STUDENT PROJECT PROPOSAL
DATA AND INFORMATION: COMPUTATIONAL LINGUISTICS

HTML Parser with NLTK
HTML Parser

Cody Bonney, *boncod27@evergreen.edu*
Linda Sok, *soklin30@evergreen.edu*
Jamie Bown, *bowjam16@evergreen.edu*

Real-World Problems Addressed by the Project

The World Wide Web has an endless amount of information available to expert and novice Internet users. Unfortunately, there are millions of pages of clutter to sift through to find accurate and useful information. As a team, our goal is to create a program that will address these problems. Our project's aim is to design a Python program and demo website that finds, extracts, and stores relevant content from the web. The program we plan to design will reduce the time that it takes to process data by removing the advertisements, images, and the html tags and will store the useful text and cite the link of the web page onto our own server. Our project's website is http://www.webstripper.org.

Our HTML parser will reduce processing time in several significant ways. First, it will bypass all of the images, third-party scripts, and advertisements by stripping them out of the source code before the webpage is ever processed and loaded onto the screen. This program will save the time it takes to load, copy, and store information from the web because the content will be reduced to raw text. Finally, it will decrease the time it takes to retrieve the files for future reference by saving the refined files locally on our own server.

This program should eliminate the majority of web-based advertisements. Over the years, web ads have gotten more and more annoying. By stripping the pages down to raw text we should be able to avoid ever seeing any flashing banner ads, interactive ads, JavaScript prompts, obnoxious pop-up ads, or ads with irritating sounds.

The automatic extraction of the text will also decrease human errors that might occur when the text is manually entered or mined from the web.

Relationship of the Project to DandI Program Themes

Our project implements all four aspects of the DandI program: Python, Case Studies, Linguistics, and our Semantic Web Series.

We will write a program in Python that will crawl for information on the web starting with a search engine's most relevant results and that will extract, capture, and store text and links to the corresponding web pages. This element of our project reinforces the semantic web series portion of the program as we delve deeper into our understanding of how language facilitates better web searches. Our meta-crawler will be similar to the methodology that William Lewis

described both in his lecture and in his paper with Fei Xia, *Developing ODIN: A Multilingual Repository of Annotated Language Data for Hundreds of the World's Languages*. We will create our own repository of texts extracted from the web and the provide links to the source of HTML documents by using Google's vast repository and methods that we heard about from Shauna Eggers. Week 7's readings *The Anatomy of a Large-Scale Hypertextual Web Search Engine* and *MapReduce: Simplified Data Processing on Large Clusters* are also related (Brin and Page, 1998; Dean and Ghemawat, 2004).

Our program will also take advantage of our Case Studies' labs dealing with Python's NLTK library. We will be using NLTK to tokenize, break down strings of text, and separate the lexemes from the lemmas, also called morphemes, to identify word stems, as well as stripping away the HTML tags from the text (pp. 59-62, 306-7). This also compliments our studies in linguistics, our readings and assignments have taught us about morphology, the structure of words, and syntax, the structure of sentences (Ch. 3-4, Fromkin, et al., 2007).

The Project Itself

The HTML Parser with NLTK incorporates DandI's Python, Case Studies, Linguistics and Semantic Web Series into a useful computational linguistic application.

Our program will start by asking the user for a search string. Once the string is entered, it will submit that string to Google's search engine. It will then analyze the search results to determine where the destination URL is for each particular result. The program then finds the source codes for each of those URLs and strips out all of the images, ads, links, and text that does not consist of more than 10 sequential words. Hopefully, we will be left with mostly raw text containing the information that we searched for. The program takes this raw text and writes it to a brand new HTML file and writes a link to the URL where the information came from. It will then ask if the user would like to upload the new information to a server. If the user selects yes, it will take all the new HTML files created by the program and load them into our online database. The deliverables we plan on presenting will be the python program and a demo of the website at http://www.webstripper.org.

After the class is over, this project will be further developed. The company that we have decided to host the website with supports Python on their servers. Because of this, in the future we can make it so that the Python scripts can run from a browser rather than a command prompt. This would also allow for it to be used by the general public. Some possible improvements in the future would consist of a friendly, web-based user interface that is fully functional and fits The World Wide Web Consortium standards for public use. We may also implement powerful categorization scripts that would organize searches and results from past queries.

What you (think you) will learn from doing the project?

We will integrate everything we have learned in the DandI program in a useful, live application. Our project will reinforce our NLTK practice from the Case Studies labs, increasing our aptitude of both Python and the NLTK library. We will learn how to write a meta-crawler to find appropriate web content, extract and capture the relevant text, remove HTML from the

document, and upload the files into a live server. We are also considering the use of pop-up windows, a GUI feature of Python's that DandI has not explored. As a small group, we will also gain collaboration experience by programming, writing a paper and presenting a demonstration of our program to our class.