Copy Web Site Translating

Ergül Ferik
ergulferik.1@gmail.com

Abstract

The current code includes Python code that saves the links of a website and then uses Google Translate to translate web pages in the specified language. A class named "LinkSaver" saves all the links of the specified homepage and keeps the list of saved links in a text file. Another class named "WebSiteTranslator" opens the saved links, translates the pages into the specified language using Google Translate, and then saves the translated pages.

1. Introduction

The “LinkSaver” class is the class responsible for saving information on a website.. The class takes two parameters in its constructor: "mainPage" which is the link of the website to be searched, and "TXT_NAME" which is the name of the text file in which the found links will be saved. If the "TXT_NAME" parameter is not provided, the default name "Links" is used.

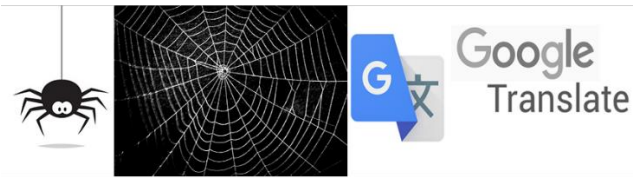
The class uses the Chrome web driver to automate the process of navigating through the website and finding links. The web driver is initialized in the constructor and a new Chrome session is started. The web driver service is set to the local path of the ChromeDriver executable file.

The class has a method named "SaveOneLevelLinks" which is responsible for finding links on the main page and saving them to the text file. The method first navigates to the main page using the web driver and then finds all the anchor elements ('a') using an XPATH query. For each anchor element found, the method calls another private method "_SaveLinks" to save the link.

The private method "_SaveLinks" checks if the link is already saved in the text file. If not, it is saved to the text file with the provided "TXT_NAME" parameter. The saved links are checked for duplicates and only unique links that start with the main page URL are saved to the text file.

Overall, this class provides a convenient way of automating the process of finding and saving links from a website. It can be used for various purposes such as web scraping, link analysis, and search engine optimization.

LinkSaver()



The WebSiteTranslator class is designed to translate web pages to a specified language using Google Translate. It has an __init__ method that initializes several attributes of the class. The MAIN_PAGE parameter is the URL of the main page that will be translated along with its linked pages. The LANG parameter specifies the target language of the translation, which is set to English by default. The TXT_NAME parameter is the name of the text file where the links of the web pages are stored.

The class also initializes a few variables such as _driver, _actions, savedLinks, and linkCount. _driver and _actions are initialized to None for later use in the class. The chromedriver_autoinstaller.install() function installs the appropriate version of the Chrome WebDriver that matches the installed version of the Chrome browser. The options attribute is used to set additional options for the Chrome

WebDriver, such as setting the target language to be used in translations.

The prefs attribute is a dictionary that is used to set preferences for the Chrome browser. The "translate" key is set to {"enabled": "true"} to enable Google Translate. The options.page_load_strategy attribute is set to 'normal' to specify that the page should be loaded normally. The options and service attributes are then used to initialize the _options and _service attributes of the class.

The savedLinks attribute is a list that will contain all the links that have been saved by the LinkSaver class. The linkCount attribute is initialized to 0 and is later updated by reading the text file containing the saved links. The with statement is used to open the text file with the 'r' mode to read the links that have been saved. The for loop then iterates through the file and increments the linkCount attribute for each line in the file.

The code provided is a Python program that translates a list of web pages into a specified language using Google Translate. The program uses Selenium WebDriver and BeautifulSoup libraries to perform web scraping and modify URLs. The program first asks the user for confirmation to translate all the web pages listed in a file, and then proceeds to open each web page, translate it, and save the translated version as a separate file. The program modifies the URLs to match the filenames of the saved files.

_ModifyUrls() function

The _ModifyUrls() function takes an HTML document as input and modifies the URLs of all the links in the document. The function uses BeautifulSoup library to parse the HTML document and find all the 'a' tags. For each 'a' tag, the function checks if the link is present in a pre-existing 'links.txt' file or if the link is a sub-link of the main page. If the link is present in either of these cases, the function calls the _CreateFileName() function to create a filename for the link and replaces the href attribute of the 'a' tag with the new filename. The function then returns the modified HTML document.

_CreateFileName() function

The _CreateFileName() function takes a URL as input and returns a filename for the corresponding HTML file. The function first strips the URL of any trailing or leading spaces. It then checks if the length of the URL is less than or equal to 1 or if the URL is the same as the main page URL or the main page URL without the trailing '/'. If any of these cases are true, the function returns "index.html" as the filename. If the URL ends with a '/', the function removes it. If the URL starts with the main page URL, the function replaces the main page URL with '-' and replaces all '/' characters with '-' characters. If the URL starts with a '/', the function removes the '/' character and replaces all '/' characters with '-' characters. The function then returns the filename with '.html' extension.

Translate() function

The Translate() function is the main function of the program. The function first asks the user for confirmation to translate all the web pages listed in a file. If the user confirms, the function uses the Selenium WebDriver to open each web page listed in the input file, and then performs the following actions: Waits for the web page to load completely. Checks if the web page is already in the target language. If not, right-clicks on the page to open the context menu, navigates down to the 'Translate to' option, selects the target language, and waits for the page to be translated. Scrolls down to the bottom of the web page. Saves the translated HTML page with the filename obtained from the _CreateFileName() function. The function prints the current link count and the total number of links being translated to keep the user informed of the progress.

_AreYouSure() function

The _AreYouSure() function asks the user for confirmation to translate all the web pages listed in the input

file. The function prints the number of links in the file and asks the user to confirm by entering 'Y' or 'N'. The function returns True if the user confirms and False otherwise.

2. Approach

First of all, I knew that using the "selenium" library would put me at ease since I had worked with it before, and that's why I chose this library to navigate the links. After a short while, I was able to easily obtain the desired data. Then, I focused on the most important issue: translation.

Based on my education and experience, I knew that Google Translate would be the best translation tool, but how should I use it? I knew that I could find Google Translate as a library in Python, but doing the translation in Python would be in fragments and there would be no coherence. Therefore, I used Google's Web Site Translator directly. With this solution I found, I was sure that there would be no errors in the translation phase.

After the websites were translated into the desired language, I needed to integrate the links I found into the websites via a one level search. Here again, I used the "Selenium" and "BeautifulSoup" libraries in a compatible manner to communicate with each other. With these libraries that communicate with each other, I was able to easily modify embedded links.

After all these steps, the only thing left in Python was to save them. I tried many different methods for saving (I will explain them later). As a result of my trials, I thought that using the "Selenium" and "BeautifulSoup" libraries again would be the most logical solution. Of course, Python's own file storage system also made my job much easier.

3. Experiments

One Level Search: As recommended in the task video, I attempted to perform a one level search using the "HTTrack" application, but I could not bypass the website security. Then, I found an alternative application called "Wget," but I read that it did not work optimally on Windows. Therefore, I installed a virtual machine with Linux on my computer, but I still could not bypass the website security with Wget. Then, I decided that I needed to write my own program rather than using ready-made applications. With my Selenium experience, I was able to gather the links on the website quickly. In fact, I initially made a mistake and performed an infinity level deep search instead of a one level deep search, resulting in finding more than 1100 URLs.

Translating: At this stage, I used Google Translate, which was recommended to me, as I wouldn't have done anything different myself. I think it is the most successful translation tool. Here, I used the "googletrans" library, which is a Python library developed by Google. However, there was a major problem: sentence coherence! When I translated with Python, the sentences came in pieces, and this was unacceptable for translation. After a little research, I found Google Translate's website translation page. This page was perfect for me because it directly translated the website address, I entered into the desired language. However, a big problem arose here. This website had a robot protection system, and I was trying to enter the site with Selenium, which automatically opened itself as a robot. I knew I could turn this off, but I thought Google had more advanced protection than that. After a thorough investigation, I learned that this website did not have robot protection before version 95 of Google Chrome. However, this brought another problem. Selenium always works with the latest version of Chrome. But there was a solution. Selenium could work with Docker in an integrated way. I ran Chrome version 89 on a Docker and directed Selenium to Docker. Everything seemed great, except for errors. Since Chrome was an old version, I encountered a lot of errors before the website even opened. Desperately browsing on ClassCentral, I right clicked and translated the page into my native language, Turkish, and started investigating. During this time, I realized that I could do the same thing in Python and that my process would work better than others. Using the "pyautogui" library, I performed automation and set the language I wanted to translate to as Hindi.

Modify Urls: My first experiences with web automation were with selenium. With this library, I was able to easily modify links within a website, but the real problem was the registration process. The website allowed me to change the links within it, but when I downloaded it or performed a different action, it added the link of the page that was open to the link. For example, when I tried to change the value of "www.example.com/example1.html" to "www.example.com/example3.html", it returned to me the value "www.example.com/example1/example3.html". While searching for a solution to this problem, I was experimenting with the BeautifulSoup library, which I

discovered, and realized that it could actually be used much more effectively with this library, so I chose to use it.

Downloading: At this point, my trust in the BeautifulSoup library had increased significantly. While browsing its documentation, I saw that there was a method to save websites, and I had seen that this method was done using Python. I thought that trusting Python in an application I developed with it was the most logical option, so I chose to use the method that I used with Python.

4. Conclusion

I used libraries such as Selenium, BeautifulSoup, and pyautogui to use Google Translate's website translator tool to translate an English website to Hindi. The translated websites (one level) were saved on the computer at a predetermined location and then uploaded to the internet using a web hosting service. The given website (www.classcentral.com) was well-designed and secure, which made it challenging and led to some errors, but the project resulted in a completed website.


Project

5. Acknowledgements

References

[1] https://www.selenium.dev/documentation/en/
[2] https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[3] https://github.com/SeleniumHQ/docker-selenium
[4] https://py-googletrans.readthedocs.io/en/latest/
[5] https://docs.python.org/3/
[6] https://www.vmware.com/