLIGIRAY; AKINLAR, 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 distribuísse as letras entre as teclas de maneira ótima. Essa pesquisa foi apresentada no Capítulo 2. A partir dessa revisão, desenvolveu-se uma metodologia de otimização evolutiva de teclados virtuais que é apresentada na seção "Metodologia evolutiva de teclados virtuais".

A metodologia proposta é responsável por realizar a distribuição das letras entre as teclas.

Método de seleção

Para realizar a seleção das teclas é utilizada 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.

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.

Método de varredura

O método de varredura adotado é linear, automático e assim que uma tecla for escolhida volta-se 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, pois necessita apenas de um tipo de estimulo do usuário: seleção (MOLINA; RIVERA; GÓMEZ, 2009). Infelizmente, as técnicas de varredura são lentas,

atingindo de cinco a vinte caracteres por minuto (TANAKA-ISHII; INUTSUKA; TAKEICHI, 2002).

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 em 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 elevado número de erros efetuados pelo usuário, esse tempo pode ser reajustado.

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; BERNABEU-SOLER, 2009).

Como o tamanho das teclas e a distância entre elas não é importante para o teclado virtual assistivo apresentado neste livro, 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.

Disposição das teclas

Segundo a Lei de Hick (1952) e 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, 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, expõe-se neste livro a proposta de um *layout* que mescla a disposição das teclas em matriz e círculo. A Figura 55 mostra esse *layout* misto.

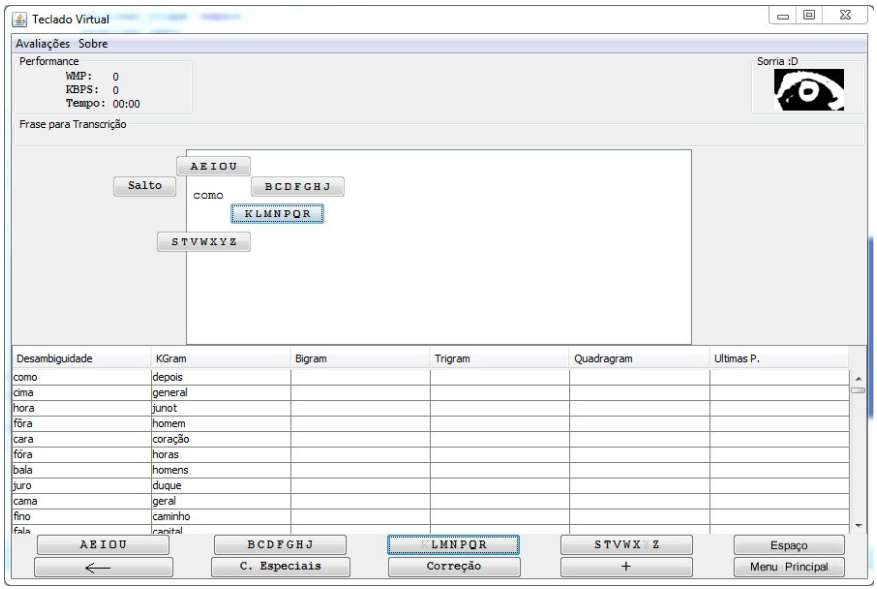


Figura 55 – *Layout* misto do Teclado Virtual Assistivo Evolutivo (Teclae)

Fonte: Elaborada pelo autor.

Nota: 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, e seu *layout* 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 em forma de matriz mostra todas as teclas necessárias para compor um texto (SHARMA *et al.*, 2012). Adotou-se essa abordagem 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.

Feedback do teclado

Os usuários digitam mais rápido se possuírem os dois tipos de *feedback,* visual e auditivo (MAJARANTA *et al.*, 2003). Assim, na pesquisa apresentada neste livro 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. O *feedback* pseudotátil não é utilizado, pois o sistema foi construído para funcionar em computadores pessoais.

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 (POLÁČEK; MÍKOVEC; SLAVÍK, 2012). O objetivo desta seção é apresentar os métodos de predição de letras e palavras utilizados no Teclae.

Dicionário

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

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, Moreira Filho e Alambert (2014). Esse apanhado de textos possui mais de 4 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, 2006). 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 à 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 na pesquisa foi proposto por Orengo e Huyck (2001) e é conhecido por Removedor de Sufixos para a língua portuguesa (RSLP). A Figura 56 ilustra o funcionamento do RSLP.

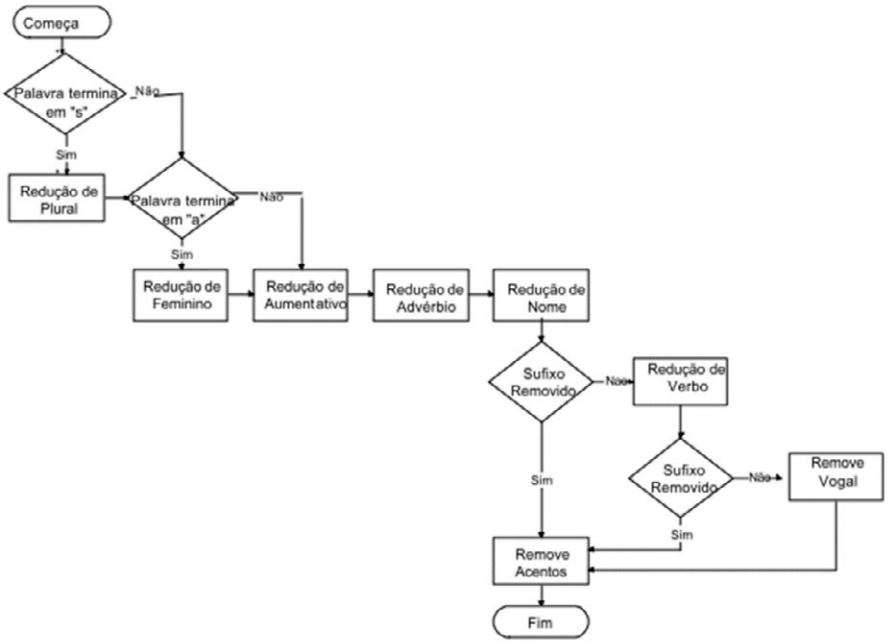


Figura 56 – Funcionamento do algoritmo RSLP
Fonte: Lopes (2004).

O processo do RSLP é composto de diversas regras, sendo que 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* (2015) e *Michaelis* (2015).

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

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 e 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 certa 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 a dez, e a palavra assistivo tenha frequência de ocorrência igual a vinte. O peso de utilização dos nós de “a” até “v” será igual a trinta, enquanto os nós “a” a “o” terão pesos iguais a dez e vinte, 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 57 ilustra a representação de um dicionário com as palavras: assistiva, assistivo, assistir, assista, assiste, assunto, assuntar, aumentativa, aumento, aumentar e auto.

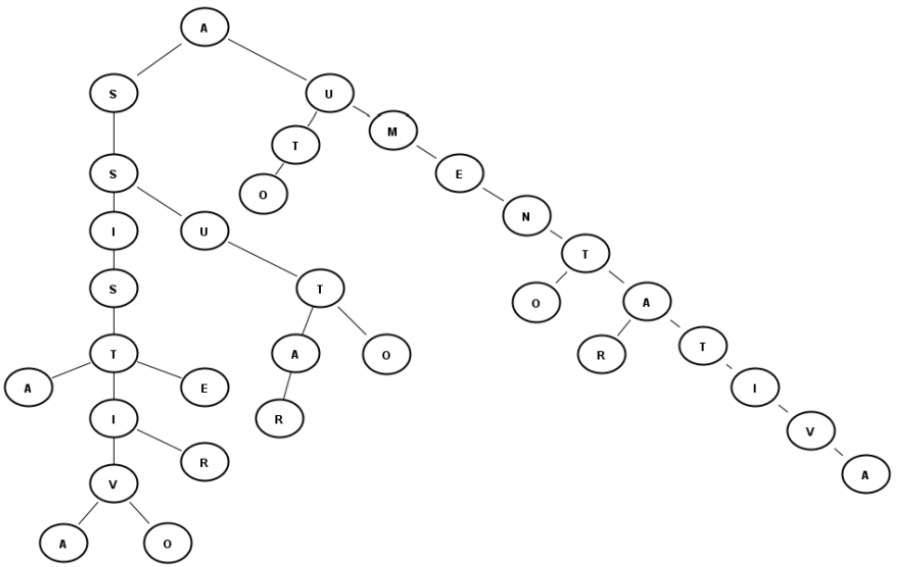


Figura 57 – Representação do dicionário em árvore
Fonte: Elaborada pelo autor.

Nota: Nesta árvore estão representadas as palavras: assistiva, assistivo, assistir, assista, assiste, assunto, assuntar, aumentativa, aumento, aumentar e auto.

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 na pesquisa, não foi encontrada nenhuma técnica de predição que fosse ideal ou muito superior a outra. Portanto, resolveu-se adicionar diversas técnicas de predição ao teclado virtual apresentado neste livro.

Predição de palavras

Com a finalidade de otimizar a técnica de predição, o Teclae 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, sendo eficientes para diminuir o esforço do usuário (GARAY-VITORIA; ABASCAL, 2006). 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 cinco a quinze termos.

Como o Teclae 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.

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 têm performance de digitação melhor do que os teclados que utilizam multitoque (MOLINA; RIVERA; GÓMEZ, 2009).

A performance de digitação é mais eficiente usando algoritmos, pois os métodos multitoque exigem quantidade maior de interação com o usuário para realizar a desambiguidade tecla a tecla (TANAKA-ISHII; INUTSUKA; TAKEICHI, 2002). 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; BERNABEU-SOLER, 2009). Assim, o usuário pode inserir novas palavras usando um teclado não ambíguo.

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 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, sendo cada uma 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 subárvore do léxico correspondente à sequência decodificada, para implementar essa etapa foi desenvolvida a função montarArvoreSugestao 2. A terceira etapa verifica as possíveis

palavras existentes nesta subárvore. 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 à tecla que foi digitada. Por exemplo, considerando um teclado que tenha a distribuição de teclas em ordem alfabética, como o teclado ilustrado na Figura 58. 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 58 – Teclado de quatro teclas com as letras distribuídas entre as teclas em ordem alfabética

Fonte: Elaborada pelo autor.

Se o valor da sequência é uma estrutura composta, então não é necessária 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 subárvores do dicionário e o léxico do sistema. Essa função inicia

atribuindo a primeira sequência de caracteres da sequência decodificada a uma variável denominada.

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, tem-se como parâmetro a sequência de caracteres menos o primeiro elemento, a subárvore que está sendo gerada e, finalmente, a subárvore do léxico que está em processo de replicação. Ao final de todas as chamadas recursivas, a variável Adec contém um conjunto de subárvores 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 subárvore Adec, concatenando as letras que estão registradas nos nós dessa árvore.

Algoritmo 2: ALGORITMO DE CRIAÇÃO DAS SUBÁRVORES 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 ←Sdec1
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
```

Assim que o algoritmo encontra um nó que possua o atributo 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 subárvores de Adec.

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
```

A lista de sugestão é ordenada de acordo com a frequência de uso. Se a palavra nunca foi utilizada pelo paciente o termo é ordenado pela frequência de ocorrência da palavra no *corpus* da língua. O Algoritmo 4 realiza todo o processo de desambiguidade, como apresentado a seguir.

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 no trabalho de que provém este livro, além de ser independente da quantidade de teclas, 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.

Algoritmo de desambiguidade das letras

O Teclado Virtual Assistivo Evolutivo (Teclae) pode proporcionar alto grau de ambiguidade entre as palavras. O nível de ambiguidade do teclado virtual está diretamente relacionado ao número de teclas, à distribuição das letras entre essas teclas e ao léxico do sistema. Um teclado que possua um número elevado de colisões entre as palavras pode diminuir a performance de digitação do usuário.

O sistema exposto neste livro 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 58. 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 quinze. 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: 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 mil 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. A Figura 59 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%.

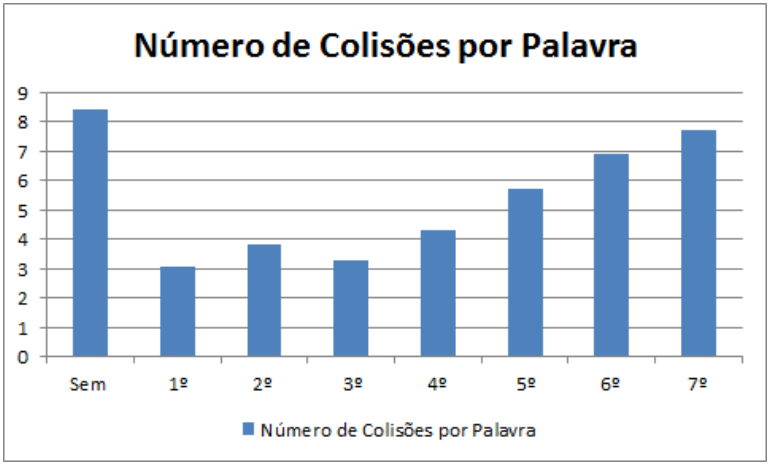


Figura 59 – Gráfico do número de colisões por palavra

Fonte: Elaborada pelo autor.

Nota: 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 multitoque 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 para o Teclae calcula qual 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 esteja utilizando o teclado mostrado na Figura 58 e digite a palavra “tecnologia”.

A sequência correspondente à palavra é 3-1-1-3. A Figura 60 ilustra as subárvores relacionadas a essa sequência.

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
```

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 60 mostra o funcionamento do método de desambiguidade parcial.

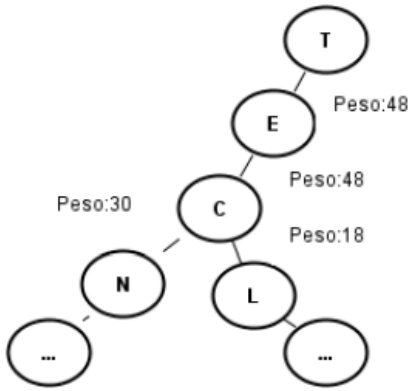


Figura 60 – Subárvore para sugestão da letra
Fonte: Elaborada pelo autor.

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 61 é mostrada ao usuário.

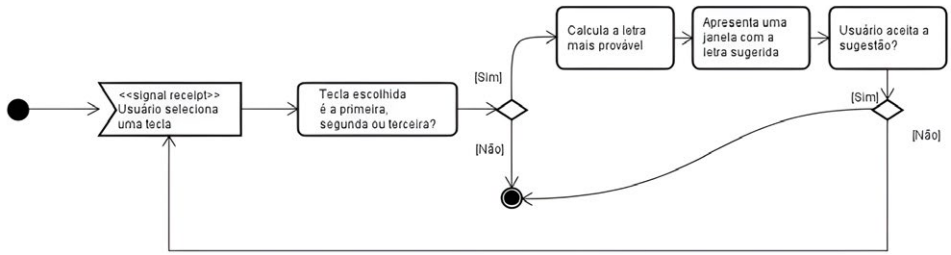


Figura 61 – Método de desambiguidade parcial
Fonte: Elaborada pelo autor.

Na janela mostrada na Figura 62, 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.

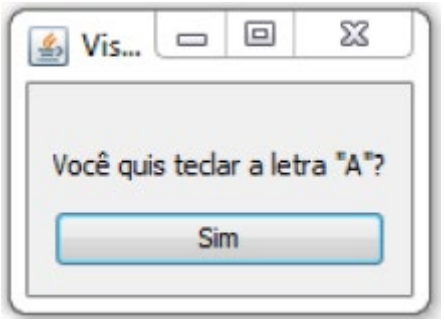


Figura 62 – Janela de desambiguidade
Fonte: Elaborada pelo autor.

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 apresentado neste livro 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.

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 (POLÁČEK; MÍKOVEC; SLAVÍK, 2012; POUPLIN *et al.*, 2014; TOPAL; BENLIGIRAY; AKINLAR, 2012). A alteração da ordem das letras deixa o usuário confuso e dificulta a localização do caractere desejado (POLÁČEK; MÍKOVEC; SLAVÍK, 2012). Entretanto, a longo prazo, desenvolver teclados que se adaptem ao vocabulário e à maneira de escrita do usuário pode aumentar a performance de digitação. Isso porque, a maneira que as letras são distribuídas

entre as teclas influencia diretamente no esforço e na velocidade de digitação (LESHER; MOULTON, 2000). O objetivo da metodologia evolutiva de teclados virtuais descrita neste trabalho 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.

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; 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 (BROUILLETTE; SARMA; KALITA, 2006; GOETTL; BRUGH; JULSTROM, 2005; LEVINE; TREPAGNIER, 1990; 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 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 contínuas.

Os algoritmos genéticos possuem algoritmos que permitem a otimização multiobjetiva. Alguns exemplos de algoritmos genéticos

multiobjetivos 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, e 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 mil indivíduos e foram realizados 9 mil ciclos de evolução. O objetivo da quantidade moderada de indivíduos iniciais é aumentar o espaço de busca do algoritmo, enquanto a finalidade dos 9 mil ciclos é aprimorar consideravelmente a solução. Executando esse algoritmo em uma máquina com oito *gigas* de memória primária e um processador de Intel *Core* i7 foram necessários trinta 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 multiobjetivos. 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.

No trabalho de que resultou este livro, utilizou esse *framework* para agilizar a performance de programação e o processo de otimização. Além disso, adotar um *framework* garante maior robustez no processo. Essa robustez é devida à minimização dos erros de implementação, pois esse framework possui mais de 1.100 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 e 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.

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 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;
- d_{qdt}: quantidade de palavras do dicionário.

Função objetivo do esforço de varredura

Para calcular o esforço de varredura é necessário primeiro definir a função que determina qual tecla pertence à determinada letra. A Equação 14 define essa função.

$$f_{pt}(t, p, i) = arg_n \{t_n = p_i\} \quad \text{Equação 14}$$

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

A Equação 14 verifica em qual tecla determinada letra está e retorna o número dessa tecla. Esse número equivale à 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 na quarta tecla.

Para definir o esforço necessário para digitar uma única palavra em determinado teclado, é necessário calcular o somatório do esforço de varredura de todas as letras dessa palavra utilizando esse teclado. A Equação 15 calcula esse esforço.

$$E_{palavra}(t,p) = \sum_{i=1}^{P_{qdt}} f_{pt}(t,p,i) \qquad \text{Equação 15}$$

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 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 16 calcula a frequência de uma palavra em determinado dicionário.

$$f_{fr}(p,d) = f_d(p) \qquad \text{Equação 16}$$

A Equação 17 calcula o esforço de varredura de um teclado em um dicionário.

$$E_g(t,d) = \sum_{i=1}^{d_{qdt}} E_{palavra}(d_i,t,d) \times f_{fr}(d_i,d) \qquad \text{Equação 17}$$

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

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 à 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 18 determina essa sequência.

$$f_{seq}(t,p) = [f_{pt}(t,p,1) \dots f_{pt}(t,p,p_{qdt})] \quad \text{Equação 18}$$

Para verificar se a sequência de uma palavra é igual à sequência de outra palavra é necessário comparar as duas sequências. A Equação 19 mostra este cálculo.

$$f_{com}(t,p,p') = \begin{cases} 1, se\ f_{seq}(t,p) = f_{seq}(t,p') \\ 0, se\ f_{seq}(t,p) \neq f_{seq}(t,p') \end{cases} \quad \text{Equação 19}$$

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 20.

$$C_{qdt}(p,t,d) = \sum_{i=1}^{d_{qdt}} f_{com}(t,p,d_i) \quad \text{Equação 20}$$

A Equação 20 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 21.

$$f_{com}(p,t,d) = \begin{cases} 1, se\ C_{qdt}(p,t,d) \geq 5 \\ 0, se\ C_{qdt}(p,t,d) < 5 \end{cases} \quad \text{Equação 21}$$

Finalmente, para calcular o índice de ambiguidade do teclado, basta realizar o somatório da Equação 21 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 22.

$$C_{total}(t,d) = \frac{\sum_{i=1}^{d_{qdt}} f_{cont}(d_i,t,d)}{d_{qdt}} \times 100 \quad \text{Equação 22}$$

Processo de evolução do teclado

A Figura 63 ilustra o processo de evolução do teclado assistivo.

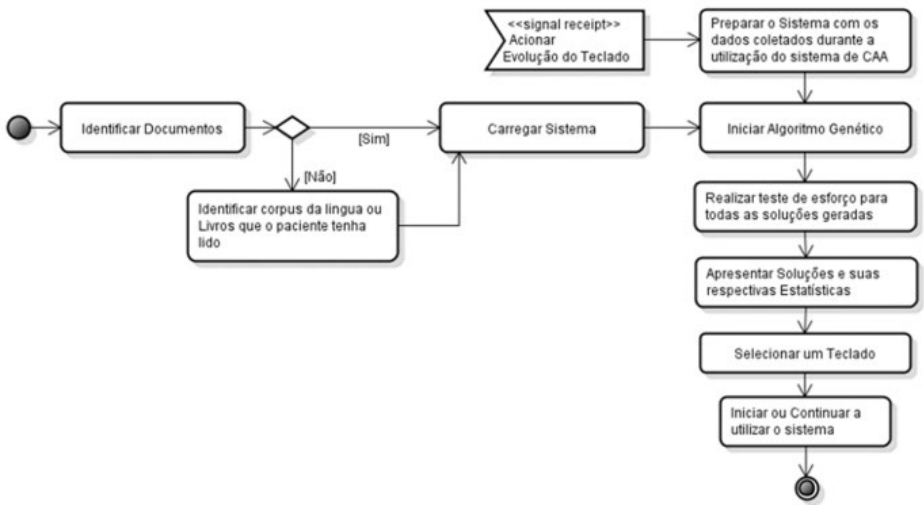


Figura 63 – Processo da metodologia evolutiva de teclados virtuais

Fonte: Elaborada pelo autor.

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, que 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 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 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.

Função de dissemelhança

Usuários que possuem certa familiaridade com determinado tipo de teclado, tendem a ter 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 substituído (EGGERS *et al.*, 2003; LEVINE; TREPAGNIER, 1990; LIGHT; ANDERSON, 1993).

Entretanto, para pessoas com SE o mais importante é que o teclado transmita a informação de maneira rápida e com mínimo de esforço. Para esse tipo de usuário, a alteração do *layout* do teclado pode trazer mais vantagens do que desvantagens. Contudo, a mudança de paradigma requer um tempo de adaptação e muitas vezes pode ser árdua. Para auxiliar nessa transição, foi desenvolvido um algoritmo que mostra o grau de diferença entre os teclados virtuais. 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 Figura 64. Para calcular o valor de dissemelhança verifica-se 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 64 – Teclado alfabético com permutações entre as letras A, Z e S
Fonte: Elaborada pelo autor.

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.

Métricas do teclado virtual

Os dados de medição de tecla de varredura são imprecisos, e isso dificulta a comparação entre os teclados virtuais (MIRÓ-BORRÁS; BERNABEU-SOLER, 2009). Atribuiu-se essa dificuldade às 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 para aferir a performance do teclado virtual será o WPM. Entretanto, de acordo com Mackenzie 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

Avaliação da metodologia e resultados obtidos

Com o objetivo de comparar o método de otimização dos teclados tradicionais e a metodologia de evolução dos teclados assistivos proposta na pesquisa de que resultou este livro, foi elaborado e executado um experimento científico. Este capítulo apresenta o protocolo, a coleta, a análise e as conclusões obtidas nesse experimento, cuja metodologia foi fundamentada em Wohlin *et al.* (2012).

Protocolo do experimento

Objetivo

Existem diversas formas de definir os objetivos. Basili e Rombach (1988) propuseram que os objetivos fossem definidos considerando o formato apresentado no Quadro 3.

Quadro 3 – Formato para a definição do objetivo

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

Fonte: Elaborado pelo autor.

O Quadro 4 mostra o objetivo deste experimento. Esse objetivo é descrito de acordo com a definição apresentada no Quadro 3.

Quadro 4 – 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 de	Pesquisador
No contexto do	Desenvolvimento dos teclados assistivos para pacientes com SE.

Fonte: Elaborado pelo autor.

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):

- on-line ou off-line;
- estudantes ou profissionais;
- problema fictício ou problema real;
- específico ou geral.

O experimento executado na pesquisa apresentada neste livro foi orientado pela tecnologia e foi realizado inteiramente em 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 teclado virtual assistivo ser uma questão real, esse problema foi desenvolvido e realizado em ambiente virtual. Logo, os sujeitos para esse experimento são todos agentes virtuais. Isso impossibilita definir a segunda dimensão para este experimento.

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 descrita neste livro.

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, diferente do método tradicional, prevê como entrada para a meta-heurística um conjunto de textos que o usuário em alguma época tenha digitado ou 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.

Formulação das hipóteses

A hipótese é uma predição ou tentativa de afirmação sobre a relação entre as variáveis. Ela é apresentada normalmente nas pesquisas quantitativas como a exposta neste livro (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 aquela aceita caso a hipótese nula seja rejeitada.

As hipóteses deste experimento são apresentadas a seguir:

- *Hipótese 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 descrita neste livro.
- *Hipótese 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.

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 com essas pessoas levaria bastante tempo. Isso porque seria necessário finalizar a construção do teclado assistivo e, 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 quantidade suficiente de dados para avaliar a metodologia.

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: disponibilidade dos textos em formato digital e gratuito; e 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. O Quadro 5 mostra a ordem de análise de cada escritor, o nome dos autores e a quantidade de obras analisadas por escritor.

Quadro 5 – 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
Guimarães Rosa	7
Paulo Coelho	13
Régine Deforges	8

Fonte: Elaborado pelo autor.

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 personalizados, variável independente, expostos neste livro.

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.

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 quatro 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.

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.

A Equação 24 mostra o cálculo do esforço de digitação de um texto. Esse cálculo aplica a Equação 23 a todas as palavras desse texto.

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:

- Contar a quantidade de teclas necessárias para digitar essa palavra (f_{ct}) utilizando um teclado t;
- 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;
- Se a palavra desejada é a primeira na lista de ambiguidade, então soma-se mais cinco, para simular a seleção da tecla de espaço;
- Se não é a primeira palavra, soma-se seis 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.

Equação da métrica:

$$f_c(t,p) = \begin{cases} f_{cv}(t,p) + f_{ct}(p) + 1, se\ f_a(t,p) \leq 1 \\ f_{cv}(t,p) + f_{ct}(p) + 6 + f_{ip}(t,p), se\ f_a(t,p) \leq 1 \end{cases}$$ Equação 23

Em que: é o teclado que está sendo analisado, representa a palavra que está sendo analisada, é a função que calcula o esforço de varredura, representa a função que calcula a quantidade de teclas necessárias para compor a palavra, é a função que calcula o índice da palavra na lista de desambiguidade e 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.

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 (23) em todos os termos do texto em análise.
Forma de coleta:
Aplicar a métrica "esforço de digitação de uma palavra" para todas as palavras do texto f_{ct} (t).
Equação da métrica:

$$f_{ct}(t,o) = \sum_{x=1}^{n_o} f_c(o_p(x))$$ Equação 24

Em que: é o número de palavras do texto, 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.

A Equação 24 mostra o cálculo do esforço de digitação de um texto. Esse cálculo aplica a Equação 23 a todas as palavras desse texto

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 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. O Quadro 6 mostra a relação entre as escalas e a estatística.

Quadro 6 – Escalas por métrica de acordo com Wohlin et al. (2012)

Tipo de escala	Medidas de tendência
Nominal	Moda
Ordinal	Moda, mediana e percentil
Intervalar	Moda, mediana, percentil, média, variância e intervalo
Racional	Moda, mediana, percentil, média, variância, intervalo, média geométrica

Fonte: Elaborado pelo autor.

Para consolidar os dados da pesquisa de que resultou este livro, 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, , fundamentado pelos dados consolidados. A rejeição dessa hipótese possibilita o pesquisador considerar, com determinado grau de confiança, se 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 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 na pesquisa de que proveio este livro (WOHLIN *et al.*, 2012). A Equação 25 mostra o cálculo necessário para realizar o t-Teste:

$$t_{calc} = \frac{\bar{x} - \mu_0}{\frac{S}{\sqrt{n}}} \qquad \text{Equação 25}$$

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;
- *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 26 mostra o cálculo do teste de Kolmogorov Smirnov.

$$D_n = sup_n|F_n(x) - F(x)| \qquad \text{Equação 26}$$

Em que: $F_n(x)$ é a função de distribuição acumulada empírica dos dados, e F_n 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.

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 em que não era possível extrair os textos, como por exemplo, PDFs protegidos por senha ou Epubs.

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.

Relato operacional do experimento

Este experimento foi executado em duas etapas. A primeira foi iniciada a partir de 25 de maio de 2015 e finalizada no dia 20 de julho de 2015. A segunda etapa teve início em 5 de outubro de 2015 e finalizada em 29 de outubro de 2015. A Figura 65 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.

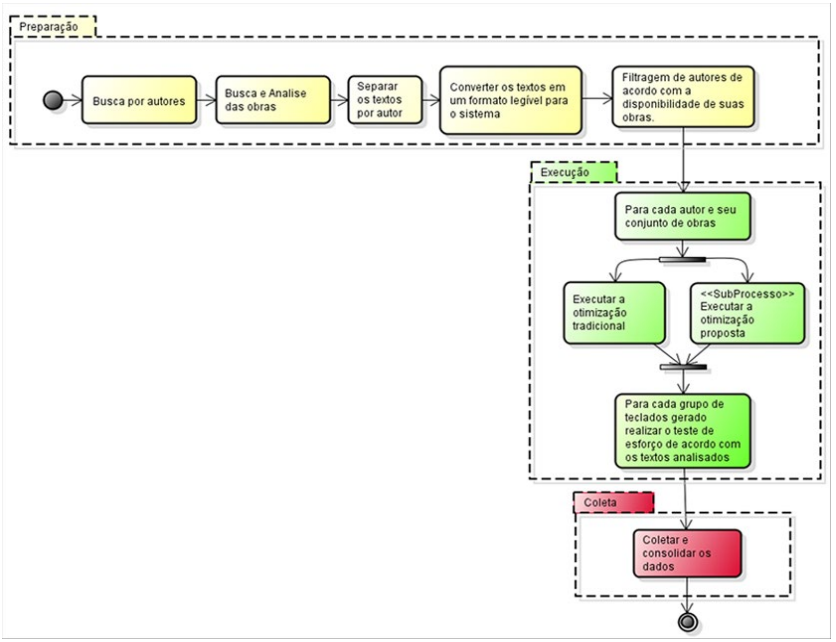


Figura 65 – Processo de execução do experimento
Fonte: Elaborada pelo autor.

Nota: Este processo foi proposto para mostrar a eficiência da metodologia desenvolvida na pesquisa de que resultou este livro.

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 tais como: José de Alencar, Machado de Assis, Clarice Lispector, Carlos Drummond de Andrade, Guimarães Rosa, Paulo Coelho, Jorge Amado, Régine Deforges, J. K. Rowling, dentre outros. Todos os autores citados foram considerados possíveis sujeitos deste experimento.

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. O Quadro 5 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, apresentada neste livro.

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 4 deste trabalho. 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 funcionaria.

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 zero. 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, sendo que 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 24 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 66 ilustra uma linha desta planilha.

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

Figura 66 – Linha da planilha de saída do processo de otimização pelo *corpus*
Fonte: Elaborada pelo autor.

Após executar a otimização com o *corpus* da língua portuguesa, foi realizado o teste com a metodologia proposta na pesquisa. 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. Depois, 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 à 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 67 ilustra uma linha desta planilha.

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

Figura 67 – Linha da planilha de saída do processo de otimização por texto

Fonte: Elaborada pelo autor.

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, as frequências 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.

O Quadro 7 foi construído a partir da consolidação dos dados coletados durante a execução do experimento.

Quadro 7 – 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
Guimarães Rosa	14321419	9066213
Paulo Coelho	1710563	1022712
Régine Deforges	20528559	12362739

Fonte: Elaborado pelo autor.

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.

A Figura 68 ilustra a diferença entre os dois métodos de acordo com cada sujeito, segundo o Quadro 7. Pode-se verificar que o método proposto apresenta melhores resultados do que o método tradicional, sendo que as diferenças entre esses dois variam de 10% no pior caso e 40% no melhor caso.

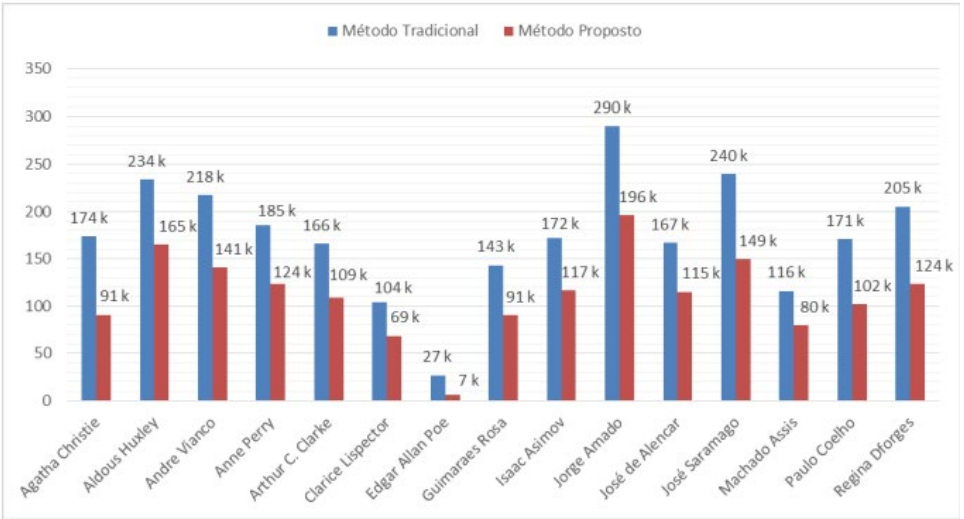


Figura 68 – Comparação entre o esforço de digitação entre os teclados desenvolvidos pelos dois métodos

Fonte: Elaborada pelo autor.

A Figura 69 mostra os resultados da metodologia proposta aplicada sequencialmente às obras dos autores selecionados para o experimento. Com a finalidade de tornar o gráfico mais claro, apenas as primeiras sete evoluções são apresentadas.

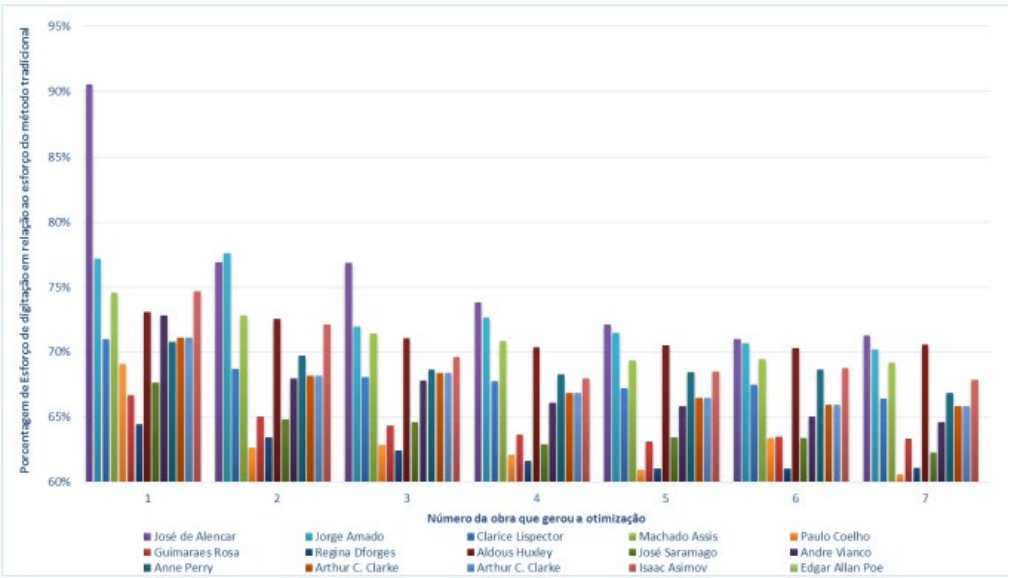


Figura 69 – Evolução da redução do esforço de digitação

Fonte: Elaborado pela autor.

Nota: Este gráfico foi elaborado de acordo com a metodologia proposta na pesquisa comparada com o esforço de digitação realizado pelo método tradicional.

Finalmente, as figuras de 70 a 84 apresentam a evolução dos teclados virtuais de acordo com cada autor.

Figura 70 – Gráfico de Agatha Christie

Fonte: Elaborada pelo autor.

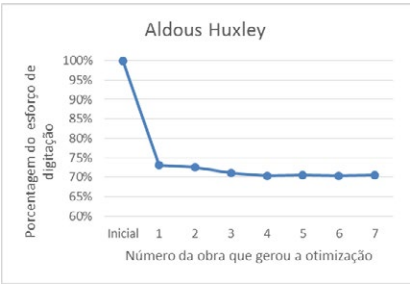


Figura 71 – Gráfico de Aldous Huxley

Fonte: Elaborada pelo autor.

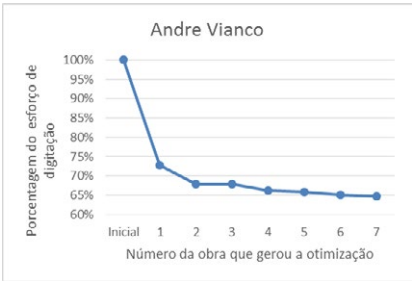


Figura 72 – Gráfico de André Vianco
Fonte: Elaborada pelo autor.

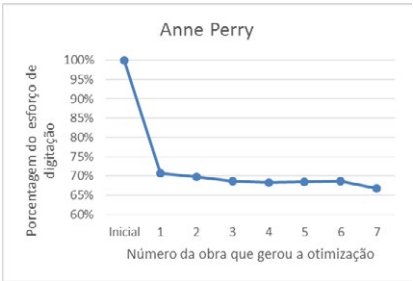


Figura 73 – Gráfico de Anne Perry
Fonte: Elaborada pelo autor.

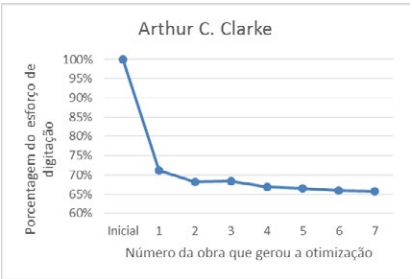


Figura 74 – Gráfico de Arthur Clarck
Fonte: Elaborada pelo autor.

Figura 75 – Gráfico de Clarice Lispector
Fonte: Elaborada pelo autor.

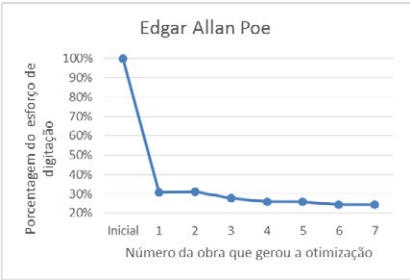


Figura 76 – Gráfico de Edgar Allan Poe
Fonte: Elaborada pelo autor.

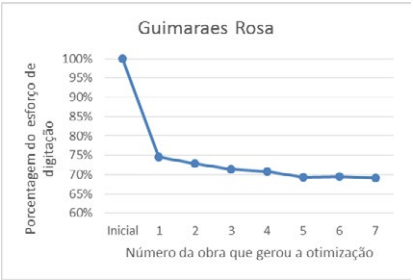


Figura 77 – Gráfico de Guimarães Rosa
Fonte: Elaborada pelo autor.

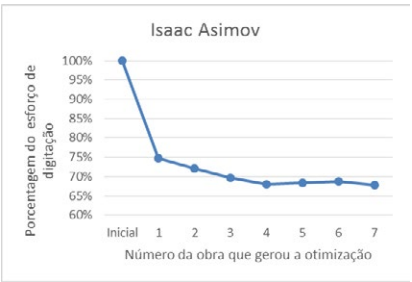


Figura 78 – Gráfico de Isaac Asimov
Fonte: Elaborada pelo autor.

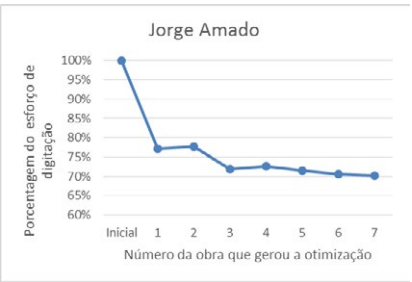


Figura 79 – Gráfico de Jorge Amado
Fonte: Elaborada pelo autor.

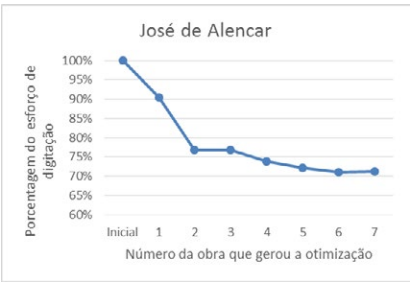


Figura 80 – Gráfico de José de Alencar
Fonte: Elaborada pelo autor.

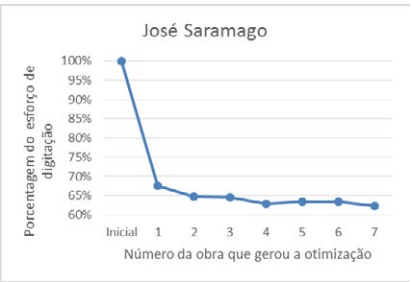


Figura 81 – Gráfico de José Saramago
Fonte: Elaborada pelo autor.

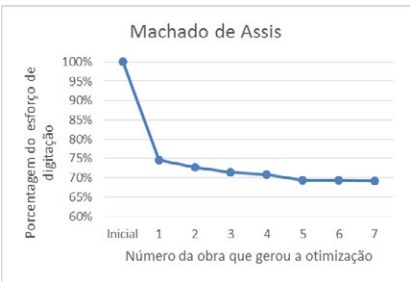


Figura 82 – Gráfico de Machado de Assis
Fonte: Elaborada pelo autor.

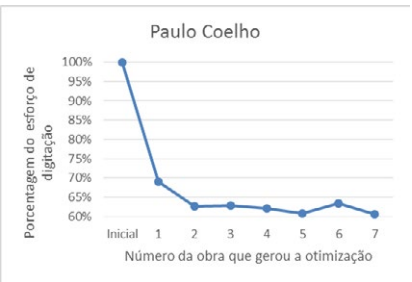


Figura 83 – Gráfico de Paulo Coelho
Fonte: Elaborada pelo autor.

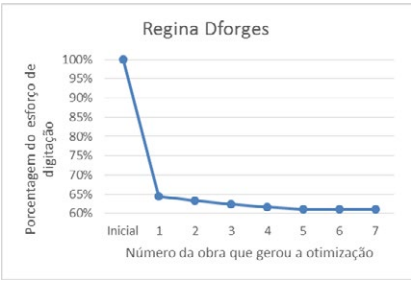


Figura 84 – Gráfico de Régine Deforges
Fonte: Elaborada pelo autor.

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 primeira 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.

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 distribuição normal. A Figura 85 ilustra

o teste de Kolmogorov Sirminov para a amostra dos dados relativos 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 85 – Resultado do teste de Kolmogorov Smirnov para a amostra dos dados relativos ao método tradicional

Fonte: Elaborada pelo autor.

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.

Figura 86 – Resultado do teste de Kolmogorov Smirnov para a amostra dos dados relativos ao método proposto na pesquisa

Fonte: Elaborada pelo autor.

Pode-se verificar na Figura 86 que os resultados obtidos no teste estatístico para o método tradicional e o método apresentado neste livro 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 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 87 mostra o resultado do t-Test das amostras utilizando o sistema SPSS.

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

Figura 87 – Resultado do t-Teste para as amostras dos dados relativos ao método tradicional e ao método proposto na pesquisa

Fonte: Elaborada pelo autor.

O resultado do grau de significância do t-Test para as amostras é igual a 0,000. Esse 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. Logo, é válida a hipótese alternativa, H_1. Então pode 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, exposta neste livro. Pode-se observar na Figura 87 que a diferença entre as médias de esforço dos dois métodos é considerável.

Uma conclusão importante até este momento é a antagonismo entre o esforço de varredura e a taxa de ambiguidade. Nos mais de 2 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.

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 apresentada neste livro possibilita o usuário otimizar o teclado virtual a qualquer momento, de acordo com 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.

Seleção dos sujeitos

Assim como no anterior, foi inviável realizar esse experimento com pessoas devido ao tempo disponível à época para a finalização da pesquisa de doutorado. Para selecionar os sujeitos optou-se por identificar autores

contemporâneos que escrevem sobre diversos assuntos periodicamente no formato digital.

Os critérios de seleção dos autores foram: disponibilidade dos textos em formato digital e gratuito; e 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 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. Dez escritores foram selecionados: Xico Sá, Inácio Araújo, Frederico Vasconcelos, Pedro Diniz, Leandro Colon, Rodolfo Lucena, Alexandre Schwartsman, Antonio Prata, Márcia Dessen e Luiz Felipe Pondé. O site Folha de São Paulo foi escolhido por possuir conteúdo disponibilizado gratuitamente e pela periodicidade com que seus escritores publicam. Os escritores foram selecionados utilizando os critérios de seleção e a diversidade de seus assuntos.

Relato operacional do experimento

Este experimento foi executado a partir de 1 de outubro de 2015 e foi finalizado no dia 2 de novembro de 2015. A Figura 88 mostra o processo desse experimento.

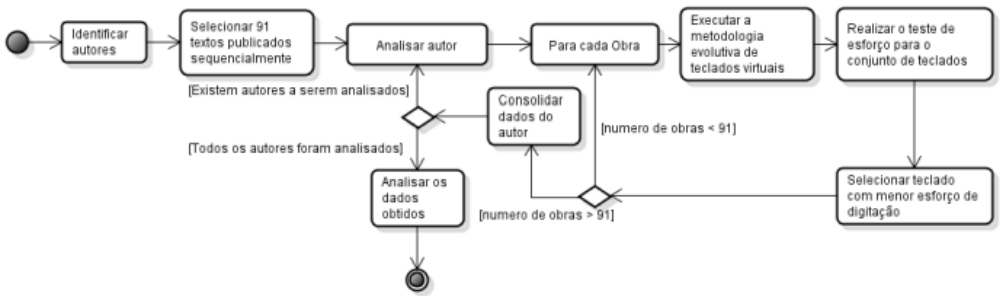


Figura 88 – Processo de execução do experimento

Fonte: Elaborada pelo autor.

Nota: O experimento foi proposto para identificar o ponto de otimização do teclado virtual assistivo.

O experimento iniciou-se com a *identificação dos autores*. Esses escritores foram escolhidos de acordo com os critérios apresentados na seção "Seleção dos sujeitos". 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.

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 e o número de palavras distintas. O esforço de digitação determina a quantidade de interações necessárias entre o usuário e o *software* para digitar seu texto. O número de palavras representa a quantidade de palavras que o texto possui, e o número de palavras distintas define a quantidade de palavras diferentes.

Se o número de textos fosse maior do que 91, os dados do autor eram consolidados e um novo autor era submetido à análise. Assim que todos os escritores foram analisados foi iniciada a *análise dos resultados*.

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 esquerdo 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 íntegra os resultados de otimização dos primeiros trinta textos. Após esses resultados as otimizações são apresentadas em intervalos de dez textos. Além disso, o lado direito do eixo y foi limitado a um valor que possibilitasse o detalhamento dos valores do gráfico nas primeiras trinta evoluções. As figuras de 89 a 98 mostram os gráficos de evolução por autor.

Na Figura 98, 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 2.800 a 3.800 palavras dentre as quais 900 a 1500 eram distintas. Ao final das dez primeiras otimizações, a metodologia evolutiva de teclados assistivos estabilizou. A partir desse ponto, o método proposto possibilitou uma diminuição do esforço de digitação de 1% a 5% em média, a cada 3 mil a 7 mil novas palavras inseridas no sistema.

Conclui-se que a princípio o usuário pode utilizar o método de otimização a cada duzentas a seiscentas 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 3 mil a 7 mil palavras.

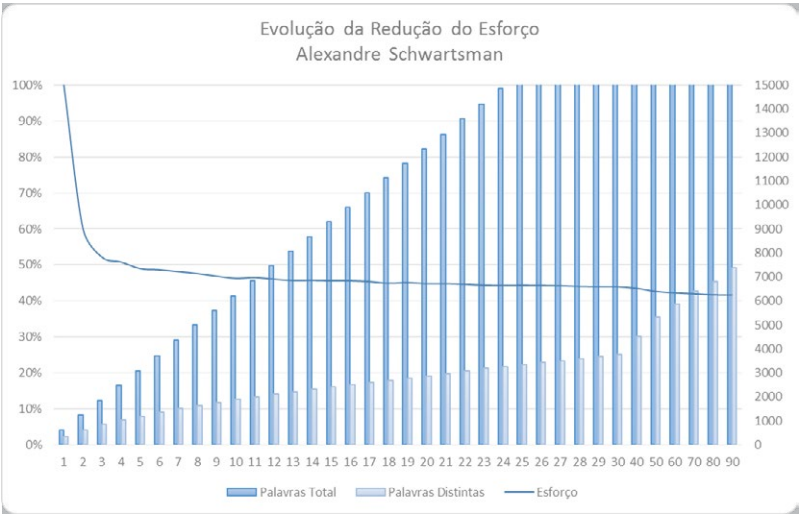


Figura 89 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Alexandre Schwartsman

Fonte: Elaborada pelo autor.

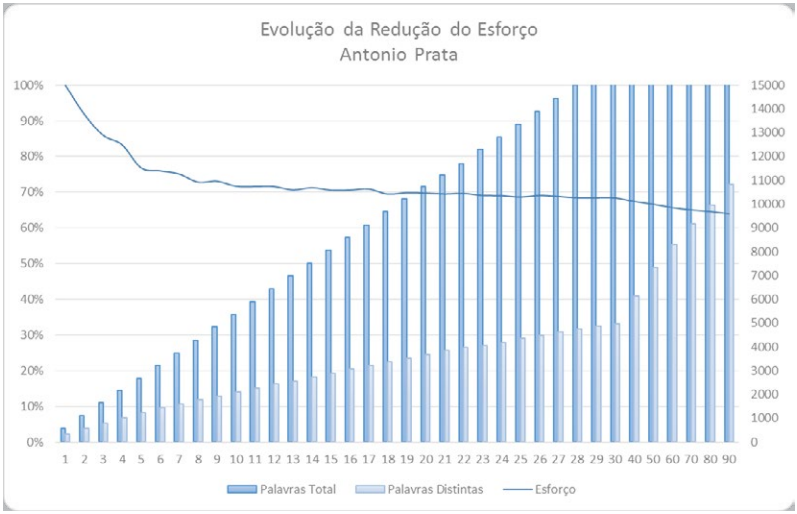


Figura 90 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Antonio Prata

Fonte: Elaborada pelo autor.

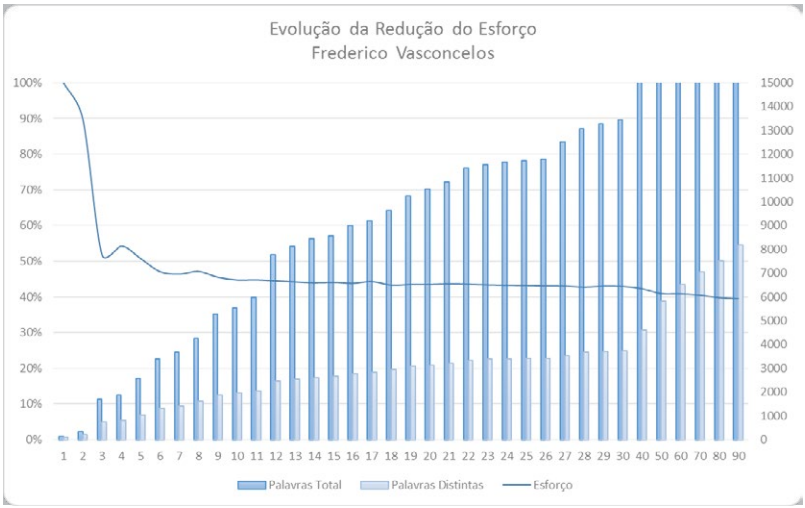


Figura 91 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Frederico Vasconcelos

Fonte: Elaborada pelo autor.

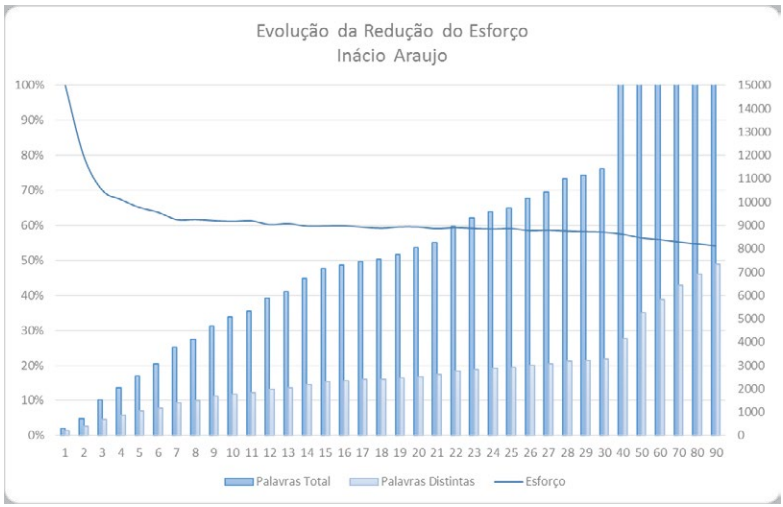


Figura 92 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Inácio Araujo

Fonte: Elaborada pelo autor.

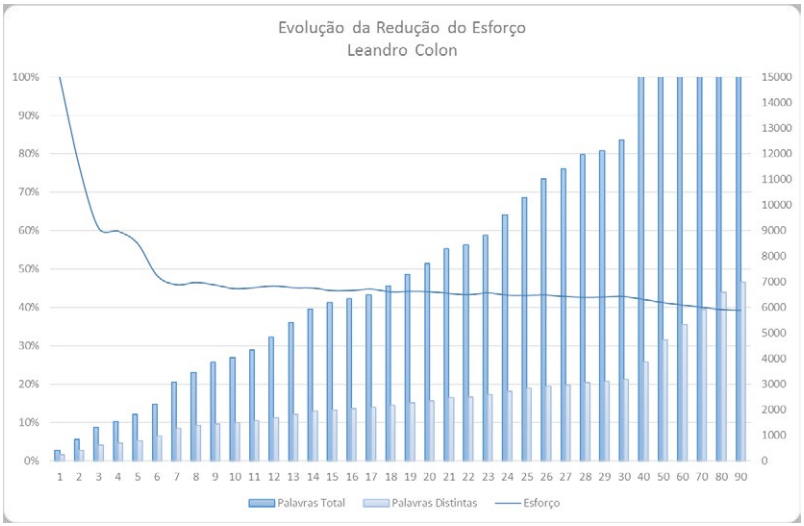


Figura 93 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Leandro Colon

Fonte: Elaborada pelo autor.

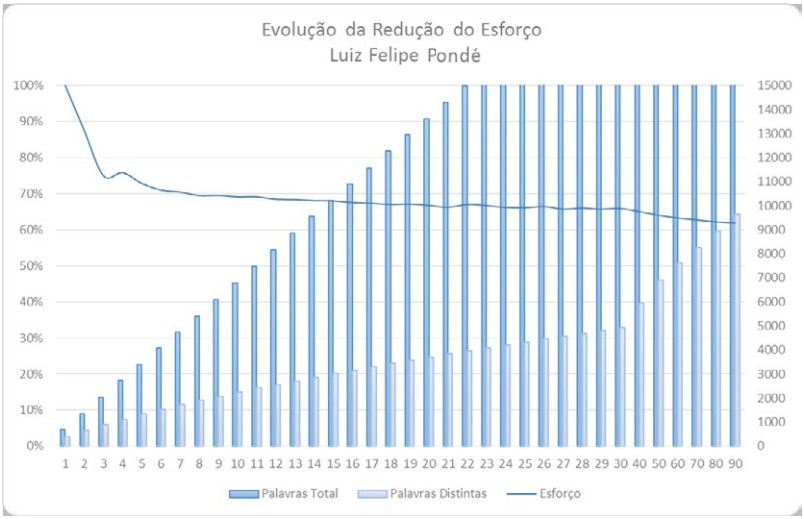


Figura 94 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Luiz Felipe Pondé

Fonte: Elaborada pelo autor.

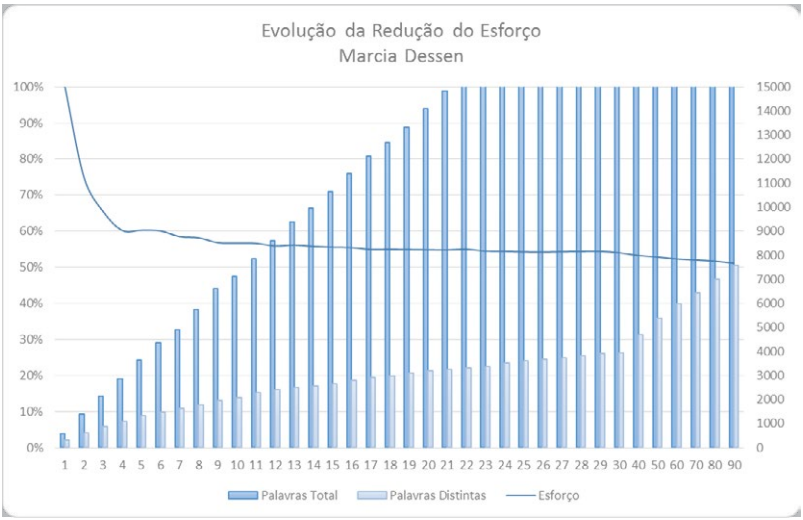


Figura 95 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Marcia Dessen

Fonte: Elaborada pelo autor.

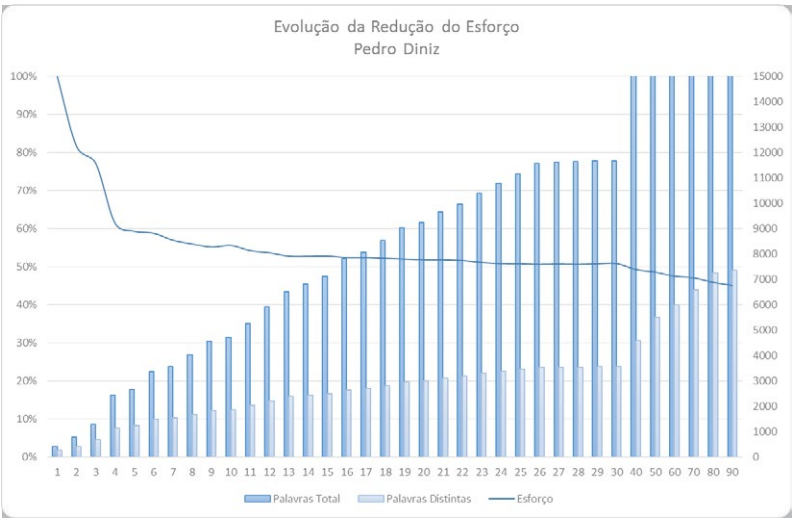


Figura 96 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Pedro Diniz

Fonte: Elaborada pelo autor.

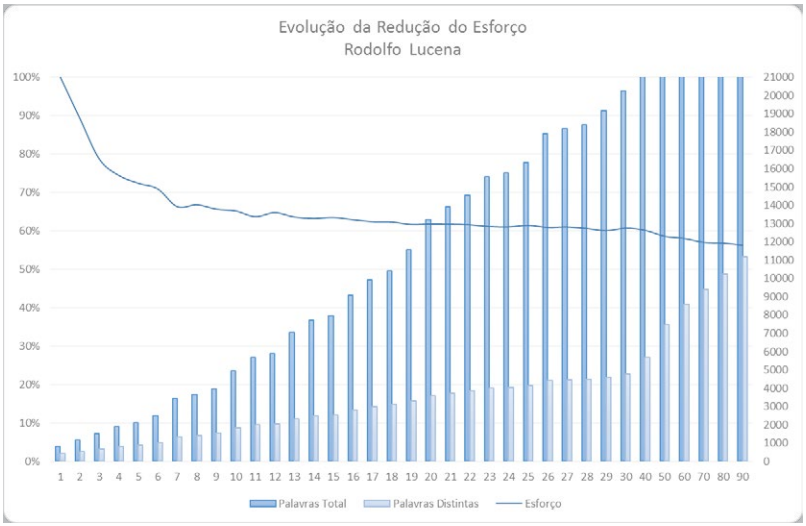


Figura 97 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Rodolfo Lucena

Fonte: Elaborada pelo autor.

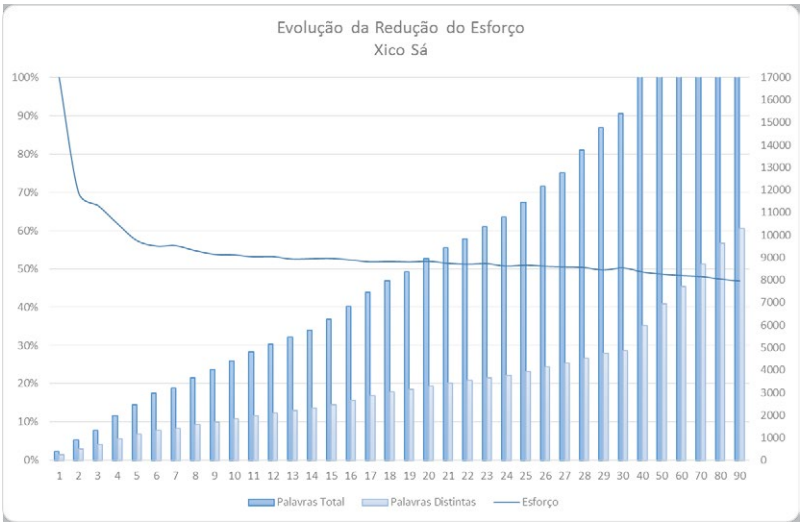


Figura 98 – Gráfico da evolução da redução do esforço de digitação durante a aplicação da metodologia aos textos de Xico Sá

Fonte: Elaborada pelo autor.

Considerações finais

Este livro apresentou a modelagem e o desenvolvimento de um teclado virtual assistivo, compacto, otimizado e evolutivo para pessoas com síndrome do encarceramento. 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 a revisão de literatura, foi apresentada a proposta de um novo teclado virtual assistivo, compacto, otimizado e evolutivo, com a identificação das características mais adequadas para serem utilizadas no novo teclado. Além de modelar o teclado virtual assistivo para adequar-se a pessoas com síndrome do encarceramento, 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 descrita a metodologia evolutiva de teclados assistivos. Essa metodologia, diferentemente 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 foi mostrar a eficiência da metodologia 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 que o esforço provocado pelo uso dos teclados gerados método tradicional. A finalidade do segundo experimento foi identificar o número de palavras necessárias 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. Para testar a metodologia de teclados assistivos foram realizados experimentos virtuais.

Considerando outras propostas de trabalhos para o futuro, é importante que os experimentos sejam efetuados com pacientes com SE. Outro ponto importante é aprimorar os métodos de predição de texto usados pelo teclado virtual. Apesar de esse teclado implementar diversos métodos, não foi implementado o método de predição utilizando gramática. Esse tipo de predição pode ser incluído no teclado virtual permitindo que o sistema se torne mais robusto.

Além do teclado virtual assistivo e da metodologia de teclados virtuais apresentada neste livro, foi desenvolvido um método de desambiguidade parcial. É importante que seja realizado um experimento científico para comprovar a efetividade desse método, verificando quanto ele pode diminuir o esforço de digitação e aumentar a performance de entrada de dados. Importante também é desenvolver o restante do módulo de comunicação da arquitetura do sistema de CAA. Para completar o módulo proposto, é necessário que seja desenvolvida uma planilha de comunicação que integre o Teclado Virtual Assistivo Evolutivo (Teclae).

Referências

Al-abdullatif, Aljawharah *et al.* Mind-controlled augmentative and alternative communication for people with severe motor disabilities. *In*: international conference on innovations in information technology, 9., 2013, Al Ain. *Proceedings* [...]. Al Ain: [*s. n.*], 2013. p. 107-112.

Al faraj, Khaldoun; Mojahid, Mustapha; Vigouroux, Nadine. 3DKey: an accordion-folding based virtual keyboard for small screen. *In*: gross, Tom *et al.* (ed.). *Human-computer-interaction-interact 2009*. Uppsala: Springer Berlin Heidelberg, 2009a. p. 634-644.

Al faraj, Khaldoun; Mojahid, Mustapha; Vigouroux, Nadine. BigKey: a virtual keyboard for mobile devices. *In*: jacko, Julie A. (ed.): *Human-computer-interaction*. Uppsala: Springer Berlin Heidelberg, 2009b. p. 3-10.

Al faraj, Khaldoun; Vigouroux, Nadine; Mojahid, Mustapha. SmartKey: a multi purpose target expansion based virtual keyboard. *In*: international acm sigaccess conference on computers and accessibility, assets, 11., 2009, Pittsburgh. *Proceedings* [...]. Pittsburgh: acm, 2009. p. 251-252.

Ann, Ong C.; Lau, Bee T. A study on the effectiveness of biometrics based alternative communication tool. *In*: international conference on information, communications and signal processing, 8., 2011, Singapure. *Proceedings* [...]. Singapure: [*s. n.*], 2011. p. 1-4.

Arboleda, Carolina *et al.* P300-based brain computer interface experimental setup. *In*: annual international conference of the ieee engineering in medicine and biology society, 31., 2009, Minneapolis. *Proceedings* [...]. Minneapolis: embc, 2009. p. 598-601.

Baas, Maxime *et al.* Système de saisie de texte visant à réduire l'effort des utilisateurs à handicap moteur. *In*: proceedings of the ergonomie et informatique avancée conference, 10., 2010, Biarritz. *Proceedings* [...]. Biarritz: [*s. n.*], 2010. p. 19-26.

Baker, Bruce R. Using images to generate speech. *byte*, [*s. l.*], v. 11, n. 3, p. 160-168, 1986.

Baljko, Melanie; Tam, Andrew. Indirect text entry using one or two keys. *In*: international acm sigaccess conference on computers and accessibility, 8., 2006, Portland. *Proceedings* [...]. Portland: acm, 2006. p. 18-25.

Basili, Victor R.; Rombach, Hans D. The tame project: towards improvement-oriented software environments. *ieee Transactions on Software Engineering*, [*s. l.*], v. 14, n. 6, p. 758-773, 1988.

Belatar, Mohammed; Poirier, Franck. Text entry for mobile devices and users with severe motor impairments: handiglyph, a primitive shapes based onscreen keyboard. *In*: international acm sigaccess conference on computers and accessibility, 10., 2008, Halifax. *Proceedings* [...]. Halifax: acm, 2008. p. 209-216.

Bersch, Rita. *Introdução à tecnologia assistiva*. Porto Alegre: Cedi, 2008.

Besio, Walter G.; Kay, Steven M.; Liu, Xiang. An optimal spatial filtering electrode for brain computer interface. *In*: annual international conference of the ieee engineering in medicine and biology society, 2009, Minneapolis. *Proceedings* [...]. Minneapolis: ieee, 2009. p. 3138-3141.

Bhattacharya, Samit; Basu, Anupam; Samanta, Debasis. Computational modeling of user errors for the design of virtual scanning keyboards. *ieee Transactions on Neural Systems and Rehabilitation Engineering*, [*s. l.*], v. 16, n. 4, p. 400-409, 2008.

Bhattacharya, Samit; Laha, Subrata. Bengali text input interface design for mobile devices. *Universal Access in the Information Society*, [*s. l.*], v. 12, n. 4, p. 441-451, 2012.

Biswas, Pradipta; Samanta, Debasis. Friend: a communication aid for persons with disabilities. *ieee Transactions on Neural Systems And Rehabilitation Engineering*, [*s. l.*], v. 16, n. 2, p. 205-209, 2008.

Blain, Stefanie; Mihailidis, Alex; Chau, Tom. Assessing the potential of electrodermal activity as an alternative access pathway. *Medical Engineering & Physics*, [*s. l.*], v. 30, n. 4, p. 498-505, 2008.

Brasil. Secretaria Especial dos Direitos Humanos. Comitê de Ajudas Técnicas. *Reunião do Comitê de Ajudas Técnicas 7 com a finalidade de discutir Tecnologia Assistiva voltada às pessoas com deficiência*. Brasília, df: Secretaria Especial dos Direitos Humanos, 2007.

Brouillette, Albert; Sarmah, Devraj; Kalita, Jugal. Multi-objective optimization for efficient brahmic keyboards. *In*: workshop on advances in text input methods, 2., 2012, Mumbai. *Proceedings* [...]. Mumbai: [*s. n.*], 2006. p. 29-44.

Caltenco, Hector A. *et al.* Understanding computer users with tetraplegia: survey of assistive technology users. *International Journal of Human-Computer Interaction*, [*s. l.*], v. 28, n. 4, p. 258-268, 2012.

Cardwell, Michael S. Locked-in syndrome. *Texas Medical*, Austin, v. 109, n. 2, 2013.

Ibge. *Censo Demográfico 2010*. Rio de Janeiro: ibge, 2010.

Chin, Seongah; Yeon lee, Chung; Lee, Jaedong. Facial expression image mapping for brain-computer interface using El type classification. *In*: international conference on information sciences and interaction sciences, 3., 2010, Chengdu. *Proceedings* [...]. Chengdu: [*s. n.*], 2010. p. 465-469.

Cipresso, Pietro *et al.* The combined use of brain computer interface and eye--tracking technology for cognitive assessment in amyotrophic lateral sclerosis. *In*: international conference on pervasive computing technologies for healthcare (pervasivehealth) and workshops, 5., 2011, Dublin. *Proceedings* [...]. Dublin: [*s. n.*], 2011. p. 320-324.

Clarkson, P. R.; Robinson, A. J. Language model adaptation using mixtures and an exponentially decaying cache. *In*: ieee international conference on acoustics, speech, and signal processing, 2., 1997, Munich. *Proceedings* [...]. Munich: ieee, 1997. p. 799-802.

Colas, Sonia *et al.* Artificial ants for the optimization of virtual keyboard arrangement for disabled people. *In*: monmarché, Nicolas *et al.* (ed.). *Artificial evolution*. Angers: Springer Berlin Heidelberg, 2008. p. 87-99.

Colker, Ruth. Americans with disabilities act: a windfall for defendants. *Harvard Civil Rights-civil Liberties Law Review*, Cambridge, v. 34, n. 1, 1999.

Cook, Albert M.; Polgar, Janice; Encarnacão, Pedro. *Assistive technologies*: principles and practice. Amsterdam: Elsevier Health Sciences, 2014.

Croes, Georges A. A method for solving traveling-salesman problems. *Operations Research*, [*s. l.*], v. 6, n. 6, p. 791-812, 1958.

Dawkins, Richard. *A escalada do monte improvável*: uma defesa da teoria da evolução. São Paulo: Companhia das Letras, 1996.

Deb, Kalyanmoy; Jain, Himanshu. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *ieee Transactions on Evolutionary Computation*, [*s. l.*], v. 18, n. 4, p. 577-601, 2014.

Deepa, V. Baby; Thangaraj, P.; Chitra, S. Investigating principal component analysis for classification of eeg data. *In*: international conference on networking and information technology, 2010, Manila. *Proceedings* [...]. Manila: [*s. n.*], 2010. p. 461-464.

De jong, Kenneth A. *Evolutionary computation*: a unified approach. Cambridge: mit Press, 2006.

Dicionário aberto. *Dicionário aberto*. Lisboa: Biblioteca Nacional de Portugal, 2015. Disponível em: https://dicionario-aberto.net/ Acesso em: 2 fev. 2022.

Dorigo, Marco; Gambardella, Luca M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *ieee Transactions on Evolutionary Computation*, [*s. l.*], v. 1, n. 1, p. 53-66, 1997.

Drewes, Heiko. Eye gaze tracking for human computer interaction. 2010. Dissertation – Ludwig-Maximilians-Universotät, München, 2010.

Eggers, Jan *et al.* Optimization of the keyboard arrangement problem using an ant colony algorithm. *European Journal of Operational Research*, Leeds, v. 148, n. 3, p. 672-686, 2003.

Engelbrecht, Andries P. *Computational intelligence*: an introduction. Hoboken: John Wiley & Sons, 2007.

Fazly, Afsaneh. *The use of syntax in word completion utilities*. 2002. Thesis (Master of Science) – University of Toronto, Toronto, 2002.

Felzer, Torsten; Rinderknecht, Stephan. 3Dscan: an environment control system supporting persons with severe motor impairments. *In*: international acm sigaccess conference on computers and accessibility, 11., 2009, Pittsburg. *Proceedings* [...]. Pittsburg: acm, 2013. p. 213-214.

Fitts, Paul M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, Washington, dc, v. 47, n. 6, p. 381-391, 1954.

Francis, Gregory; Johnson, Elizabette. Speed — accuracy tradeoffs in specialized keyboards. *Journal of Human Computer Studies*, [*s. l.*], v. 69, n. 7-8, p. 526-538, 2011.

Francis, Gregory; Oxtoby, Claire. Building and testing optimized keyboards for specific text entry. *Human Factors*: The Journal of the Human Factors and Ergonomics Society, [*s. l.*], v. 48, n. 2, p. 279-287, 2006.

Fu, Yu-Fen; Ho, Cheng-Seen. A fast text-based communication system for handicapped aphasiacs. *In*: international conference on intelligent information hiding and multimedia signal processing, 5., 2009, Kyoto. *Proceedings* [...]. Kyoto: ieee, 2009. p. 583-594.

Galvão filho, Teófilo A.; García. Jesus C. D. *Pesquisa nacional de tecnologia assistiva*. São Paulo: its brasil, 2012.

Garay-vitoria, Nestor; Abascal, Julio. Text prediction systems: a survey. *Universal Access in the Information Society*, [*s. l.*], v. 4, n. 3, p. 188-203, 2006.

Garcia doval, Fátima M.; Pousada carballo, José M.; Vez jeremias, J. M. tictac: Information and communication technologies for augmentative commu-

nication boards. *In*: IEEE ENGINEERING EDUCATION (EDUCON), 2010, Madrid. *Proceedings* [...]. Madrid: IEEE, 2010. p. 1783-1787.

GELORMINI, Derek; BISHOP, Benjamin. Optimizing the android virtual keyboard: a study of user experience. *In*: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO WORKSHOPS, 2013, San Jose. *Proceedings* [...]. San Jose: IEEE, 2013. p. 1-4.

GEYER-SCHULZ, Andreas. *Fuzzy rule-based expert systems and genetic machine learning*. Heidelberg: Physica Verlag, 1997.

GHOSH, Soumalya *et al.* Effective virtual keyboard design with size and space adaptation. *In*: IEEE STUDENTS TECHNOLOGY SYMPOSIUM (TECHSYM), 2010, Kharagpur. *Proceedings* [...]. Kharagpur: IEEE, 2010. p. 262-267.

GHOSH, Soumalya; SARCAR, Sayan; SAMANTA, Debasis. Designing an efficient virtual keyboard for text composition in Bengali. *In*: INTERNATIONAL CONFERENCE ON HUMAN COMPUTER INTERACTION, 3., 2011, Bangalore. *Proceedings* [...]. Bangalore: ACM, 2011. p. 84-87.

GIVEN, Lisa M. (ed.). *The Sage encyclopedia of qualitative research methods*. Thousand Oaks: Sage Publications, 2008.

GLOVER, Fred; KOCHENBERGER, Gary A. *Handbook of metaheuristics*. Berlin: Springer, 2003.

GOETTL, Jeffrey S.; BRUGH, Alexander W.; JULSTROM, Bryant A. Call me e-mail: arranging the keyboard with a permutation-coded genetic algorithm. *In*: ACM SYMPOSIUM ON APPLIED COMPUTING, 2005, Santa Fe. *Proceedings* [...]. Santa Fe: ACM, 2005. p. 947-951.

GOLDBERG, David E. *Genetic algorithms in search, optimization, and machine learning*. Boston: Addison-Wesley, 1989.

GOODMAN, Joshua *et al.* Language modeling for soft keyboards. *In*: INTERNATIONAL CONFERENCE ON INTELLIGENT USER INTERFACES, 7., 2002, San Francisco. *Proceedings* [...]. San Francisco: [*s. n.*], 2002.

GRANGE, Aristide. L'interface du clavier virtuel Chewing Word. *In*: CONFERENCE INTERNATIONALE FRANCOPHONE SUR L'INTERACTION HOMME-MACHINE, 22., 2010, Luxembourg. *Proceedings* [...]. Luxembourg: [*s. n.*], 2010. p. 237-240.

GROVER, Dale L.; KING, Martin T.; KUSHLER, Clifford A. *Reduced keyboard disambiguating computer*. [*S. l.*]: Google Patents, 1998.

GUERRIER, Yohan *et al.* Comparative study between AZERTY-type and K-Hermes virtual keyboards dedicated to users with cerebral palsy. *In*: STEPHANIDIS, Constantine (ed.). *Universal access in human-computer interaction*: intelligent and ubiquitous interaction environments. Berlin: Springer, 2011. p. 310-319.

HADKA, David. MOEA Framework, a Java library for multiobjective evolutionary

algorithms. *MOEA Framework*, [*s. l.*], 2015. Disponível em: http://moeaframework.org/. Acesso em: 9 fev. 2022.

Hanson, Elizabeth K. *et al.* The impact of alphabet supplementation and word prediction on sentence intelligiblity of electronically distorted speech. *Speech Communication*, [*s. l.*], v. 52, n. 2, p. 99-105, 2010.

Harbusch, Karin; Kühn, Michael. Towards an adaptive communication aid with text input from ambiguous keyboards. *In*: conference on european chapter of the association for computational linguistics, 10., 2003, Budapest. *Proceedings* [...]. Budapest: [*s. n.*], 2003. p. 207-210.

Herrero, Dafne; Monteiro, Carlos 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*, São Paulo, v. 18, n. 2, p. 163-169, 2008.

Hick, William E. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, [*s. l.*], v. 4, n. 1, p. 11-26, 1952.

Hill, Katya. Advances in augmentative and alternative communication as quality-of-life technology. *Physical Medicine and Rehabilitation Clinics of North America*, [*s. l.*], v. 21, n. 1, p. 43-58, 2010.

Holland, John H. *Adaptation in natural and artificial systems*: an introductory analysis with applications to biology, control, and artificial intelligence. Cambridge: mit Press, 1975.

Hoste, Lode; Signer, Beat. Speeg2: a speech- and gesture-based interface for efficient controller-free text input. *In*: international conference on multimodal interaction, 15., 2013, Sydney. *Proceedings* [...]. Sydney: [*s. n.*], 2013. p. 213-220.

Hyman, Ray. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology*, Washington, dc, v. 45, n. 3, p. 188-188, 1953.

Ibm. *ibm spss Software*: Brasil. Armonk: ibm, 2015.

Jain, Siddharth; Bhattacharya, Samit. Virtual keyboard layout optimization. *In*: students technology symposium (techsym), 2010, Kharagpur. *Proceedings* [...]. Kharagpur: ieee, 2010. p. 312-317.

Janpinijrut, Siwacha; Nattee, Cholwich; Kayasith, Prakasith. An approach for improving thai text entry on touch screen mobile phones based on distance and statistical language model. *In*: theeramunkong, Thanaruk *et al.* (ed.). *Knowledge, information, and creativity support systems*. Berlim: Springer, 2011. p. 44-55.

Johnson, Jeanne M. *et al.* Perspectives of speech language pathologists regarding success versus abandonment of aac. *Augmentative and Alternative Communication*, [*s. l.*], v. 22, n. 2, p. 85-99, 2006.

Joshi, Anirudha *et al.* Design and evaluation of Devanagari virtual keyboards for touch screen mobile phones. *In*: international conference on human computer interaction with mobile devices and services, 13., 2011, Munich. *Proceedings* [...]. Munich: [*s. n.*], 2011.

Keegan, Johnalan; Burke, Edward; Condron, James. An electrooculogram-based binary saccade sequence classification (bssc) technique for augmentative communication and control. *In*: annual international conference of the ieee engineering in medicine and biology society, 2009, Minneapolis. *Proceedings* [...]. Minneapolis: ieee, 2009. p. 2604-2607.

Kennedy, James. Particle swarm optimization. *In*: sammut, Claude; Webb, Geoffrey I. (ed.). *Encyclopedia of machine learning*. Boston: Springer, 2011. p. 760-766.

Kirkpatrick, Scott; Vecchi, M. P.; Gelatt, C. D. Optimization by simmulated annealing. *Science*, [*s. l.*], v. 220, n. 4598, p. 671-680, 1983.

Knuth, Donald E. Dynamic huffman coding. *Journal of Algorithms*, [*s. l.*], v. 6, n. 2, p. 163-180, 1985.

Knuth, Donald E. *The art of computer programming*: sorting and searching, v. 3. Boston: Addison Wesley, 1998.

Kraskowsky, Laura H.; Finlayson, Marcia. Factors affecting older adults use of adaptive equipment: review of the literature. *The American Journal of Occupational Therapy*, [*s. l.*], v. 55, n. 3, p. 303-310, 2001.

Krolak, Aleksandra; Strumillo, Pawel. Vision-based eye blink monitoring system for human-computer interfacing. *In*: conference on human system interactions, 2008, Krakow. *Proceedings* [...]. Krakow: [*s. n.*], 2008. p. 994-998.

Krug, Paulo D.; Piconand, Bárbara Corrêa; Amaral, Karine Medeiros. Esclerose lateral amiotrófica. *Protocolos Clínicos e Diretrizes Terapêuticas*, Brasília, df, p. 313-317, 2002.

Kübler, A. *et al.* bci meeting 2005-workshop on clinical issues and applications. *ieee Transactions on Neural Systems and Rehabilitation Engineering*, [*s. l.*], v. 14, n. 2, p. 131-134, 2006.

Kwon, Sunghyuk; Lee, Donghun; Chung, Min 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*, [*s. l.*], v. 39, n. 5, p. 888-893, 2009.

LADANY, Shaul P. A model for optimal design of keyboards. *Computers & Operations Research*, [*s. l.*], v. 2, n. 1, p. 55-59, 1975.

LESHER, Gregory W.; MOULTON, Bryan J. A method for optimizing single-finger keyboards. *In*: RESNA ANNUAL CONFERENCE: TECHNOLOGY FOR THE NEW MILLENNIUM, 2000, Orlando. *Proceedings* [...]. Orlando: RESNA, 2000. p. 91-93.

LESHER, Gregory; MOULTON, Bryan; HIGGINBOTHAM, D. Jeffery. Techniques for augmenting scanning communication. *Augmentative and Alternative Communication*, [*s. l.*], v. 14, n. 2, p. 81-101, 1998a.

LESHER, Gregory W.; MOULTON, Bryan J.; HIGGINBOTHAM, D. Jeffery. Optimal character arrangements for ambiguous keyboards. *IEEE Transactions on Rehabilitation Engineering*, [*s. l.*], v. 6, n. 4, p. 415-423, 1998b.

LEVINE, Stephen H.; TREPAGNIER, Cheryl. Customised text entry devices for motor-impaired users. *Applied Ergonomics*, [*s. l.*], v. 21, n. 1, p. 55-62, 1990.

LIGHT, Lissa; ANDERSON, Peter. *Designing better keyboards via simulated annealing*. [*S. l.*]: Miller Freeman, 1993.

LOJA, Luiz F. B. *et al.* A concept-environment for computer-based augmentative and alternative communication founded on a systematic review. *Research on Biomedical Engineering*, [*s. l.*], v. 31, n. 3, 2015.

LOPES, Maria C. S. *Mineração de dados textuais utilizando técnicas de clustering para o idioma português*. 2004. Tese (Doutorado em Engenharia) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2004.

MACKENZIE, I. Scott. KSPC (keystrokes per character) as a characteristic of text entry techniques. *In*: INTERNATIONAL SYMPOSIUM ON MOBILE HUMAN-COMPUTER INTERACTION, 4., 2002, Pisa. *Proceedings* [...]. Pisa: Springer-Verlag, 2002. p. 195-210.

MACKENZIE, I. Scott. Evaluation of text entry techniques. *In*: MACKENZIE, I. Scott; TANAKA-ISHII, Kumiko (ed.). *Text entry systems*: mobility, acessibility, universality. Burlington: Morgan Kaufmann, 2007. p. 75-101.

MACKENZIE, I. Scott *et al.* LetterWise: prefix-based disambiguation for mobile text input. *In*: UIST'01: ANNUAL ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 14., 2001, Orlando. *Proceedings* [...]. Orlando: ACM, 2001. p. 111-120.

MACKENZIE, I. Scott; ZHANG, Shawn X. The design and evaluation of a high-performance soft keyboard. *In*: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS , 1999, Boston. *Proceedings* [...]. Boston, 1999. p. 25-31

MACKENZIE, I. Scott; ZHANG, Shawn X.; SOUKOREFF, R. William. Text entry using soft keyboards. *Behaviour & Information Technology*, [*s. l.*], v. 18, n. 4, p. 235-244, 2010.

Majaranta, Päivi *et al.* Auditory and visual feedback during eye typing. *In*: human factors in computing systems, 2003, Fort Lauderdale. *Proceedings* [...]. Fort Lauderdale: acm, 2003. p. 766-767.

Mak, Joseph N.; Wolpaw, Jonathan R. Clinical applications of brain-computer interfaces: current state and future prospects. *ieee Reviews in Biomedical Engineering*, [*s. l.*], v. 2, p. 187-199, 2009.

Mason, C.; Chinn, K. M. Augmentative alternative communication. *International Encyclopedia of Education*, [*s. l.*], p. 544-549, 2010.

Massey, Frank J. The Kolmogorov-Smirnov Test for goodness of fit. *Journal of the American Statistical Association*, Alexandria, v. 46, n. 253, p. 68-78, 1951.

Mccoy, Kathleen F. *et al.* Speech and Language processing as assistive technologies. *Computer Speech and Language*, [*s. l.*], v. 27, n. 6, p. 1143-1146, 2013.

Mitchell, Melanie. *An introduction to genetic algorithms*. Cambridge: mit Press, 1999.

Merino, Manuel *et al.* Assessment of biosignals for managing a virtual keyboard. *In*: miesenberger, Klaus *et al.* (ed.). *Computers helping people with special needs*. Berlim: Springer, 2012. p. 331-337.

Metropolis, Nicholas *et al.* Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, [*s. l.*], v. 21, n. 6, p. 1087-1092, 1953.

Michaelis. *Dicionário português Online Home Page*. São Paulo: uol, 2015.

Michalewicz, Zbigniew. *Genetic algorithms* + Data *structures* = Evolution *programs*. Berlim: Springer Science, 1996.

Millet, Barbara; Asfour, Shihab; Lewis, James R. Selection-based virtual keyboard prototypes and data collection application. *Behavior Research Methods*, [*s. l.*], v. 41, n. 3, p. 951-956, 2009.

Miró-borrás, Julio; Bernabeu-soler, Pablo. Text entry in the e-commerce age: two proposals for the severely handicapped. *Journal of Theoretical and Applied Electronic Commerce Research*, Talca, v. 4, n. 1, p. 101-112, 2009.

Miró, Julio; Bernabeu, Pabloa. Text entry system based on a minimal scan matrix for severely physically handicapped people. *In*: miesenberger, Klaus *et al.* (ed.). *Computers helping people with special needs*. Berlin: Springer, 2008. p. 1216-1219.

Molina, Alberto J.; Rivera, Octavio; Gómez, Isabel. Measuring performance of virtual keyboards based on cyclic scanning. *In*: international conference on autonomic and autonomous systems, 5., Valencia. *Proceedings* [...]. Valencia: ieee, 2009. p. 174-178.

MOLINA, Alberto *et al.* Evaluation of unambiguous virtual keyboards with character prediction. *Assistive Technology Research Series*, Amsterdam, v. 25, 2009.

NICOLAU, Hugo *et al.* Mobile text-entry and visual demands: reusing and optimizing current solutions. *Universal Access in the Information Society*, Berlin, v. 13, n. 3, p. 291-301, 2013.

NORMAN, Donald A.; RUMELHART, David E. Studies of typing from the LNR Research Group. *In*: COOPER, William (ed.). *Cognitive aspects of skilled typewriting*. Berlin: Springer, 1983. p. 45-65.

NUANCE. Nuance home page. *Nuance*, [*s. l.*], 2015a.

NUANCE. XT9 Smart Input is a multi-modal text input solution that supports a variety of device form factors and input methods. *Nuance*, [*s.* l.], 2015b.

OKASAKA, Sho; HOSHINO, Yukinobu. Development of estimation method about activity states for NIRS-based BCI system. *In*: INTERNATIONAL CONFERENCE ON SOFT COMPUTING AND INTELLIGENT SYSTEMS, 6., 2012, Kobe. *Proceedings* [...]. Kobe: IEEE, 2012. p. 1144-1149.

OOMMEN, Basantkumar J.; VALIVETI, Radhakrishna S.; ZGIERSKI, Jack. A fast learning automaton solution to the keyboard optimization problem. *In*: INTERNATIONAL CONFERENCE ON INDUSTRIAL AND ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS, 3., Charleston, 1990. *Proceedings* [...]. Charleston: ACM, 1990. p. 981-990, v. 2.

ORENGO, Viviane; HUYCK, Christian. A stemming algorithmm for the Portuguese language. *In*: SYMPOSIUM ON STRING PROCESSING AND INFORMATION RETRIEVAL, 8., 2001, Laguna de San Rafael. *Proceedings* [...]. Laguna de San Rafael: [*s. n.*], 2001. p. 186-193.

ORHAN, Umut *et al.* RSVP keyboard: an EEG based typing interface. *In*: INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 2012, Kyoto. *Proceedings* [...]. Kyoto: IEEE, 2012. p. 645-648.

PANWAR, Prateek; SARCAR, Sayan; SAMANTA, Debasis. EyeBoard: a fast and accurate eye gaze-based text entry system. *In*: INTERNATIONAL CONFERENCE ON INTELLIGENT HUMAN COMPUTER INTERACTION, 4., 2012, Kharagpur. *Proceedings* [...]. Kharagpur: IEEE, 2012. p. 1-8.

PARDALOS, Panos M.; RENDL, Franz; WOLKOWICZ, Henry. The quadratic assignment problem: a survey and recent developments. *In*: PARDALOS, Panos; WOLKOWICZ, Henry. (ed.). *Quadratic Assignment and Related Problems, v. 16*. Providence: AMS, 1993. p. 1-42.

PARK, Si-Woon *et al.* Augmentative and alternative communication training using eye blink switch for Locked-in syndrome patient. *Annals of Rehabilitation Medicine*, [*s. l.*], v. 36, n. 2, p. 268-272, 2012.

Poláček, Ondrej; Míkovec, Zdenek; Slavík, Pavel. Predictive scanning keyboard operated by hissing. *In*: iasted international conference assistive technologies, 2., 2012, Innsbruck. *Proceedings* [...]. Innsbruck: iasted, 2012. p. 862-869.

Pouplin, Samuel *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*, [*s. l.*], v. 51, n. 3, p. 467-480, 2014.

Prabhu, Vijit; Prasad, Girijesh. Designing a virtual keyboard with multi-modal access for people with disabilities. *In*: world congress on information and communication technologies, 2011, Mumbai. *Proceedings* [...]. Mumbai: [*s. n.*], 2011. p. 1133-1138.

Przedborski, Serge; Vila, Miquel; Jackson-lewis, Vernice. Series introduction: neurodegeneration: what is it and where are we? *The Journal of Clinical Investigation*, [*s. l.*], v. 111, n. 1, p. 3-10, 2003.

Raynal, Mathieu; Vigouroux, Nadine. Genetic algorithm to generate optimized soft keyboard. *In*: extended abstracts on human factors in computing systems, 2005, Portland. *Proceedings* [...]. Portland: [*s. n.*], 2005. p. 1729-1732.

Sadler, Thomas W. *Langman embriología médica*: con orientación clínica. Buenos Aires: Editorial Médica Panamericana, 2007.

Sarcar, Sayan *et al.* Virtual keyboard design: state of the arts and research issues. In: ieee students' technology symposium (techsym), 2010, Kharagpur. *Proceedings* [...]. Kharagpur: ieee, 2010. p. 289-299.

Sardinha, Tony; Moreira filho, José; Alambert, Eli. 2014.

Sassaki, Romeu Kazumi. Inclusão: acessibilidade no lazer, trabalho e educação. *Revista Nacional de Reabilitação*, São Paulo, v. 10, n. 57, p. 8-16, 1996.

Schadle, Igor. Sibyl: aac system using nlp techniques. *In*: miesenberger, Klaus *et al.* (ed.). *Computers helping people with special needs*. Berlim: Springer, 2004. p. 1009-1015.

Schalk, Gerwin *et al.* Brain computer interfaces (bcis): detection instead of classification. *Journal of Neuroscience Methods*, [s. l.], v. 167, n. 1, p. 51-62, 2008.

Schwartzman, José S. Paralisia cerebral. *Arquivos Brasileiros de Paralisia Cerebral*, São Paulo, v. 1, n. 1, p. 4-17, 2004.

Shang, Heping; Merrettal, T. H. Tries for approximate string matching. *ieee Transactions on Knowledge and Data Engineering*, [*s. l.*], v. 8, n. 4, p. 540-547, 1996.

Shannon, Claude E. Prediction and entropy of printed English. *The Bell System Technical Journal*, [*s. l.*], v. 30, n. 1, p. 50-64, 1951.

Sharma, Manoj K. *et al.* Parameters effecting the predictive virtual keyboard. *In*: ieee students' technology symposium, 2010, Kharagpur. *Proceedings* [...]. Kharagpur: ieee, 2010. p. 268-275.

Sharma, Manoj K. *et al.* Visual clue: an approach to predict and highlight next character. *In*: international conference on intelligent human computer interaction, 4., 2012, Kharagpur. *Proceedings* [...]. Kharagpur: [*s. n.*], 2012. p. 1-7.

Sik-lányi, Cecilia; Molnár-lányi, Ágnes. Psychological and pedagogic testing of handicapped children with locomotion disorder using multimedia programs. *In*: international conference on disability, virtual reality and associated technologies, 3., Alghero, 2000. *Proceedings* [...]. Alghero: [*s. n.*], 2000. p. 99-106.

Silfverberg, Miika; Mackenzie, I. Scott; Korhonen, Panu. Predicting text entry speed on mobile phones. *In*: conference on human factors in computing systems, 2000, The Hague. *Proceedings* [...]. The Hague: [*s. n.*], 2000. p. 9-16.

Silva, Fernando; Pereira, Francisco. Communication between people with motion and speech disabilities. *In*: iberian conference on information systems and technologies, 6., 2011, Chaves. *Proceedings* [...]. Chaves: [*s. n.*], 2011. p. 1-4.

Simathamanand, Jirapat; Piromsopa, Krerk. Performance optimization of thai virtual keyboard for social networking. *In*: international joint conference on computer science and software engineering, 8., 2011, Nakhonpathom. *Proceedings* [...]. Nakhonpathom: [*s. n.*], 2011. p. 210-213.

Simpson, Richard. Making better decisions [modeling the assistive technology assessment]. *ieee Engineering in Medicine and Biology Magazine*, [*s. l.*], v. 27, n. 2, p. 23-28, 2008.

Smith, Eimear; Delargy, Mark. Locked-in syndrome. *bmj* : British Medical Journal, [*s. l.*], v. 330, n. 7488, 2005.

Sörensen, Kenneth. Multi-objective optimization of mobile phone keymaps for typing messages using a word list. *European Journal of Operational Research*, [*s. l.*], v. 179, n. 3, p. 838-846, 2007.

Sorger, Bettina *et al.* Another kind of bold Response: answering multiple-choice questions via online decoded single-trial brain signals. *Progress in Brain Research*, [*s. l.*], v. 177, p. 275-292, 2009.

Soukoreff, R. William; Mackenzie, I. Scott. Measuring errors in text entry tasks: an application of the levenshtein string distance statistic. *In*: extended abstracts on human factors in computing systems, 2001, Seattle. *Proceedings* [...]. Seattle: acm, 2001. p. 319-320.

Sun, Gufei; Hu, Jinglu; Wu, Gengfeng. A novel frequency band selection method for Common Spatial Pattern in Motor imagery based Brain Computer Interface. *In*: international joint conference on neural networks, 2010, Barcelona. *Proceedings* [...]. Barcelona: [*s. n.*], 2010. p. 1-6.

Tanaka-ishii, Kumiko; Inutsuka, Yusuke; Takeichi, Masato. Entering text with a four-button device. *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, v. 1, p. 1-7, 2002.

Thomas, Kavitha P. *et al.* An adaptive filter bank for motor imagery based Brain Computer Interface. *In*: annual international conference of the ieee engineering in medicine and biology society, 30., 2008, Vancouver. *Proceedings* [...]. Vancouver: ieee, 2008. p. 1104-1107.

Topal, Cihan; Benligiray, Burak; Akinlar, Cuneyt. On the efficiency issues of virtual keyboard design. *In*: ieee international conference on virtual environments human-computer interfaces and measurement systems, 2012, Tianjin. *Proceedings* [...]. Tianjin: ieee, 2012. p. 38-42.

Truong, Chau Thai *et al.* Collaborative smart virtual keyboard with word predicting function. *In*: kurosu, Masaaki. (ed.). *Human-computer-interaction*. Berlin: Springer, 2013. p. 513-522.

Usakli, Ali B. *et al.* A hybrid platform based on eog and eeg signals to restore communication for patients afflicted with progressive motor neuron diseases. *In*: annual international conference of the ieee engineering in medicine and biology society, 2009, Minneapolis. *Proceedings* [...]. Minneapolis: embc, 2009. p. 543-546.

Varcholik, Paul D.; Laviola, Joseph J.; Hughes, Charles E. Establishing a baseline for text entry for a multi-touch virtual keyboard. *International Journal of Human-Computer Studies*, [*s. l.*], v. 70, n. 10, p. 657-672, 2012.

Wandmacher, Tonio *et al.* Sibylle, an assistive communication system adapting to the context and its user. *acm Transactions on Accessible Computing*, [*s. l.*], v. 1, n. 1, p. 1-30, 2008.

Ward, David J.; Blackwell, Alan F.; Mackay, David J. C. Dasher — a data entry interface using continuous gestures and language models. *In*: annual acm symposium on user interface software and technology, 13., 2000, San Diego. *Proceedings* [...]. San Diego: acm, 2000. p. 129-137.

Wilkinson, Krista M.; Hennig, Shannon. The state of research and practice in augmentative and alternative communication for children with developmental/intellectual disabilities. *Mental Retardation and Developmental Disabilities Research Reviews*, [*s. l.*], v. 13, n. 1, p. 58-69, 2007.

WOBBROCK, Jacob O. Measures of text entry performance. *In*: MACKENZIE, I. Scott; TANAKA-ISHII, Kumiko (ed.). *Text entry systems*. Burlington: Morgan Kaufmann, 2007. p. 47-74.

WOHLIN, Claes *et al*. *Experimentation in software engineering*. Berlin: Springer, 2012.

WU, Fong-Gong; HUANG, Yu-Chun; WU, Meng-Long. New chording text entry methods combining physical and virtual buttons on a mobile phone. *Applied Ergonomics*, [*s*. *l*.], v. 45, n. 4, p. 825-832, 2014.

XPERT KEYBOARD. *XPERT Keyboarding*: Touch Typing Speeds without Training. 2015.

YAMADA, Hisao. A historical study of typewriters and typing methods, from the position of planning Japanese parallels. *Journal of Information Processing*, [*s*. *l*.], v. 2, n. 4, 1980.

YIN, Peng-Yeng; SU, En-Ping. Cyber Swarm optimization for general keyboard arrangement problem. *International Journal of Industrial Ergonomics*, [*s*. *l*.], v. 41, n. 1, p. 43-52, 2011.

ZHAI, Shumin; HUNTER, Michael; SMITH, Barton A. The metropolis keyboard — an exploration of quantitative techniques for virtual keyboard design. *In*: ANNUAL ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 13., 2000, San Diego. *Proceedings* [...]. San Diego: ACM, 2000. p. 119-128.

ZHAI, Shumin; KRISTENSSON, Per Ola. Introduction to shape writing. *In*: MACKENZIE, I. Scott; TANAKA-ISHII, Kumiko (ed.). *Text entry systems*: mobility, accessibility, universality. San Francisco: Morgan Kaufmann Publishers, 2007. p. 139-158.

ZHAI, Shumin; SMITH, Barton A. Alphabetically biased virtual keyboards are easier to use: layout does matter. *In*: HUMAN FACTORS IN COMPUTING SYSTEMS, 2001, Seattle. *Proceedings* [...]. Seattle, 2001. p. 321-322.

Créditos

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS

Créditos

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS

Reitora
Oneida Cristina Gomes Barcelos Irigon

Pró-Reitora de Pesquisa e Pós-Graduação
Lorena Pereira de Souza Rosa

Coordenadora da Editora
Vanderleida Rosa de Freitas e Queiroz

Projeto Gráfico
Pedro Henrique Pereira de Carvalho

Diagramação
Renata Rosa Franco

Revisão
Angelo Cuissi (Tikinet)
Olliver Robson Mariano Rosa
Vanderleida Rosa de Freitas e Queiroz

Formato 160 x 230mm

Tipografia Myriad Pro Bold 12/18 (títulos)
Chaparral Pro 12/18 (texto)

Imagem da Capa Freepik

Luiz Fernando Batista Loja é doutor em Engenharia Elétrica (2015) pela Universidade Federal de Uberlândia (UFU), mestre em Ciência da Computação (2011) pela Universidade Federal de Goiás (UFG), especialista em Gestão de Software (2009) pela Uni-Anhanguera, graduado em Ciência da Computação (2006) pela Pontifícia Universidade Católica de Goiás (PUC-GO). É professor do Instituto Federal de Educação, Ciência e Tecnologia de Goiás no Câmpus Anápolis. Possui experiência na área de Ciência da Computação, com foco em desenvolvimento de sistemas, tendo trabalhado principalmente em padrões e metodologias de desenvolvimento, modelagem e construção de arquiteturas para implementação de sistemas e tecnologia assistiva.

Nós só podemos ver um pouco do futuro,
mas o suficiente para perceber que há muito a fazer.
Alan Mathison Turing

A Tecnologia Assistiva tem se tornado uma das áreas de conhecimento mais abrangentes e promissoras no mundo contemporâneo. Embora a utilização de recursos tecnológicos remonte aos primórdios da história, a TA está em processo de construção. Com um enfoque multidisciplinar, essa área tem sido crucial na busca por soluções que possam melhorar a qualidade de vida de pessoas com deficiência. Por meio do uso de tecnologia, a TA tem transformado limitações em possibilidades, permitindo a comunicação, mobilidade e inclusão social daqueles que anteriormente enfrentavam obstáculos intransponíveis. É uma demonstração de como a tecnologia pode ser empregada para trazer dignidade e bem-estar a todos os seres humanos.

Luiz Fernando Batista Loja
PROFESSOR DO INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS NO CÂMPUS ANÁPOLIS

ISBN 978 85 67022 80 2
9 788567 022802

INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Goiás

editora ifg

ABEU
Associação Brasileira
das Editoras Universitárias