

Forum:[Hack Forums](#)

Thread: [Why is Python such a strong language for machine learning?](#)

The One Who Knocks - April 17, 2020, 2:48 pm

Since Python is an interpreted language, it's much slower than compiled languages like Java or C, In fact, it's so much slower that it's not even a viable option for competitive programming competitions afaiK.

So with that being said, python is still such a strong and common language used for machine learning. Why is that? I understand it's very flexible, and flexibility is probably a valuable property when programming neural networks, genetic algorithms, deep learning systems etc. But would other languages be "faster" or more efficient if some ML program was made in them (such as Java or C++)?

I apologize if this is a stupid question. But I have some classes this summer that will introduce us to AI and I was wondering about this since I've so far only learned Python and a bit of Java, so I can't compare it in my own experience.

Thanks!

Aussie - April 17, 2020, 8:27 pm

Not too sure about this one, can agree with you on the Java and C++ comparison though.

It's probably due to the sheer support and size of machine learning libraries that exist in Python that keeps its position as num. 1 for any data science work.

4Txe4y33C9zvNhMj - April 18, 2020, 1:36 am

Python is fairly easy to learn people can get into building stuff without having to re-write the hard parts for machine learning every time.

The performance isn't too bad either since popular libraries that need high performance (such as TensorFlow, Keras, NumPy, etc.) that do the actual number crunching are written in C/C++.

Charger - April 18, 2020, 12:32 pm

It's super fast to create a program to process data.

Need to pull a JSON api and loop through the data?

code stolen from SO

```
Code import urllib.request
import json
url = 'http://www.reddit.com/r/all/top/.json'
req = urllib.request.Request(url)
```

```
##parsing response
r = urllib.request.urlopen(req).read()
```

```

cont = json.loads(r.decode('utf-8'))
counter = 0

###parcing json
for item in cont['data']['children']:
    counter += 1
    print("Title:", item['data']['title'], "\nComments:", item['data']['num_comments'])
    print("----")

##print formatted
#print (json.dumps(cont, indent=4, sort_keys=True))
print("Number of titles: ", counter)
Now pulling the data in c++ (ew)
Code // main.cpp
#include <cstdlib>
#include <iostream>
#include <memory>
#include <string>

#include <curl/curl.h>
#include <json/json.h>

namespace
{
    std::size_t callback(
        const char* in,
        std::size_t size,
        std::size_t num,
        std::string* out)
    {
        const std::size_t totalBytes(size * num);
        out->append(in, totalBytes);
        return totalBytes;
    }
}

int main()
{
    const std::string url("http://date.jsontest.com/");

    CURL* curl = curl_easy_init();

    // Set remote URL.
    curl_easy_setopt(curl, CURLOPT_URL, url.c_str());

    // Don't bother trying IPv6, which would increase DNS resolution time.
    curl_easy_setopt(curl, CURLOPT_IPRESOLVE, CURL_IPRESOLVE_V4);

    // Don't wait forever, time out after 10 seconds.
    curl_easy_setopt(curl, CURLOPT_TIMEOUT, 10);

    // Follow HTTP redirects if necessary.
    curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

```

```

// Response information.
int httpCode(0);
std::unique_ptr<std::string> httpData(new std::string());

// Hook up data handling function.
curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, callback);

// Hook up data container (will be passed as the last parameter to the
// callback handling function). Can be any pointer type, since it will
// internally be passed as a void pointer.
curl_easy_setopt(curl, CURLOPT_WRITEDATA, httpData.get());

// Run our HTTP GET command, capture the HTTP response code, and clean up.
curl_easy_perform(curl);
curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &httpCode);
curl_easy_cleanup(curl);

if (httpCode == 200)
{
    std::cout << "\nGot successful response from " << url << std::endl;

    // Response looks good - done using Curl now. Try to parse the results
    // and print them out.
    Json::Value jsonData;
    Json::Reader jsonReader;

    if (jsonReader.parse(*httpData, jsonData))
    {
        std::cout << "Successfully parsed JSON data" << std::endl;
        std::cout << "\nJSON data received:" << std::endl;
        std::cout << jsonData.toStyledString() << std::endl;

        const std::string dateString(jsonData["date"].asString());
        const std::size_t unixTimeMs(
            jsonData["milliseconds_since_epoch"].asUInt64());
        const std::string timeString(jsonData["time"].asString());

        std::cout << "Natively parsed:" << std::endl;
        std::cout << "\tDate string: " << dateString << std::endl;
        std::cout << "\tUnix timeMs: " << unixTimeMs << std::endl;
        std::cout << "\tTime string: " << timeString << std::endl;
        std::cout << std::endl;
    }
    else
    {
        std::cout << "Could not parse HTTP data as JSON" << std::endl;
        std::cout << "HTTP data was:\n" << *httpData.get() << std::endl;
        return 1;
    }
}
else
{

```

```

    std::cout << "Couldn't GET from " << url << " - exiting" << std::endl;
    return 1;
}

return 0;
}

```

or in C, slightly less ew:

```

Code /*
 2 * Copyright (c) 2013-2018 Joris Vink <joris@coders.se>
 3 *
 4 * Permission to use, copy, modify, and distribute this software for any
 5 * purpose with or without fee is hereby granted, provided that the above
 6 * copyright notice and this permission notice appear in all copies.
 7 *
 8 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
WARRANTIES
 9 * WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
10 * MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE
FOR
11 * ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY
DAMAGES
12 * WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN
AN
13 * ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING
OUT OF
14 * OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
15 */
16
17 #include <kore/kore.h>
18 #include <kore/http.h>
19
20 #include <yajl/yajl_tree.h>
21
22 int    page(struct http_request *);
23
24 int
25 page(struct http_request *req)
26 {
27     ssize_t    ret;
28     struct kore_buf    *buf;
29     char    *body;
30     yajl_val    node, v;
31     char    eb[1024];
32     u_int8_t    data[BUFSIZ];
33     const char    *path[] = { "foo", "bar", NULL };
34
35     /* We only allow POST/PUT methods. */
36     if (req->method != HTTP_METHOD_POST &&
37         req->method != HTTP_METHOD_PUT) {
38         http_response_header(req, "allow", "POST, PUT");
39         http_response(req, HTTP_STATUS_METHOD_NOT_ALLOWED, NULL, 0);
40         return (KORE_RESULT_OK);

```

```

41 }
42
43 /*
44  * Read the entire received body into a memory buffer.
45  */
46 buf = kore_buf_alloc(128);
47 for (;;) {
48     ret = http_body_read(req, data, sizeof(data));
49     if (ret == -1) {
50         kore_buf_free(buf);
51         kore_log(LOG_NOTICE, "error reading body");
52         http_response(req, 500, NULL, 0);
53         return (KORE_RESULT_OK);
54     }
55
56     if (ret == 0)
57         break;
58
59     kore_buf_append(buf, data, ret);
60 }
61
62 /* Grab our body data as a NUL-terminated string. */
63 body = kore_buf_stringify(buf, NULL);
64
65 /* Parse the body via yajl now. */
66 node = yajl_tree_parse(body, eb, sizeof(eb));
67 if (node == NULL) {
68     if (strlen(eb)) {
69         kore_log(LOG_NOTICE, "parse error: %s", eb);
70     } else {
71         kore_log(LOG_NOTICE, "parse error: unknown");
72     }
73
74     kore_buf_free(buf);
75     http_response(req, 400, NULL, 0);
76     return (KORE_RESULT_OK);
77 }
78
79 /* Reuse old buffer, don't need it anymore for body. */
80 kore_buf_reset(buf);
81
82 /* Attempt to grab foo.bar from the JSON tree. */
83 v = yajl_tree_get(node, path, yajl_t_string);
84 if (v == NULL) {
85     kore_buf_appendf(buf, "no such path: foo.bar\n");
86 } else {
87     kore_buf_appendf(buf, "foo.bar = '%s'\n", YAJL_GET_STRING(v));
88 }
89
90 /* Release the JSON tree now. */
91 yajl_tree_free(node);
92
93 /* Respond to the client. */

```

```
94 http_response(req, 200, buf->data, buf->offset);
95 kore_buf_free(buf);
96
97 return (KORE_RESULT_OK);
98 }
```

or use a lib <https://github.com/udp/json-parser>

tux1 - April 18, 2020, 3:29 pm

(Apr 17, 2020 02:48 PM)The One Who Knocks Wrote: [\[CITE\]](#) Since Python is an interpreted language, it's much slower than compiled languages like Java or C, In fact, it's so much slower that it's not even a viable option for competitive programming competitions afaik.

So with that being said, python is still such a strong and common language used for machine learning. Why is that? I understand it's very flexible, and flexibility is probably a valuable property when programming neural networks, genetic algorithms, deep learning systems etc. But would other languages be "faster" or more efficient if some ML program was made in them (such as Java or C++)?

I apologize if this is a stupid question. But I have some classes this summer that will introduce us to AI and I was wondering about this since I've so far only learned Python and a bit of Java, so I can't compare it in my own experience.

Thanks!

Python is around 50 times slower than C so you may think why would anyone use it for number crunching but there are several reasons.

1. It's an easy language to use so people can focus more on the data side of things rather than complicated software development and debugging
2. Lots of support and an active community
3. Modern computers are powerful enough you probably wouldn't notice much difference in performance
4. Python has the ability to use C/C++ compiled libraries to do the hard work. For example I believe TensorFlow's core is written in C++ and has a Python wrapper to do the other stuff, so the C++ part does the real number crunching

Hope that helps

Nayeon - April 18, 2020, 9:44 pm

It's exactly as Charger & tux1 says feels way too close to home. Python is a slow language but their vast libraries and ease of use makes up for it.