Test for Disruptive Data Summer School

Curlie Category Extractor

# Introduction

The objective of the Curlie Category Extractor is to extract the categories listed on the website <http://www.curlie.org> keeping track of the hierarchy, and save them all to a CSV file.

# Requirements study

Here I will describe the process I followed throughout the development of the application.

## Investigation

The first step was to investigate Curlie website, searching for a public API that allowed me to obtain categories information. Unfortunately, it was not available. Then, I looked for a downloadable format of the data, but alas that was not available either. Therefore, the path I chose was to download HTML pages, parse them and extract useful data.

## Choice of programming language and libraries

My programming language of choice was C#. I have good experience with it, code has high maintainability and readability and has a good ecosystem of useful libraries. In particular, the HtmlAgilityPack (HAP) is available: HAP is a .NET core library (therefore cross-platform) which enables loading HTML pages and traversing them with XPaths.

## Analysis of Curlie HTML pages structure

Through the Developer Tools of Google Chrome, I analyzed Curlie HTML pages to understand their structure and understand where the required data lives. I found two main structures:

* The homepage, where root categories are listed
* Category page, where subcategories are listed

These two pages differ greatly, therefore parsing must be adapted for both.

## Throttling and proxies

During the development, I discovered that Curlie enforces throttling on the number of requests per hour available to a single IP address. The first attempt was adding a throttling to the application, making one request per second. This was not enough to prevent being blocked by Curlie for an hour or so. Therefore, I added a proxy mechanism to alternate the IP address from which the requests are generated. In this way, each IP makes a request every N seconds, where N is the number of proxies available.

# Development

Here I will describe the classes of which the application is composed.

## Category

It abstracts a Curlie Category. It has 3 properties: Name, Url, and Id.

Each Category, except root Categories, are listed as Subcategory of one or more Categories, called Parent Categories. Each Category keeps track of its Parents.

A Category can have a Primary Parent: Primary Parent url is the prefix of the Category url, i.e. /Arts/ is the Primary Parent of /Arts/Music.

## CurlieWebClient

CurlieWebClient handles the outgoing requests to Curlie. It enforces throttling and assigns a proxy to each request. It is coded through the Task-Based Asyncronous Pattern used in C#.

For each page request, a Page Request Task (PRT) is created and enqueued. A background Task handles the start of each PRT, throttling their execution at a configurable rate.

Each PRT prepares an HTTP request to Curlie, chooses a proxy from those available and sends the request through that proxy. Then, the response is transformed in a HtmlDocument and returned to the caller.

## ProxyList

It reads a list of proxies from the file proxies.txt (which must exist in the directory from which the script is called) in the format ip:port:user:pass. It exposes a method NextProxy which returns the next proxy to be used by a PRT.

## Curlie

It is the core of the application. It has four methods.

* Begin: initializes the application and parses Curlie Homepage, starting the extraction of each root category.
* ParseRootCategory: parses the HtmlNode which contains information about a root Category and starts the extraction for each Subcategory.
* ParseSubCategories: parses the page which contains information about Subcategories and for each of them, recursively starts the extraction of their Subcategories.
* WriteToCsv: Formats the extracted Categories into a csv file. Each record has id, name, url, primary parent, and secondary parents. Secondary parents are a list of integers separated by semicolon, while columns are separated by commas.

# Demo

Since there are a huge number of categories and not many free proxies to use, also time is limited, the application does not extract all the Categories in Curlie. Instead, it extracts Categories under the Business root Category, and extracts only 4 Subcategories per Category. In this way, the script can end in reasonable time and the proxies IP are not blocked by Curlie.

# My presentation

I am very passionate about Computer Science and I love working with data, trying to understand it and manipulate it to obtain new useful information.

I recently obtained my M.Sc. in Computer Engineering at the University of Pisa, with Full Marks and Honor, concluding my University studies. During my studies, the exam which I liked the most was surely Performance Evaluation of Computer Systems and Networks, where we were taught how to properly understand and manipulate simulation data: we studied how to parse it and output it in a more usable format, how to compute useful statistics, how to store them and re-use it at a later time.

My Master Thesis was about the training of an Object Detector for Personal Protection Equipment. There I tackled the problem of having little to no data available for training, while still needing the object detector. I studied the feasibility of using virtual datasets, i.e. datasets composed of images generated by 3D Graphics Engines, for the training. The experiments yielded very promising results.

In 2017-2018 I worked at an independent research project, the HMR Project. We studied the feasibility of Knowledge Base Driven Information Extraction, i.e. using Ontology Based Information Extraction with an extra step of assessing the reliability and accuracy of information by learning on existing data on the Knowledge Base in a semi-supervised fashion.

I had also some working experience: I worked at Inera SRL, a Pisa-based IT company, where I developed full-stack an enterprise website, and researched and applied a NoSQL solution for storing the data extracted by an already existing web-crawler.

I worked also with an American company, East Coast Datacom, for which I worked on the GUI for their embedded WAN Emulator, from backend to frontend, from development to deployment solutions. In this job I could put in practice many skills in information extraction and parsing, as I had to parse the low-level information given by UNIX utilities and kernel files to use them in high-level languages.

Besides those already mentioned, my most relevant projects are:

* Face Recognition on Android, an app which uses Machine Learning to recognize faces, mainly for intruder detection.
* Audio Recognition on Android, an app which leverages Audio Fingerprinting for recognizing songs.
* KP-ABE Photo Sharing: A service for secure sharing of images. It uses Attribute Based Encryption to enforce policies on images and allow reading only to authorized users.
* Meme Audiobot for Telegram, a Telegram inline bot which uses cloud technologies (mainly Azure) with CI/CD. It allows to send popular audio quotes or snippets without leaving the cat context.

## Why I want to attend DDSS?

In November I will start my PhD in Machine Learning and Explainable AI. One of the key skills in Machine Learning is the ability of understanding and manipulating data in a smart way. I think that this Summer School will allow me to further hone my skills in Data Analysis, discovering and exploring new concepts and techniques. Furthermore, it is a great opportunity to broaden my professional network, getting to know many new people as passionate as me.