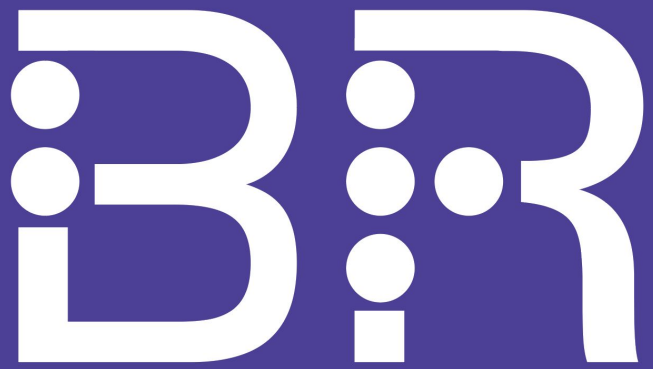


BetterRead, better heard

William Cupp, Madeline Eckhart, Sean Hearne, Isiah Lloyd

Advisor: Dr. Badri Vellambi

Goals

BetterRead's main goal is to create an accessibility tool meant to improve the web browsing experience for those with visual impairments. BetterRead uses machine learning to identify important content of web pages and ignore the superfluous information that someone with visual impairments shouldn't worry about.

Intellectual Merits

- Built a piece of software designed to be used by those that are visually impaired
- Voice speed, pitch, and volume available in web extension for customization
- BetterRead uses innovative machine learning to identify important content of web pages
- Focuses on identifying and presenting the main content of a webpage while avoiding extra components such as ads and pictures
- Built a dataset for model training based on nearly 6000 webpages from Wikipedia's collection of recommended articles

Broader Impacts

BetterRead's purpose is to promote diverse internet usage by allowing vision-impaired users to navigate the internet and give them experiences similar to able-bodied users.

- BetterRead directly impacts vision-impaired users and we hope to improve their experience and relationship with the internet
- Through this project, we hope to bring more awareness to the vision-impared community and the technologies available to them

Logo Design

- Simple, clean, and straightforward
- ‘B’ and ‘R’ from BetterRead name
- Incorporated braille for B and R
- Tagline “BetterRead, better heard”

Logo Presentation

BetterRead, better heard

Meaning

BR

B R

Color

#EC9762

#F26822

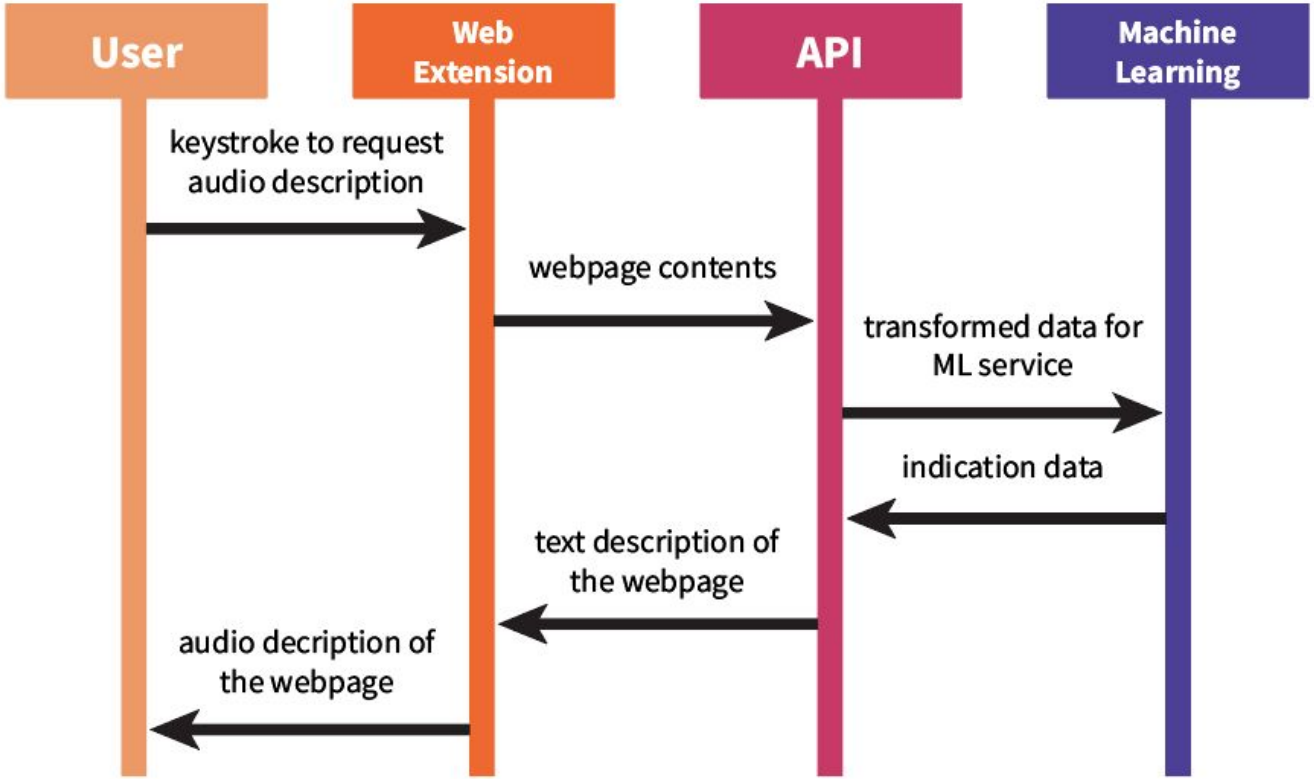
#C83965

#4C3d98

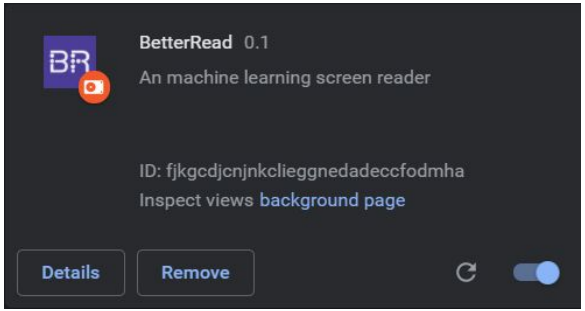
Design Specifications

- Three components make up our project:
  - WebExtension
  - REST API
  - Machine Learning Service
- WebExtension is the user facing application
- WebExtension communicates to the machine learning service using the REST API

Design Specifications



Technology - Web Extension



- Written in HTML/CSS/Javascript
- Multi-platform (Chrome, Firefox, Edge)
- Can be activated by a keystroke or by clicking the logo
- No special permissions (Privacy!)
- Allows users to change speech engine settings, such as rate, pitch, and voice type
- How it works:
  - On user activation a “content script” is injected into current website
  - The content script collects the website’s DOM and sends it back to the extension
  - The extension sends the DOM to the REST API using Fetch
  - When promise is completed, data returned from API is spoken using `chrome.tts` API
  - Using the `chrome.commands` API, we allow users to use (and customize) a keystroke to activate the extension

Technology - REST API

- Hosted on Amazon Web Services with Serverless
- Written in Python
- How it works:
  - Breaks down DOM received from web extension to prepare the data
  - Sends prepared data to the ML service in multi-threaded process
  - After receiving ML response, creates a new text string to send to web extension

Technology - Machine Learning

- ML model developed with Keras Frontend for TensorFlow
- Hosted on Amazon Web Services SageMaker
- How it works:
  - Our Convolution Neural Network model is trained on our custom dataset in AWS SageMaker
  - The model is deployed to an endpoint
  - The REST API sends data to the ML endpoint.
  - The trained model classifies the data
  - The ML endpoint replies to the REST API with the result.



Scope

- Our original goal was to create a machine learning model capable of handling any website regardless of contents or structure
- After starting development of the ML model, we decided to adjust our scope to focus directly on informational websites and wikipedia style articles
- The new focus allowed us to develop a dataset and ML model that produces better results with improved accuracy
- Our reduced scope provided us a faster iteration time in our development cycle

Milestones

Milestone	Description	Delivery
v 0.1	Gather Research	1/27/2021
v 0.2	Develop Web Extension	2/14/2021
v 0.3	Develop REST API	2/21/2021
v 0.4	Develop Machine Learning model	3/7/2021
v 0.5	Refine Connections, Clean up code	3/14/2021
v 1.0	Test screen reader on website for final presentation	3/23/2021

Results

- We have developed a frontend for our system that users can interact with
- Built a backend API that connects the user to our machine learning model by preparing webpage DOMs into usable data
- Engineered our machine learning model to provide useful results for the user
- All of these components connect to make a system that allows the web to be more accessible to those with visual impairments

Challenges

- Learning new technologies
  - Serverless framework to create REST-API to communicate with AWS SageMaker
- A significant challenge was designing an interface for a web extension that is accessible to people with vision impairments.
  - To overcome this challenge, we researched design of existing software tools for vision impaired people.
- Another significant challenge was the development of a machine learning model that can give useful results to the user.
  - To overcome this challenge, we tested different models on small amounts of data and compared our results.
- In order to achieve a well trained model, we needed good data.
  - We gathered raw data and devised a method to transform our data in such a way that it would be useful and conducive to training.

Thank You