Automated W3C Accessible Document Transformation

Sponsor: WSU Libraries

Mentor(s): Ananth Jillepalli

Trent Bultsma, Reagan Kelley, Marisa Loyd

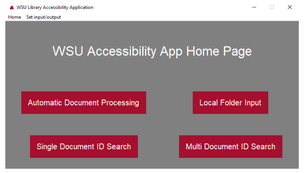
School of Electrical Engineering and Computer Science

Project Overview

The main goal of our project was to harvest documents from the Research Exchange website, process the data within them and export a PDF containing this data that is accessible according to the WCAG guidelines specified by the W3C organization. For this project, our team decided to focus on three components of accessibility : metadata, tags and color contrast. In our pipeline, after a document is downloaded from the Research Exchange website, correct metadata is added and then a tag tree is generated for the document. The tag tree is saved as a layout blocks object in a local folder so that the exporter can access them.

User Interface

Our user interface was designed to be simple, easy to use, and responsive. It consists of a homepage for navigation, four main pages for our different modes of document processing, and a page to adjust settings for input and output. Our four processing modes include: a bulk processing mode to just go through each document from the online repository and run the accessibility pipeline over each one, a mode that allows for inputting documents from a local folder instead of off the online repository, a mode that downloads a single document given its unique identifier and processes it, and a mode that processes through a list of documents given their identifiers. To help with responsiveness, each of these modes have a type of feedback on the progress of the application, be that a progress bar or text displaying how many documents have been processed so far. Additionally, our user interface implements the use of multi-threading to allow the program to be running in the background, processing documents, while also being able to be clicked on and interacted with in the foreground. For ease of use, each page describes how documents are to be inputted and processed. Our user interface design from a technical perspective allows for easy adaptation or a switch to a different UI framework due to the separation of the application class and an application controller class. A new interface for something like a mobile or web application perhaps could be developed and still use the same backend controller class to run everything without having to change the functionality, which is a good example of designing for change in our project.

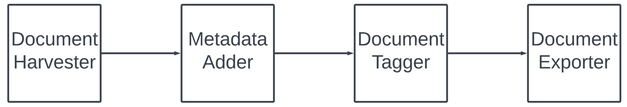


Metadata Adder

Adding metadata to the PDF documents is an important accessibility feature as it allows readers to access key information about the document in a consistent location. Metadata fields include things such as the title or authors of a document as well as a description or keywords and are embedded in the properties of the file, so a reader doesn't need to go through search for it on their own. Typically, documents on the Research Exchange repository do not include this kind of data in the file itself but do often have it on the website. We are able to access this metadata using HTTP get requests to the repository. We then embed that metadata into the actual file when it gets exported to allow for greater document accessibility. We also developed a method of adding metadata through a spreadsheet upon the input of local files as opposed to ones downloaded from off the repository. Once that data is obtained, it is added to the file in the same way for both methods.

One unique form of metadata we gathered that wasn't from the repository itself was that of keywords or phrases. We developed a system that goes through the document looking for commonly occurring combinations of words and picks the most common handful to designate as keywords. Initially, a solution was implemented that compared word frequency of a particular document to the frequency of a larger dataset, but this was abandoned for the simpler local approach due to time constraints. In the future, a better keyword system could be developed if a representative distribution of words for the types of documents on the repository was collected.

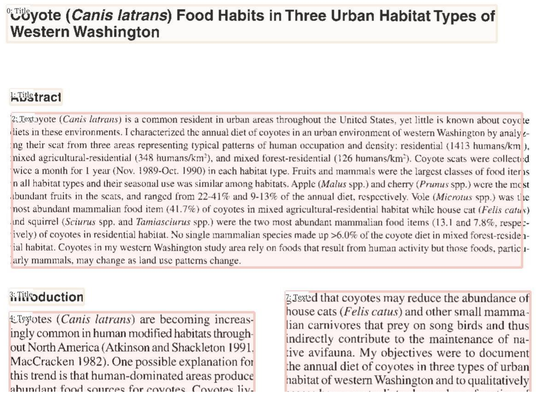
Pipeline process



Document Harvester

When we read a PDF, the read-order is usually quite intuitive for us. We read left-to-right and up-to-down. Moreover, if we a reading documents with columned text, we can naturally parse and easily see which paragraph we should read next. If a paper contains figures, images, or tables, we can easily read through them without disturbing the flow of information.

We wanted to capture this intuition when harvesting the PDF data, but the PDF file data itself would be of little help. There is no universal way to construct a PDF. The end product that we read with our eyes as a plain-text document may seem simple in its design, but in the backend, data is aggregated into sections that don't necessarily follow any concrete framework. This makes data extraction hard since if we were to just read a PDF as any other file, we often get gibberish and unordered paragraphs.


Coyote (*Canis latrans*) Food Habits in Three Urban Habitat Types of Western Washington

Abstract

Coyotes (*Canis latrans*) are becoming increasingly common in human modified habitats throughout North America (Atkinson and Shackleton 1991, MacCracken 1982). One possible explanation for this trend is that human-dominated areas produce abundant food sources for coyotes. Coyotes liv

Introduction

Coyotes (*Canis latrans*) is a common resident in urban areas throughout the United States, yet little is known about coyote diets in these environments. I characterized the annual diet of coyotes in an urban environment of western Washington by analyzing their scat from three areas representing typical patterns of human occupation and density: residential (1413 humans/km²), mixed agricultural-residential (348 humans/km²), and mixed forest-residential (126 humans/km²). Coyote scats were collected twice a month for 1 year (Nov. 1989-Oct. 1990) in each habitat type. Fruits and mammals were the largest classes of food items in all habitat types and their seasonal use was similar among habitats. Apple (*Malus* spp.) and cherry (*Prunus* spp.) were the most abundant fruits in the scats, and ranged from 22-41% and 9-13% of the annual diet, respectively. Vole (*Microtus* spp.) was the most abundant mammalian food item (41.7%) of coyotes in mixed agricultural-residential habitat while house cat (*Felis catus*) and squirrel (*Sciurus* spp. and *Tamiasciurus* spp.) were the two most abundant mammalian food items (13.1 and 7.8%, respectively) of coyotes in residential habitat. No single mammalian species made up >6.0% of the coyote diet in mixed forest-residential habitat. Coyotes in my western Washington study area rely on foods that result from human activity but those foods, particularly mammals, may change as land use patterns change.

Coyotes may reduce the abundance of house cats (*Felis catus*) and other small mammalian carnivores that prey on song birds and thus indirectly contribute to the maintenance of native avifauna. My objectives were to document the annual diet of coyotes in three types of urban habitat of western Washington and to qualitatively

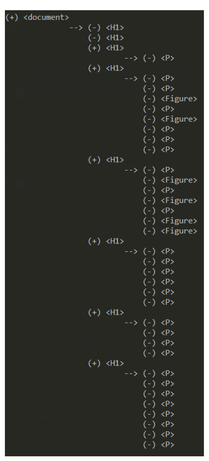
In our solution to this, we relied on machine learning models. Instead of parsing the PDF data, we snapped an image of a PDF, and used pretrained image processing models that could detect blocks of text through an image. These models could also detect what type of information was being displayed, such as titles, body-text, or figures. While this simplified the problem by giving us a collection of boundary boxes, we did not yet have the layout or read order for these blocks. We then created a complex logic tree, through our understanding of English which used the XY coordinates of these blocks to sort them by read order. These blocks were still just images of text, so we then used OCR to extract the text within the images to get text data but also the layout information, maintaining the comprehension of the document.

Document Tagger

Once we were able to harvest the document text and identify each layout block's type using machine learning we could now build a reliable tag tree. As required by the W3C guidelines, accessible web content requires tag trees. The tag tree provides structure and labeling to each element in a document giving software the means to use the document to its fullest potential and eliminate any ambiguity.

The tag tree generation process requires a tree data structure of which each node in the tree contains a tag label and the data for that element. In headers and paragraphs this may be plain text or strings. However, for figures or tables, the data may be more complex as it may represent pixel data for image output.

In the tree generation process we traverse each layout block in its processed read order. The tree determines the hierarchy of elements by a predefined tag precedence. The logic behind this is that paragraphs typically are after headers and are also expansions of what ideas are described in the header. Thus, paragraph tags are children of header tags. We expand this to all tags and we can generate a tag tree.



Document Exporter

The main goal of the document exporter in our project is to export an accessible version of a PDF downloaded from the Research Exchange website based on WCAG guidelines and the data processed in the previous parts of the pipeline. These guidelines are in place to help create a universal standard of accessibility for all documents including PDFs, word documents, and HTML. In order to export the processed data to a PDF, an HTML is generated from the tag tree's data. Within the tag tree structure there are six heading tags and a paragraph tag and the font size for the document is determined by the type of each individual tag in the tree. The HTML is created with these set font sizes, which adhere to WCAG standards, and is then passed to the exporter function to create the new PDF. This function uses the Puppeteer library to create a web page for the new PDF and transfer the content of the HTML onto this page. The Puppeteer library is also able to set the margins and format of the PDF created to ensure they match the WCAG guidelines.

The biggest challenge faced during the creation of the exporting functionality of the project was integrating multiple different languages. Python was the base programming language for our project and node js was a secondary language used to create the accessible PDF from the HTML document. Quite a bit of research and trial and error were used to find a working implementation of the node js function in python. In the end, a subprocess of the node js function was created in python to call the node instruction needed to run the function, followed by the node js function and, finally, the name of the HTML. A PDF is created based on this input and placed in the output folder in the project directory. The name of this PDF will be determined based on the name of the HTML used as input when calling the function.

Future work

Future work for our project would include adding more accessibility features such as alternative text for images, extracting images using the layout parser and putting them back into the accessible PDF, formatting in the document exporter, and finding a way to upload the accessible PDFs directly to the Research Exchange website.

Glossary

WCAG – Web Content Accessibility Guidelines that define how to make web content and physical documents more accessible to people with disabilities.

W3C – World Wide Web Consortium, which is a community that develops open standards for the internet to ensure long term growth of the web.

PDF – Portable Document Format that displays information, including text and graphics, in a way that looks like a printed document.

HTML - Hypertext Markup Language that is used to display information on a webpage.

Alternative text for images – Text explaining why an image belongs in a document and how it relates to the other content therein.

OCR – Optical character recognition; converts an image of text into a machine-readable format.

Research Exchange – Online repository containing articles, book chapters and other educational resources for Washington State University.

Acknowledgements

Our team would like to thank Ananth Jillepalli, our professor for his wisdom and help during the project. We would also like to thank our sponsor, WSU Libraries, for submitting this project, as well as Talea Anderson, our contact from WSU Libraries, for meeting with our team weekly and providing guidance throughout the project.

WSU Libraries Accessibility Team