Some Early Bonus Points:
Using Python to Parse Internet Data

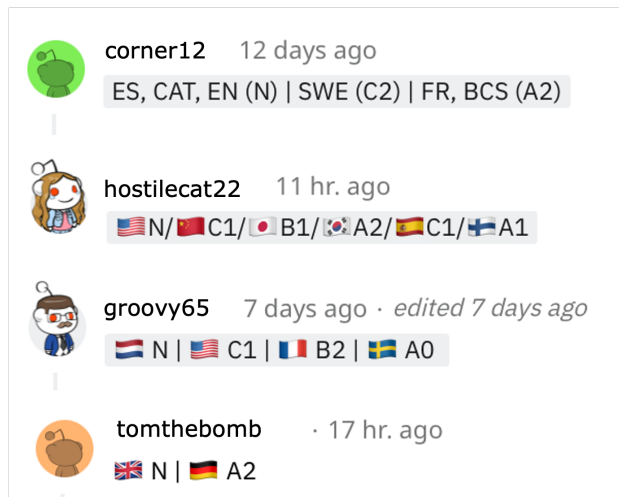
ML4FS
Prof. Overdorf

Spring 2024

Motivation

Data taken from the Internet is often messy. Nonetheless, we must be able to process it into something useful. In this assignment, you'll need to do just that. With real data pulled from the webpage `reddit.com/r/languagelearning`, you'll need to write a program that determines the language proficiency of a user based on their free-form inputs. The challenge is that the users can input their language proficiencies in any form they like.

In the figure to the right, we see that `corner12` speaks English, Spanish, and Catalan natively, Swedish at a C2 level, and French and Bosnian/Croatian/Serbian at an A2 level. User `hostilecat22` speaks English natively, C1 Chinese and Spanish, B1 Japanese, A2 Korean, and A1 Finish. User `groovy65` speaks native Dutch, C1 English, B2 French, and A0 Swedish. User `tomthebomb` speaks native English and A2 German. These are easy for us to interpret, but the challenge is writing a program to interpret them.



Assignment Files

For this assignment, we will provide you with 5 files. You do not *need* to use all the files - but you might find them useful.

1. `input_file.csv` an input file containing examples of the user inputs

`input_file.csv`

username,	**user_input**
username1,	ES, CAT, EN (N) | SWE (C2) | FR, BCS (A2)
username2,	🇺🇸N/🇨🇳C1/🇯🇵B1/🇰🇷A2/🇪🇸C1/🇫🇮A1
username3,	🇳🇱 N | 🇺🇸 C1 | 🇫🇷 B2 | 🇸🇪 A0
username4,	🇬🇧 N | 🇩🇪 A2

2. `expected_output_file.csv` the expected output file. If your program is perfect, your output file should match this output file

`exptected_output_file.csv`

username,	**N**,	**C2**,	**C1**,	**B2**,	**B1**,	**A2**,	**A1**
username1,	[es, ca, en],	[sv],	[ ],	[ ],	[ ],	[fr],	[ ]
username2,	[en],	[ ],	[zh, es],	[ ],	[ja],	[ko],	[fi]
username3,	[nl],	[ ],	[en],	[fr],	[ ],	[ ],	[ ]
username4,	[en],	[ ],	[ ],	[ ],	[ ],	[de],	[ ]
...							

3. `iso_codes.csv` a .csv file of iso-language codes. Your output file should represent languages using these codes. (i.e., 🇲🇽 should yield 'es' - not 'mx' and 🇺🇸 should yield 'en' not 'eng' or 'us')

4. `requirements.txt` which lists the Python packages that we will have installed in the VM where we will run your code. If you use a package that is not in requirements.txt make sure you add it to requirements.txt and turn it in with your python file. Note that you do not need to use all of the libraries in `requirements.txt` and can use libraries that are not listed there.

`iso_codes.csv`

Name-en	**ISO-Code**
Afar,	aa
Abkhazian,	ab
Avestan,	ae
Afrikaans,	af
Akan,	ak
Amharic,	am

`requirements.txt`

```
numpy==1.23.5
pandas==1.3.0
scikit-learn==0.24.2
nltk==3.7
scipy==1.9.3
scikit-learn==1.2.2
...
```

5. `countries_to_langs.csv` a file containing counties and the languages spoken in them (by percentage). You might find this useful

`countries_to_langs.csv`

country,	**en**	**es**	**fr**	**de**	**it**	...
BE	'percent': 59.0, 'official': False	NaN	'percent': 38.0, 'official': True	'percent': 22.0, 'official': True	NaN	
CA	'percent': 86.0, 'official': True	'percent': 1.6, 'official': False	'percent': 30.0, 'official': True	'percent': 0.78, 'official': False	'percent': 0.91, 'official': False	
CH	'percent': 61.0, 'official': False	NaN	'percent': 21.0, 'official': True	'percent': 73.0, 'official': True	'percent': 4.3, 'official': True	
AT	'percent': 73.0, 'official': False	NaN	'percent': 11.0, 'official': False	'percent': 97.0, 'official': True	'percent': 9.0, 'official': False	
AU	'percent': 96.0, 'official': True	NaN	NaN	NaN	'percent': 1.9, 'official': False	

6. `checker.py` a file to check your code against the sample input file. Just run:
`checker.py language_parser_lastname.py`
and it will print the number of rows which are correct and display the errors (if any).

7. `language_parser_lastname.py` a skeleton python file to get you started. Change the name of this file to your actual last name (if you have more than one last name, join them with an _ or just use the first) and edit this file. You can add anything to and delete anything from this file, as long as in the end, it takes two command line parameters: the input file to read and where to write the output.

```python
import sys

def parse_language(user_input):

    native_langs = []
    N = []
    C2 = []
    C1 = []
    B2 = []
    B1 = []
    A2 = []
    A1 = []
    # YOUR CODE HERE
    return native langs, C2, C1, B2, B1, A2, A1


if __name__ == "__main__":
    # Check if filename argument is provided
    if len(sys.argv) != 2:
        print("Usage: python3 language_parser_lastname.py input_file.csv output_file.csv")
        sys.exit(1)

        # Get filename from command line argument
    infile = sys.argv[1]
    outfile = sys.argv[2]

    # read in the file, line by line
    with open(outfile, 'w') as out:
        out.write('username,N,C2,C1,B2,B1,A2,A1\n')
        with open(infile, 'r') as file:
            headers = next(file) # 'username,user_input'
            for line in file:
                line = line.strip().split(',') # ['username','user_input']
                username = line[0]
                user_input = line[1]
                parsed = parse_language(user_input) # 'user_input'
                # convert parsed to a string
                s = ','.join(str(lang_level) for lang_level in parsed) + '\n'
                out.write(username + ',' + s)
```

Submission Format

You are required to submit 1 Python file called `language_parser_lastname.py`, which we will run on 100 never-before-seen samples. You can assume that the libraries in `requirements.txt` will be installed on the VM we will use for grading. The file must take 2 command line parameters, the name of the input file and the name of the output file. The headers for the .csv file must be `username,N,C2,C1,B2,B1,A2,A1`.

Your code will be run in the command line like this:

```
$ python3 -m venv .
$ source bin/activate
$ pip install -r requirements.txt
$ python3 lastname.py input_file.csv output_file.csv
$ python3 grade.py output_file.csv reference_file.csv output_file.csv
$ deactivate
```

Submit the code via Moodle.

Bonus Point Distribution

Your score for this assignment will be the percentage of samples that your code was able to classify correctly. Any code that takes longer than 5 minutes to run is disqualified. A 1/2 bonus point on the midterm grade will be awarded to students with the top 10% of scores higher than 25%. **You must work on this assignment alone and without the assistance of an LLM**. Otherwise, there are no limitations to what you can use or how you can write the code. Please feel free to use the Internet to it's fullest extent.