Smogon API Technology Review

By Jason Wang, Marques J Chacon, Ramiro Steinmann Petrasso, and Yangyang Yao

Slide 1: Background & Use Case

- Our project is focused on creating an API that allows its users to programmatically access data from https://Smogon.com
- Using simple GET requests, an app that uses our API should be able to receive the public data on Smogon, but in JSON or CSV format
- In order to accomplish this, we need to be able to deploy our Python Code online
- We also need to be able to parse the raw Smogon sites and scrape the data from there

Slide 2: Python Package Choices

Package requirement: We need a package that enables us to parse Smogon and extract web data

- Possible candidate: BeautifulSoup
    - Author: Leonard Richardson
    - Description: BeautifulSoup allows us to parse HTML code in Python by grabbing a website as HTML from the web and representing it as an HTML tree
- Possible candidate: Selenium
    - Author: Jason Huggins (originally)
    - Description: Selenium allows to collect data by automating browser tasks; it allows our code to take control of a browser and automate it to grab the data we need

Package requirement: We need a package that enables us to deploy our Python code to a web server so it can act as an API

- Possible candidate: web2py
    - Author: Massimo DiPierro
    - Description: Web2py allow us to dynamically create web content with python code, allowing us to generate websites with our code
- Possible candidate: FastAPI
    - Author: Sebastián Ramírez
    - Description: FastAPI is a python package that allows us to make REST API endpoints using python, allowing us to deploy them to a web server

Slide 3a: Package Comparison (BeatifulSoup vs Selenium)

BeatifulSoup Pros:

- Tree representation of HTML documents makes it easy to hone in on what's necessary
- Runs fairly quickly

BeatifulSoup Cons:

- Cannot easily interact with dynamic javascript-based pages
- Cannot "navigate" to other pages if necessary (other than by extracting hyperlinks and loading them again with BeatifulSoup)
- Has caused people to be blacklisted if cooldowns not implemented (worst case scenario for an API)

Selenium Pros:

- Can interact with dynamic pages very easily through a browser
- Can navigate through pages
- Can inject HTML into a page if necessary (good for adding placeholders and resolving inconsistencies)

Selenium Cons:

- Requires a browser to be installed and drivers to be configured (difficult to set up in a web server or container)

Slide 3b: Package Comparison (Web2py vs FastAPI)

Web2py pros:

- Easier to make a frontend (good for embedded documentation of the API)
- Plenty of dedicated sites exist that make deployment a one-click process

Web2py cons:

- Harder to return non-HTML components
- Can be slow

FastAPI pros:

- Native REST endpoint support, making it easy to return JSON or CSV data
- Tends to be fast, which is good for preventing request timeouts
- Code tends to be more lightweight and easier to read/debug

FastAPI cons:

- Harder to deploy (usually involves configuring a dockerfile)
- Harder to return HTML for documentation

Slide 4: Our Choices

- For our packages, we ended up choosing BeautifulSoup and FastAPI
- While Selenium is generally the best choice for scraping web pages that are even partly dynamic, remotely configuring a headless browser and the accompanying driver with the limited time we have available for the project means having to sacrifice a lot of time we would otherwise use to actually code
    - This is especially impractical given that the majority of Smogon.com content seems to be readable off of the HTML alone
- We chose FastAPI for our web framework because the REST support would enable us to fulfill our core functionality the easiest, and because of the ease of debugging

Slide 5: Drawbacks/Remaining Concerns

- Lacking access to Selenium from a web server may make it harder to integrate our Smogon API with data from other sources, if we decide to go that route
- We'll likely still need to provide some sort of landing page to explain how to use our API, probably by returning raw html as a String
- A lot of Smogon data is formatted in html but contained inside a script tag, within a JSON object
    - Not a huge problem overall, since BeautifulSoup can look at it even when surrounded by other JS code, since it supports malformed trees
    - May need a way to traverse JSON to fish out some data

Slide 6: Demo

We now switch screens to our demo!