MARCOS LUCAS RIBEIRO RIOS

DESENVOLVIMENTO DE SUMARIZADOR AUTOMÁTICO PARA AUXÍLIO NA NAVEGAÇÃO DE USUÁRIOS CEGOS EM PÁGINAS WEB

LAVRAS – MG

2019

MARCOS LUCAS RIBEIRO RIOS

DESENVOLVIMENTO DE SUMARIZADOR AUTOMÁTICO PARA AUXÍLIO NA NAVEGAÇÃO DE USUÁRIOS CEGOS EM PÁGINAS WEB

Artigo de Graduação apresentado ao Departamento de Ciência da Computação para a obtenção do título de Bacharel em Ciência da Computação.

Prof. Dra. Paula Christina Figueira Cardoso
Orientadora

LAVRAS – MG

2019

MARCOS LUCAS RIBEIRO RIOS

DESENVOLVIMENTO DE SUMARIZADOR AUTOMÁTICO PARA AUXÍLIO NA NAVEGAÇÃO DE USUÁRIOS CEGOS EM PÁGINAS WEB

Artigo de Graduação apresentado ao Departamento de Ciência da Computação para a obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 26 de junho de 2019.

Prof. Dra. Paula Christina Figueira	UFLA
Prof. Dr. André Pimenta Freire	UFLA
Profa. Dra. Renata Teles Moreira	UFLA

Paula Christina Figueira
Profa. Paula Christina Figueira Cardoso DCC/UFLA

Prof. Dra. Paula Christina Figueira Cardoso
Orientadora

LAVRAS – MG
2019

RESUMO

Apesar de muitos avanços na área de acessibilidade Web, pessoas com deficiência visual têm dificuldade de acessar páginas na Web devido a má estrutura semântica das mesmas. Devido a diversos problemas enfrentados por usuários cegos, o presente artigo objetiva o desenvolvimento de um sumarizador automático para navegadores a fim de reduzir o tempo gasto de usuários cegos na Web e amenizar efeitos da sobrecarga de informação.

Palavras-chave: sumarização automática. modelagem de tarefas. acessibilidade Web. sobrecarga de informação. processamento de linguagem natural.

ABSTRACT

Despite many advancements in field of Web accessibility, people with visual problems have difficulty accessing the Web pages due to a semantic structure of the same. No problem faced by users is not easy, this article aims to develop an automatic algorithm for browsers and therefore the use of time spent on the Web and in terms of information.

Keywords: automatic summarization. task modeling. Web accessibility. information overload. natural language processing.

SUMÁRIO

Desenvolvimento de sumarizador automático para auxílio na navegação de usuários cegos em páginas Web

Marcos Lucas R. Rios[1], **Paula Christina F. Cardoso**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Lavras (UFLA)
Caixa Postal 3037 – Lavras – MG – Brazil

Abstract. *Despite many advancements in field of Web accessibility, people with visual problems have difficulty accessing the Web pages due to a semantic structure of the same. No problem faced by users is not easy, this article aims to develop an automatic algorithm for browsers and therefore the use of time spent on the Web and in terms of information.*

Resumo. *Apesar de muitos avanços na área de acessibilidade Web, pessoas com deficiência visual têm dificuldade de acessar páginas na Web devido a má estrutura semântica das mesmas. Devido a diversos problemas enfrentados por usuários cegos, o presente artigo objetiva o desenvolvimento de um sumarizador automático para navegadores a fim de reduzir o tempo gasto de usuários cegos na Web e amenizar efeitos da sobrecarga de informação.*

1. Introdução

A internet vem cada dia mais fazendo parte do nosso cotidiano. No Brasil, segundo o Instituto Brasileiro de Geografia e Estatística (IBGE) [IBGE 2018], de 2016 para 2017, o percentual de utilização da internet nos domicílios subiu de 69,3% para 74,9%, ou seja, três em cada quatro domicílios brasileiros possuem internet. Foi um salto de 5,6 pontos percentuais, em um ano. Na área urbana, esse percentual de utilização cresceu de 75,0% para 80,1% e na área rural, de 33,6% para 41,0%. Com a abrangência contínua da internet e seus avanços tecnológicos surgem-se vários desafios relacionados à acessibilidade e usabilidade.

No Brasil, segundo o Sistema IBGE de Recuperação Automática (SIDRA) [SIDRA 2013], banco de dados e estatísticas do IBGE, com dados do censo de 2013, há mais de 6 milhões de pessoas com deficiência visual. Estes utilizam de diversas ferramentas para acesso à conteúdos da internet, tais como, os leitores de tela.

Segundo [Teixeira 2015], leitores de tela são sistemas de software usados para obter resposta do computador por meio sonoro. O programa percorre textos e imagens e lê em voz alta tudo o que ele encontra na tela, assim como as operações que o usuário realiza com as teclas alfanuméricas e os comandos digitados. O usuário com deficiência visual usa das funcionalidades dos leitores de tela para se guiar e absorver o conteúdo da página.

Infelizmente os leitores não conseguem fazer seu papel corretamente quando se deparam com sites mal estruturados, seja pela falta de padrões de acessibilidade, semântica das *tags* do *HyperText Markup Language* (HTML) ou outros fatores. Nesses casos o usuário pode se perder guiando-se pelo leitor, não entender o conteúdo e talvez, ter reações de estresse.

Um exemplo de falta de padrões de acessibilidade em relação a imagens. Em muitos sites figuras não contém descrição e o leitor de tela pode descrevê-las de uma forma não muito amigável. Usando o leitor de tela ChromeVox, por exemplo, quando passamos por uma imagem clicável sem descrição é dito a seguinte frase ao usuário: "image, link".

No contexto de sites de cunho informativo, a pessoa com deficiência visual pode perder muito tempo em páginas com notícias que não lhe interessa devido o tempo para procurar informação e entender seu conteúdo.

Com a área de processamento de linguagem natural existem vários projetos que trabalham resumindo textos, extraindo suas palavras-chave e outras aplicações. Neste sentido, técnicas de PLN podem ser adaptadas para Web, visando extrair conteúdo relevante que possa ser lido para usuários cegos, e dessa forma reduzir a sobrecarga de informação que tais usuários sofrem.

O objetivo deste trabalho foi desenvolver um *plugin* para sumarizar o conteúdo das notícias de um site Web para reduzir a sobrecarga de informação de usuários cegos na Web.

Este trabalho está estruturado da seguinte forma. Após a esta introdução, a Seção 2 demonstra aspectos teóricos definindo acessibilidade Web; descreve conceitos gerais sobre diretrizes de acessibilidade; define de forma sucinta sobre DOM, modelagem de tarefas e KLM; sumarização automática e apresenta trabalhos que buscaram reduzir a sobrecarga de informação de usuários cegos na Web. A Seção 3 apresenta o desenvolvimento do trabalho. As Seções 4 e 5 contém respectivamente, testes, resultados finais e considerações finais.

2. Referêncial Teórico

2.1. Acessibilidade Web

Segundo [W3C 2014] acessibilidade na Web significa que pessoas com deficiência, idosos e pessoas em mudança devido ao envelhecimento podem usar a Web. Acessibilidade na Web busca inclusão, de forma que, qualquer pessoa consiga perceber, entender, navegar, interagir e contribuir para Web.

A W3C (*World Wide Werb Consourcing*) foi fundada em 1994 com objetivo de criar padrões na Web. Um pouco depois, em 1997, foi criada a WAI (Web Accessibility Initiative). A WAI faz parte da W3C e hoje, temos padrões que auxiliam desenvolvedores de páginas Web a desenvolverem seu conteúdo de forma acessível.

Desde seu surgimento, a internet está em constante evolução trazendo diversas funcionalidades e formas diferentes de se produzir conteúdo e, em paralelo, a abordagem a pessoas com deficiência que acessam a páginas Web ficou mais ampla. Segundo a cartilha de acessibilidade na Web da W3C [W3C 2014], para que haja acessibilidade na Web é necessário que vários componentes estejam trabalhando em conjunto, sendo esses componentes:

1. Conteúdo: informação contida na página ou aplicação Web que inclui:
    - a informação natural, tal como texto, imagem e áudio;
    - o código ou marcação, que define a estrutura, a forma de apresentação, etc.

2. Navegador: sofware para acesso à internet.
3. Tecnologia assistiva: usada por pessoas com deficiência e mobilidade reduzida, como é o caso dos programas leitores de tela, dos ampliadores de tela, dos teclados alternativos, entre outros.
4. O conhecimento do usuário, sua experiência e, em alguns casos, suas estratégias adaptativas para a utilização da Web.
5. Programadores, designers, codificadores, autores, entre outros, incluindo pessoas com deficiência que são desenvolvedores e usuários que contribuem com conteúdo.
6. Ferramentas de autoria (*authoring tools*): softwares usados para criar páginas Web.
7. Ferramentas de avaliação: avaliadores de acessibilidade, validadores de HTML, validadores de Cascading Style Sheets (CSS), entre outros.

2.2. Web Content Accessibility Guideline (WCAG)

Existem diretrizes que devem ser seguidas para auxílio na criação de sites acessíveis. Tais diretrizes estão na WCAG, atualmente na versão 2.1. Para os testes de acessibilidade contidas neste artigo foram utilizadas as diretrizes da WCAG 2.0.

Segundo [eMAG 2014], a WCAG 2.0 é estruturada em príncipios:

1. Perceptível: A informação e os componentes da interface do usuário têm de ser apresentados aos usuários em formas que eles possam perceber.
2. Operável: Os componentes de interface de usuário e a navegação têm de ser operáveis.
3. Compreensível: A informação e a operação da interface de usuário têm de ser compreensíveis.
4. Robusto: O conteúdo tem de ser robusto o suficiente para poder ser interpretado de forma concisa por diversos agentes do usuário, incluindo recursos de tecnologia assistiva.

Cada príncípio contém recomendações:

1. Perceptível:
   (a) Fornecer alternativas textuais para qualquer conteúdo não textual
   (b) Fornecer alternativas para multimídia
   (c) Criar conteúdo que possa ser apresentado de modos diferentes sem perder informação ou estrutura
   (d) Tornar mais fácil aos usuários a visualização e audição de conteúdos incluindo as separações das camadas da frente e de fundo
2. Operável:
   (a) Fazer com que todas as funcionalidades estejam disponíveis no teclado
   (b) Prover tempo suficiente para os usuários lerem e usarem o conteúdo
   (c) Não projetar conteúdo de uma forma conhecida por causar ataques epiléticos
   (d) Prover formas de ajudar os usuários a navegar, localizar conteúdos e determinar onde se encontram
3. Compreensível:
   (a) Tornar o conteúdo de texto legível e compreensível

(b) Fazer com que as páginas da Web apareçam e funcionem de modo previsível
(c) Ajudar os usuários a evitar e corrigir erros

4. Robusto:
   (a) Maximizar a compatibilidade entre os atuais e futuros agentes do usuário, incluindo os recursos de tecnologia assistiva

As recomendações possuem critérios de sucesso que são pontos especifícos a serem atingidos. Para cada critério de sucesso estão disponíveis técnicas específicas com exemplos de como o objetivo do critério pode ser atingido e testado.

Cada critério de sucesso é indicado por um nível de conformidade, que pode ser A, AA ou AAA:

- Nível A: barreiras mais significativas de acessibilidade. Estar em conformidade apenas com os critérios de nível A não garante um site altamente acessível;
- Nível AA: estar em conformidade com todos os critérios de sucesso de nível AA garante um site bastante acessível, ou seja, o site será acessível para a maioria dos usuários, sob a maior parte das circunstâncias e utilizando-se a maioria das tecnologias.
- Nível AAA: o nível de conformidade triplo A é bastante meticuloso, ou seja, visa garantir um nível otimizado de acessibilidade. A maioria dos critérios de sucesso de nível AAA refere-se a situações bastante específicas, normalmente objetivando refinar os critérios de sucesso de nível AA. Manter uma conformidade com certos critérios de sucesso de nível AAA pode ser um processo custoso e, às vezes, de difícil implementação. Portanto, muitos sites não possuem conteúdo que se aplica aos critérios de sucesso de nível AAA.

2.3. Análise de Leitores de Tela

Uma das ferramentas que auxiliam na acessibilidade Web é o leitor de tela. Ele faz parte de um conjunto de recursos da Tecnologia Assistiva. O Comitê de Ajudas Técnicas (CAT) [CAT 2009] define Tecnologia Assistiva da seguinte forma:

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

O primeiro leitor de tela foi criado pela IBM em 1984 para funcionar no sistema operacional DOS. Existem diversos leitores de telas gratuitos e comercializados. Segundo pesquisa realizada pela W3C em 2013, através de um questionário online [W3C 2013], os seis leitores de tela mais utilizados no país são, respectivamente, NVDA, JAWS, Dos Vox, Virtual Vision, Voice Over e Orca.

1. NVDA[1]: Desenvolvido pela Microsoft, é um software gratuito e de código aberto. Possui muitos atalhos no teclado para alterar suas configurações dispensando o acesso ao painel de controle. Pode ser usado na leitura de páginas Web e traz diversos comandos no teclado para esse propósito.

[1]NVDA - NonVisual Desktop Access - Disponível em http://www.nvaccess.com

2. JAWS[2]: Utilizado na plataforma Windows, permite ao deficiente visual um acesso quase completo ao sistema, desde criação de manipulação de pastas e arquivos até personalização do sistema operacional. É um software pago distribuído pela Freedon Scientific.
3. DosVox[3]: É um micro sistema computacional gratuito criado por um núcleo de desenvolvimento da Universidade Federal do Rio de Janeiro (UFRJ) que traz em si diversas funcionalidades como jogos, abertura de arquivos de áudio e executáveis, impressão de arquivos e outros.
4. Virtual Vision[4]: Criado pela empresa brasileira MicroPower, é um leitor de telas pago para plataforma Windows que traz fluidez na navegação do sistema com diversos atalhos para o usuário.
5. Voice Over[5]: Para MACs, contém mais de 30 vozes, incluindo uma brasileira chamada Raquel. Além de repassar a informação da página, o leitor de tela orienta a como utilizar opções do menu ou ativar um botão usando o trackpad ou teclado.
6. Orca[6]: É um leitor de tela disponível para plataformas linux. É um recurso nativo em algumas distribuições possuindo integrações com softwares como LibreOffice, Chrome e Firefox. Um ponto negativo é sua voz ser robotizada que compromete o entendimento do usuário em alguns casos.

2.4. Document Object Model (DOM)

De acordo com [Maldonado 2018], o DOM é um objeto que representa o documento HTML lido pelo navegador de forma estruturada definindo meios de acesso a sua estrutura.

O objeto criado pelo DOM tem o formato de uma árvore. É possível, acessando o DOM, transformar e obter informações através de linguagens como *Javascript*. Os métodos de acesso a informações pelo DOM podem ser desde acesso direto em *classes* ou *ids* de *tags* HTML.

Neste trabalho é feito acessos ao DOM para extração do texto da página Web para ser sumarizado pelo *plugin*.

2.5. Modelagem de tarefas e KLM

Segundo [PATERNO' 2001, tradução nossa], a modelagem de tarefas é um modelo que descreve precisamente as relações entre as várias tarefas. A modelagem de tarefas pode ser usada em projetos de aplicações e avaliações.

Existem diversas abordagens para modelagem de tarefas, sendo uma delas, o GOMS. O acrônimo GOMS significa, segundo seus autores [Card et al. 1983], *Goals* (metas), *Operators* (operações), *Methods* (métodos) e *Selection rules* (regras de seleção), blocos base para construção do modelo.

De acordo com [Schrepp e Hardt 2007], uma meta descreve o que o usuário deseja realizar. Um método é uma ação que usuário precisa fazer para atingir sua meta. Cada

[2]JAWS - Disponível em http://www.freedomscientific.com
[3]DosVox - Disponível em http://intervox.nce.ufrj.br/dosvox/
[4]Virtual Vision - Disponível em alvision.com.br
[5]Voice Over - Disponível em https://www.apple.com/br/accessibility/iphone/vision/
[6]Orca - Disponível em https://help.gnome.org/users/orca/stable/index.html

método é descrito por uma sequência de operações. Se em algum momento houver mais de um método possível para atingir um objetivo, são usadas regras de seleção para decidir qual dos métodos será utilizado.

Neste trabalho foi utilizada a modelagem de tarefas KLM (*Keystroke-Level Model*) [Card et al. 1980] em conjunto com GOMS. O KLM 'é uma versão mais simplificada do GOMS que não possui metas, métodos ou até mesmo regras de seleção. O KLM usa apenas as teclas pressionadas, os movimentos do mouse e as teclas do mouse, como forma de avaliar e analisar uma tarefa' [Silva et al. 2017, tradução nossa].

2.6. Sumarização automática

Desde a revolução da imprensa no século XV, com a criação da máquina de impressão tipográfica do alemão Johann Gutenberg até hoje, na denominada Era Digital, somos envoltos cada vez mais de informação. Isso gera uma oportunidade de maior obtenção de conhecimento, visto que a informação agora, é de fácil e rápido acesso. Todavia, da mesma forma que os avanços na distribuição de informação trouxeram benefícios, também trouxeram problemas como a sobrecarga de informação.

A sobrecarga de informação (*Information Overload*), ou Síndrome de Fadiga por Informação (IFS) é definida em diversos contextos porém, basicamente caracteriza-se pelo excesso de informação ao nível em que não há como processá-la adequadamente, prejudicando assim, tomadas de decisão e a obtenção do conhecimento. Nos primeiros estudos e definições sobre o tema foi elaborado um gráfico em formato de parábola mostrado na Figura 1.

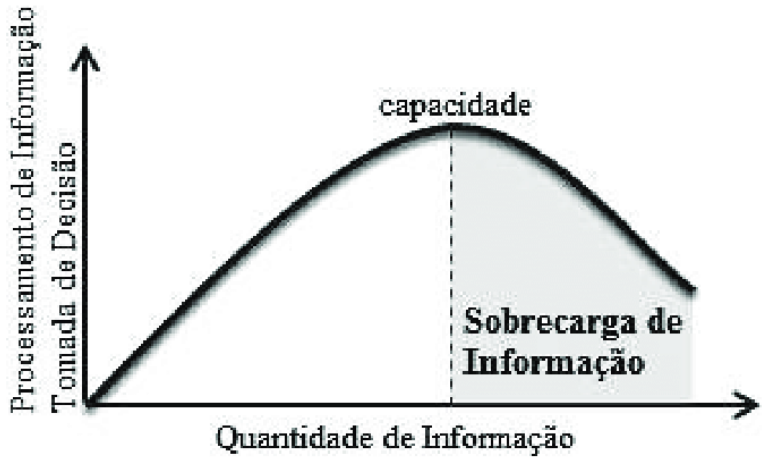


Figura 1. Representação da sobrecarga de informação em U invertido
[Marques 2016]

Como se pode observar, quanto maior a quantidade de informação, maior o processamento de informação e tomada de decisão porém, quando a quantidade de informação ultrapassa a nossa capacidade de processá-la, começamos a sofrer sobrecarga de informação e, por conta disto, nossa tomada de decisão e processamento das próximas informações são prejudicados.

Existem várias técnicas de leitura que visam buscar informação de forma rápida, evitando a necessidade de leitura completa de um texto sem afetar sua compreensão. Algumas técnicas de leitura são denominadas *Skimming*, *Scanning* e *Selectivity*. Tais técnicas auxiliam na diminuição de sobrecarga de informação.

Segundo [Roberto 2017], o *Skimming*, também conhecido como 'leitura por cima', diz que devemos fazer uma leitura rápida para se ter uma ideia geral do texto. É uma estratégia que exige observação das informações visuais que acompanham o texto (título, subtítulo, autor, fonte, data, distribuição gráfica do texto, tabelas, fotografias). O *Scanning* é uma estratégia que permite ao leitor localizar informações específicas no texto. É uma leitura rápida em que os olhos do leitor percorrem o texto para selecionar as ideias que realmente lhe interessam ou que são mais enfatizadas. O *Selectivity*, também chamada de 'leitura seletiva' é a técnica na qual selecionam-se os trechos onde se deseja encontrar uma determinada informação.

Quando realizamos alguma técnica de leitura inconscientemente é feita uma sumarização do conteúdo. Segundo [Rino e Pardo 2003], a sumarização humana pode ser definida como "a tarefa de redução do tamanho de um texto-fonte, com preservação de seu conteúdo mais relevante". Buscando facilitar a vida das pessoas e partindo de pressupostos como o de que cada ser humano sumariza um conteúdo de forma diferente, a sumarização automática (SA) ganha espaço.

Segundo [Ahmed et al. 2012] a sumarização automática é o resultado de algoritmos que condensam textos sem perder a essência. Tem como abordagens a sumarização por extração e por abstração. A sumarização extrativa, mais próxima do *skimming*, visa extrair sentenças e frases mais importantes do texto. Já a sumarização por abstração busca comprimir sentenças, parafraseando o conteúdo e gerando um novo modelo textual divergente do original, mas passando ao leitor a mesma informação.

2.7. Trabalhos que buscaram reduzir a sobrecarga de informação de usuários cegos na Web

Existem diversos trabalhos que auxiliam na redução de sobrecarga de informação de usuários cegos na Web como BraileSum [Wan e Hu 2015], CSurf [Mahmud et al. 2007] e o trabalho de Alves [Alves 2017].

BraileSum é um sumarizador para pessoas com deficiência visual. Seu método de sumarização é baseado numa programação linear de inteiros (ILP) gerando o resultado da sumarização em Braille. O conceito do método de sumarização com ILP é introduzido por [Gillick e Favre 2009], onde o objetivo do método é maximizar a soma dos bigramas. O método ILP é muito poderoso para sumarização extrativa porque pode selecionar frases importantes e remover a redundância ao mesmo tempo. O BrailleSum é apenas uma proposta de sistema, não estando disponível para uso.

CSurf é um leitor de telas que contém uma funcionalidade de extrair informações importantes de uma página Web através de processamento de linguagem natural. CSurf usa o conceito de contextos em sua sumarização. Os contextos consistem em, dada a árvore de dados de uma página Web, um conjunto de nós irmãos que contém o texto de um links importantes da página. O leitor sumariza textos em lingua inglesa e não está disponível para uso.

[Alves 2017] desenvolveu um sumarizador automático para Web que trabalha com a lingua portuguesa. Para validar o protótipo do sumarizador, modelou tarefas com GOMS e, em seguida, efetuou testes com usuários. A sumarização se dá por meio do método do caminho denso onde as sentenças mais importantes do texto são aquelas que possuem maior sobreposição de vocabulário com relação às outras [Salton et al. 1997]. O sumarizador aparenta não está disponível.

3. Metodologia

Para uma primeira versão do *plugin*, foi escolhida uma página Web para enfoque do trabalho. Por conseguinte, foi realizada uma análise de acessibilidade da página escolhida a fim de entender sua estrutura e trabalhar em cima de alguns problemas de acessibilidade da página com o *plugin*. Páginas com muitos erros de acessibilidade estruturais traria dificuldade de se encontrar o sumário na página através de leitores de tela.

3.1. Escolha e análise de acessibilidade de uma página Web

A escolha de uma página Web para verificação de acessibilidade e enfoque do trabalho se baseou em dois fatores: quantidade de acessos no Brasil e propósito do site.

Segundo Alexa [Alexa 2019], assistente virtual da Amazon, os cinco sites mais acessados no Brasil até junho de 2019 foram:

1. Google.com
2. Youtube.com
3. Facebook.com
4. Google.com.br
5. Globo.com

Para trabalharmos a sumarização automática foi buscado sites de notìcias. No Brasil, o site de nóticias mais acessado é o Globo.com tendo, como concorrente próximo, Uol.com.br na sétima colocação geral.

Para testes de acessibilidade no site G1 da globo.com foi efetuado um acesso à uma página de notícia do site e verificado se sua estrutura condiz com o apresentado nas diretrizes da WCAG 2.0. Essa análise foi feita com a junção e comparação do resultado de duas ferramentas de análise de acessibilidade, AcessMonitor[7] e TAW[8]. Como ambos são ferramentas automáticas, tem como limitação não atenderem todos os critérios da WCAG.

O AcessMonitor é um validador de acessibilidade de sites online que verifica os padrões WCAG 1.0 e 2.0. Foi produzido pela FTC (Fundação para Ciência e Tecnologia), uma organização do Ministério da Ciência, Tecnologia e Ensino Superior de Portugal.

TAW [TAW 2019] é uma ferramenta de análise de acessibilidade em sites. É referência no mundo hispânico e verifica os padrões da WCAG 2.0.

O AcessMonitor apresentou resultado para a página com um índice de 5.1, de uma escala de 1 a 10, que representa o nível de acessibilidade alcançado. Além disso, considerou que a página não passa da bateria de testes do nível A. Sucintamente, os erros encontrados foram os seguintes:

[7]AcessMonitor - Disponível em http://www.acessibilidade.gov.pt/accessmonitor/
[8]TAW - Disponível em https://www.tawdis.net

- Falta de legenda em imagens (atributo *alt* da *tag img* do HTML): Foram encontradas 7 de 15 imagens sem legenda.
- Marcação de *Links*, menus e texto dos *links*: Foram encontrados 6 links em que o conteúdo é composto apenas por uma imagem não legendada. Além disso foram identificados 3 casos em que o atributo *title* da *tag* a se limita a repetir o texto existente no link.
- Links para contornar blocos de informação: Constatou-se que o primeiro link da página não nos conduz até o conteúdo principal.
- Standards W3C ((X)HTML + CSS): Foram encontrados 203 valores repetidos do atributo *id*.
- Metadados (título, navegação, redireccionamento, reinicialização): A página contém 3 *tags title*.

A propriedade *alt* é extremamente importante para representar em forma falada o o conteúdo de uma imagem. Caso o autor do site não queira destacar o atributo *alt* de uma imagem e não deseje que o usuário cego navegue na imagem através de alguma tecnologia assistiva, ele deve usar o atributo *aria-hidden* na *tag image*. A *tag title* deve ser utilizada somente uma vez por página. O atributo *title* é utilizado para proporcionar informação complementar à existente no texto do *link*, não sendo uma cópia do mesmo.

Usando o TAW foram identificados 402 erros e 581 advertências. A ferramenta agrupa seus resultados de acordo com os príncipios e recomendações da WCAG. Segundo TAW, quanto aos príncipios da WCAG, há 7 erros e 67 advertências relacionados ao princípio perceptível, 27 erros e 31 advertências ao operável, 2 erros e 13 advertências ao compreensível e 366 erros e 470 advertências ao robusto.

3.2. Desenvolvimento do *plugin* para sumarizar textos na Web a fim de diminuir a sobrecarga de informação

Para criação do *plugin* foi usada a linguagem de programação *Javascript*. *Plugins* de navegadores utilizam por padrão, HTML e CSS para criação de interface e *Javascript* para manipulações no DOM da página e execução de rotinas.

A primeira versão do *plugin* foi criada para o Google Chrome devido ao fato de ser, segundo dados de statCounter [statCounter 2019] e W3Counter [LLC 2019], o navegador mais utilizado no Brasil.

Existem muitas bibliotecas na linguagem Python para processamento de linguagem natural. Para utilizá-las em *plugin*, seria necessário a criação do código Python e, logo após, converter para *Javascript* através de um *parser*. Outra opção seria usar um serviço em *NodeJS*, um interpretador *Javascript*, em *background* que rodaria código Python.

Devido à possibilidade do código gerado pelo parser não ser otimizado e o uso do NodeJS impactar em performance, foi necessária a codificação em *Javascript* de funções do *plugin* que estão presentes em bibliotecas python como spaCy[9] e NLTK[10].

O sumarizador desenvolvido trabalha em *background* do navegador com um *listener* que dispara um *script* de sumarização quando a página carregada pelo navegador é

[9]spaCy - Disponível em https://spacy.io/
[10]NLTK - Disponível em https://www.nltk.org/

uma notícia do G1. Tal configuração é feita no arquivo *manifest.json* do *plugin* onde é definido em quais *URLs* o *script* deve ser executado.

Podemos representar o processo de sumarização do *script* do *plugin* com os seguintes passos:

1. Extração do conteúdo a ser sumarizado
2. Remoção de stopwords do texto
3. Remoção de sufixos das palavras
4. Extração das sentenças do texto que contém a maior frequência da soma de palavras
5. Inserção das sentenças resultantes da extração na página Web

No primeiro passo, é extraido no DOM da página, através de *ids* específicas, o título, subtítulo (caso tenha) e conteúdo da nóticia. O conteúdo da notícia é um conjunto de *tags p* que são concatenados afim de gerar uma variável com todo o texto.

Logo após, é feito o segundo e terceiro passo, onde são removidas as *stopwords* e os sufixos das palavras do conteúdo extraído no passo um. Para remoção dos sufixos é utilizado o algoritmo de stemmer Orengo [Orengo e Huyck 2001], o qual tem seu fluxograma apresentado na Figura 2.

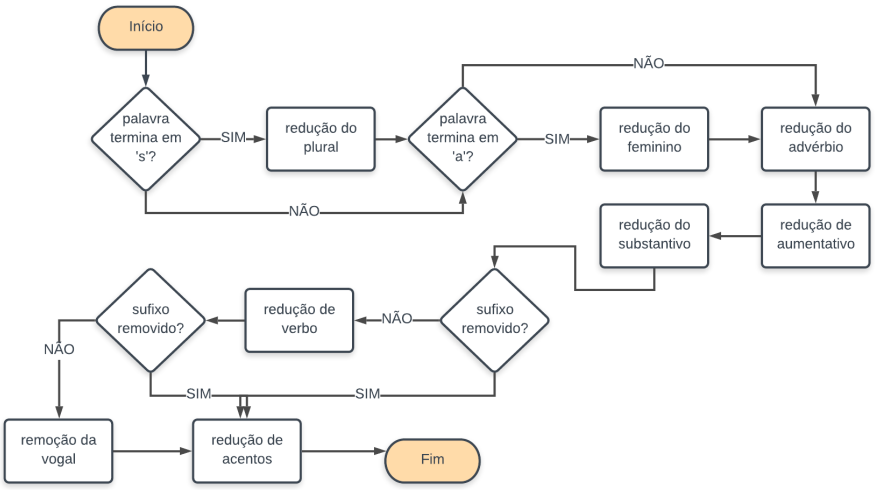


Figura 2. Fluxograma do algoritmo de Orengo - adaptado de [Orengo e Huyck 2001]

Cada passo do algoritmo de Orengo segue um conjunto de regras da lingua portuguesa.

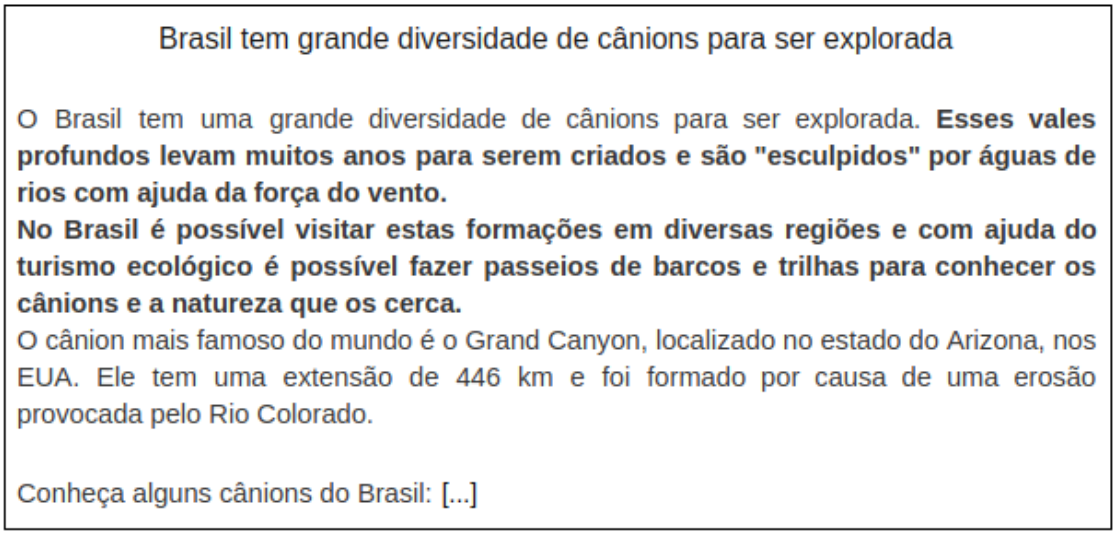
Após remoção de stopwords e sufixos do conteúdo da nóticia, é iniciado o passo quatro onde são selecionadas as frases mais relevantes partindo de uma abordagem de soma de frequência de palavras. O conteúdo manipulado até este passo, pode ser intitulado como representação interna do conteúdo. De forma geral, a abordagem do passo 4 leva os seguintes passos:

1. Gerar ranking de palavras da representação interna do conteúdo. A pontuação de cada palavra é dada de acordo com a quantidade de vezes que ela apareceu no texto.

2. Segmentar o conteúdo original e a representação interna em sentenças.
3. Gerar um ranking de sentenças da representação interna. A pontuação de cada sentença é a soma da quantidade de vezes que cada palavra que pertence à sentença foi usada. Para a soma, é necessário os valores do ranking de palavras gerado no passo 1. Por exemplo: suponha três palavras quaisquer representadas por a, b e c. Se a, b e c ocorreram no texto 10, 20 e 30 vezes respectivamente, a sentença 'a b' teria pontuação 30 e a sentença 'b c' teria pontuação 50.
4. Extrair a posição no texto das n primeiras sentenças do ranking de sentenças da representação interna. O valor n representa o número de frases que o sumarizador retornará.
5. Juntar as sentenças do conteúdo original nas posições encontradas no passo anterior para retornar ao usuário. Nesse passo já teremos o texto sumarizado.

Neste trabalho, foi decidido de forma empírica o retorno de 2 sentenças no sumário. Por fim, o texto sumarizado é inserido no DOM da página para visualização do usuário, completando o quinto e último passo.

Na Figura 3, exemplifica-se um texto fonte e, em negrito, são identificadas as sentenças selecionadas pelo algoritmo utilizado neste trabalho.


Brasil tem grande diversidade de cânions para ser explorada

O Brasil tem uma grande diversidade de cânions para ser explorada. **Esses vales profundos levam muitos anos para serem criados e são "esculpidos" por águas de rios com ajuda da força do vento.**
No Brasil é possível visitar estas formações em diversas regiões e com ajuda do turismo ecológico é possível fazer passeios de barcos e trilhas para conhecer os cânions e a natureza que os cerca.
O cânion mais famoso do mundo é o Grand Canyon, localizado no estado do Arizona, nos EUA. Ele tem uma extensão de 446 km e foi formado por causa de uma erosão provocada pelo Rio Colorado.

Conheça alguns cânions do Brasil: [...]

Figura 3. Exemplificação de resultado do sumarizador em negrito

4. Testes e Resultados Finais

Neste trabalho, o GOMS foi utilizado para criação de objetivos e ações de usuários para completar tarefas sobre as quais os testes se basearam. As tarefas foram executadas com o leitor de tela ChromeVox. As tarefas propostas foram:

1. Encontrar o resumo passando anteriormente pelo título e subtítulo (caso exista) da notícia;
2. Encontrar o resumo passando anteriormente pelo título e subtítulo (caso exista) da notícia e ouví-lo.

Para a primeira tarefa, usuários deveriam encontrar o resumo na página usando a navegação com setas para baixo do leitor de tela. É necessário que o usuário passe pelo título e subtítulo da notícia (caso exista) antes de alcançar o resumo.

A segunda tarefa é um complemento da primeira. Nesta tarefa, além de encontrar o resumo na página, os usuários deveriam ouvì-lo.

Para análise das tarefas com KLM, foi considerado os operadores K (*Keys*) e M (*mental operations*). Considerando o foco a usuários cegos, apenas cliques no teclado foram usados como entradas para realização as tarefas definidas, representado pelo operador K. O operador M foi escolhido devido a consideração de possíveis problemas de acessibilidade em páginas Web que façam o usuário cego pensar antes de mover-se para outro local do site.

A tabela 1 demonstra os resultados obtidos em cacda tarefa utilizando KLM para o leitor de tela ChromeVox. Além disso, o modelo GOMS para as tarefas são demonstrados nas Figuras 4 e 5.

Tarefa	K	M	Tempo
1	10	9	51s
2	10	8	87s

Tabela 1. Tempo estimado para tarefas usando ChromeVox

```
GOAL: Encontrar o resumo passando anteriormente pelo título e subtítulo (caso exista) da notícia:
    Pressionar a tecla modificadora do ChromeVox + seta para baixo 10K + 9M

Tarefa 1: 10K + 9M
```

Figura 4. Modelo GOMS para tarefa 1 no leitor de tela ChromeVox

```
GOAL: Encontrar o resumo passando anteriormente pelo título e subtítulo (caso exista) da notícia e ouví-lo:
    GOAL: Encontrar o resumo passando anteriormente pelo título e subtítulo (caso exista) da notícia:
        Pressionar a tecla modificadora do ChromeVox + seta para baixo 10K + 8M
    GOAL: Ouvir a notícia
        Ouvir a notícia 0K + 0M

Tarefa 2: 10K + 8M
```

Figura 5. Modelo GOMS para tarefa 2 no ChromeVox

Com os dados das tarefas demonstrados na Tabela 1, podemos observar que a estimativa de tempo para encontrar o resumo na página foi menos de um minuto. Considerando o tempo para ouvir o conteúdo do resumo, temos uma estimativa de 87 segundos, ou seja, menos de 2 minutos. O tempo de ouvir o resumo vária de notícia para notícia devido não haver um limite de quantidade de palavras definido para o resumo resultante do plugin.

Na Figura 6 mostra-se os locais, demonstrado em quadrados amarelos, onde o leitor de tela ChomeVox passou para alcançar o resumo.

Analisando os locais que o leitor de tela passou, foi observado que 5 das ações de navegação pelo leitor feitas para navegar ao resumo passavam por ícones de redes sociais

Figura 6. Locais que os leitores passaram durante a tarefa

presentes ao lado da data e horário da notícia. Esses ícones não estavam com nenhum tratamento de acessibilidade fazendo com que o leitor de tela falasse ’botão’ ao passar por eles.

Visando diminuir o tempo e os esforços mentais que o usuário cego enfrenta até chegar ao resumo foi adicionado na *tag* dos ícones de redes sociais o atributo *aria-hidden=true*. O leitor de tela não passa por *tags* com esse atributo.

Depois dessa melhoria no *script* foi feita uma segunda bateria de testes. A Tabela 2 demostra os resultados obtidos utilizando KLM para o leitor de tela ChromeVox. Além disso, o modelo GOMS para as tarefas dessa bateria de testes estão presentes nas Figuras 7 e 8.

Tarefa	K	M	Tempo
1	5	6	43s
2	5	5	76s

Tabela 2. Tempo estimado para tarefas usando ChromeVox após melhoria de *script*

```
GOAL: Encontrar o resumo passando anteriormente pelo título e subtítulo (caso exista) da notícia:
    Pressionar a tecla modificadora do ChromeVox + seta para baixo 5k + 6M

Tarefa 1: 5k + 6M
```

Figura 7. Modelo GOMS para tarefa 1 no leitor de tela ChromeVox

```
GOAL: Encontrar o resumo passando anteriormente pelo título e subtítulo (caso exista) da notícia e ouví-lo:
    GOAL: Encontrar o resumo passando anteriormente pelo título e subtítulo (caso exista) da notícia:
        Pressionar a tecla modificadora do ChromeVox + seta para baixo 5k + 5M
    GOAL: Ouvir a notícia
        Ouvir a notícia 0K + 0M

Tarefa 2: 5k + 5M
```

Figura 8. Modelo GOMS para tarefa 2 no leitor de tela ChromeVox

Comparando os dados das tarefas da Tabela 2 com a Tabela 1 vimos que foi obtida uma melhoria na estimativa do tempo de resolução para a tarefa 1 de 8 segundos. Já para a segunda tarefa, foi obtida uma melhoria de 11 segundos.

De forma geral, a melhoria da estima de tempo entre as baterias de teste foi ìnfima e, com base nos dados dos testes, podemos inferir que, caso estejamos em uma página de notícia que não nos interessa, o tempo estimado de acesso na página é menos de 2 minutos. Considerando a não existência de um resumo, o tempo de acesso estimado pode ser bem maior devido a necessidade do usuário de buscar mais informações para entender o conteúdo da notícia da página. Para isto, o número de operações mentais e teclas pressionadas para navegação pelo leitor de tela tendem a aumentar.

Utilizando o leitor ChromeVox, em alguns momentos, há uma repetição do conteúdo ou quebra da fala inusitadamente. Este fato é bem claro quando passamos com os leitores pela data, hora e tempo de atualização da notícia. O ChromeVox considera a data e hora separadas do texto de atualização da noticia. Essa separação faz com que o leitor tenha dois fluxos no teclado para ler data, hora e tempo de atualização. Além disso, quando passamos pelo tempo de atualização o leitor não fala corretamente o conteúdo, falando ao usuário, em alguns momentos, ”Atualizado tempo”.

5. Considerações finais

Com os resultados obtidos neste trabalho, foi observado que existem sites que não são acessíveis e que podem gerar estresse a usuários cegos, seja através de sobrecarga de informação ou outros fatores. Além disso, mostra-se necessário a produção e melhoria de ferramentas que objetivam o auxílio a comunidade de pessoas cegas na internet.

Neste trabalho foi desenvolvido um plugin que sumariza o conteúdo de notícias do site G1 a fim de diminuir a sobrecarga de informação de usuários cegos. Espera-se que a comunidade cega seja beneficiada ao navegar no site de notícias do G1 e em futuros sites que forem inseridos ao *plugin*.

Como trabalhos futuros, pretende-se o aprimoramento do *plugin* a fim de expandir as páginas Web a qual sumariza. Para tal, é necessário estudar as estruturas de outras páginas e fazê-las visíveis ao sumarizador produzido. Além disso, para avançar no trabalho usando modelos como GOMS e KLM, devem ser feitos mais estudos para explorar valores para tempos das operações, além da criação de novas tarefas, dada a inserção de novas páginas. É recomendável testes com usuários cegos para validações mais precisas do *plugin*. Quanto ao KLM, é interessante em futuros testes adicionar o operador wait (W) para considerar o tempo de espera para a sintese de voz de cada palavra pelo leitor de tela.

Além disso, é interessante testar novos métodos de sumarização a fim de que o sumarizador gere um resumo de mais qualidade. Para isso, pode ser feita a abordagem de Ahmed [Ahmed et al. 2012], a qual, são elaboradas perguntas sobre o conteúdo de um texto para os usuários que tem em mãos apenas o resumo gerado pelo sumarizador.

Pretende-se também, a adaptação do *plugin* para outros navegadores como Firefox, Opera, Safari e Edge.

Referências

Ahmed, F., Borodin, Y., Soviak, A., Islam, M., Ramakrishnan, I., e Hedgpeth, T. (2012). Accessible skimming: Faster screen reading of web pages. *UIST'12 - Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, pages 367–378.

Alexa (2019). Top sites in brazil. https://www.alexa.com/topsites/countries/BR. Acessado em 09/06/2019.

Alves, E. A. (2017). Sumarização automática para redução da sobrecarga de informações para usuários cegos na web.

Card, S., P. Moran, T., e Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Commun. ACM*, 23:396–410.

Card, S. K., Newell, A., e Moran, T. P. (1983). *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

CAT (2009). Tecnologia assistiva. https://www.pessoacomdeficiencia.gov.br/app/sites/default/files/publicacoes/livro-tecnologia-assistiva.pdf. Acessado em 13/06/2019.

eMAG (2014). Recomendações de acessibilidade wcag 2.0. http://emag.governoeletronico.gov.br/cursoconteudista/desenvolvimento-web/recomendacoes-de-acessibilidade-wcag2.html. Acessado em 09/06/2019.

Gillick, D. e Favre, B. (2009). A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18. Association for Computational Linguistics.

IBGE (2018). Pnad contínua tic 2017: Internet chega a três em cada quatro domicílios do país. https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/23445-pnad-continua-tic-2017-internet-chega-a-tres-em-cada-quatro-domicilios-do-pais. Acessado em 09/06/2019.

LLC, A. W. S. (2019). Browser platform market share - may 2019. https://www.w3counter.com/globalstats.php. Acessado em 14/06/2019.

Mahmud, J., Borodin, Y., e Ramakrishnan, I. (2007). Csurf: A context-driven non-visual web-browser. *16th International World Wide Web Conference, WWW2007*, pages 31–40.

Maldonado, L. (2018). Entendendo o dom (document object model). https://tableless.com.br/entendendo-o-dom-document-object-model/. Acessado em 2019-06-09.

Marques, R. P. (2016). *Sobrecarga de Informação na Era Digital: Causa ou Consequência?*, pages 19–28.

Orengo, V. M. e Huyck, C. (2001). A stemming algorithm for the portuguese language. In *Proceedings Eighth Symposium on String Processing and Information Retrieval*, pages 186–193. IEEE.

PATERNO', F. (2001). Task models in interactive software systems. In *Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals*, pages 817–836. World Scientific.

Rino, L. H. M. e Pardo, T. A. S. (2003). A sumarização automática de textos: principais características e metodologias. In *Anais do XXIII Congresso da Sociedade Brasileira de Computação*, volume 8, pages 203–245.

Roberto (2017). Você sabe aplicar as técnicas ”skimming”, ”scanning”e ”selectivity”nos textos? https://profes.com.br/robertomedeiros/blog/voce-sabe-aplicar-as-tecnicas-skimming-scanning-e-selectivity-nos-textos. Acessado em 13/06/2019.

Salton, G., Singhal, A., Mitra, M., e Buckley, C. (1997). Automatic text structuring and summarization. *Information processing & management*, 33(2):193–207.

Schrepp, M. e Hardt, A. (2007). Goms models to evaluate the quality of an user interface for disabled users. *Challenges for Assistive Technology*, pages 646–651.

SIDRA (2013). Tabela 5753: Pessoas com deficiência visual, total, percentual e coeficiente de variação por grupos de idade e situação do domicílio. https://sidra.ibge.gov.br/tabela/5753. Acessado em 13/06/2019.

Silva, L. F., de Faria Oliveira, O., Freire, E. R. C. G., Mendes, R. M., e Freire, A. P. (2017). How much effort is necessary for blind users to read web-based mathematical formulae?: A comparison using task models with different screen readers. In *Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems*, IHC 2017, pages 29:1–29:10, New York, NY, USA. ACM.

statCounter (2019). Browser market share in brazil - may 2019. http://gs.statcounter.com/browser-market-share/all/brazil. Acessado em 14/06/2019.

TAW (2019). O que é taw. https://www.tawdis.net/index. Acessado em 17/05/2019.

Teixeira, F. (2015). Acessibilidade: como funcionam os leitores de tela. https://brasil.uxdesign.cc/acessibilidade-como-funcionam-os-leitores-de-tela-3d9b610216e1. Acessado em 18/05/2019.

W3C (2013). Pesquisa sobre uso de tecnologias assistivas: Ampliadores e leitores de tela. http://acessibilidade.w3c.br/pesquisa/resultados-preliminares/. Acessado em 14/06/2019.

W3C (2014). Cartilha de acessibilidade na web. http://www.w3c.br/pub/Materiais/PublicacoesW3C/cartilha-w3cbr-acessibilidade-web-fasciculo-I.htm1. Acessado em 20/05/2019.

Wan, X. e Hu, Y. (2015). Braillesum: A news summarization system for the blind and visually impaired people. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 578–582.