Woody International Publish Limited
An Multidisciplinary Academic Journal Publisher
Journal of Artificial Intelligence and Information, Volume 2, 2025
https://www.woodyinternational.com/

Design and Implementation of Web Crawler Based on C++

Zijin Song

School of Computer and Software, Chengdu Jincheng University, Chengdu 611731, Sichuan, China

Abstract: ***With the booming development and continuous progress of the design and artificial intelligence industries, the demand for high-quality and diverse images is showing an explosive growth trend. This trend is not only reflected in creative design and product development, but also deeply rooted in multiple cutting-edge fields such as machine learning and computer vision. However, in the face of massive image resources, manually accessing websites and downloading images one by one appears time-consuming and laborious, greatly limiting work efficiency and the scale of data acquisition. Therefore, developing efficient and automated tools to collect images has become a top priority. In this context, this article elaborates on the design and implementation of a web crawler system based on the C++programming language. This web crawling system focuses on processing websites that use the HTTP protocol, and can efficiently penetrate webpage structures, accurately locate and extract image resources. Users only need to provide the URL of the target website as a starting point, and the crawler will automatically start working by parsing the webpage content, identifying and downloading all available image files. This process not only greatly reduces the manual burden, but also significantly improves the speed and breadth of image collection, providing a solid foundation for subsequent image processing, analysis, and model training. In addition, the design of the crawler system considers flexibility and scalability, making it easy to adjust and optimize according to specific needs to adapt to the constantly changing network environment and image acquisition requirements.***

Keywords: Crawler; C++; Http; Picture.

1. INTRODUCTION

In modern society, the design industry is developing at an unprecedented speed, and in this process, designers often need a large number of images as creative materials or inspiration references[1]. Similarly, in the field of artificial intelligence, massive amounts of image data are also indispensable for training various deep learning models. However, in the huge ocean of Internet, how to efficiently collect and download these image resources has become a major problem for many professionals[2]. In order to solve this problem, web crawling technology has emerged. A web crawler is a program that automatically captures network information based on preset rules. It can simulate human browsing behavior, quickly traverse web pages, and extract the required content. In this specific web crawling project, we mainly focus on crawling image resources from websites that use the HTTP protocol. The core of the project is to analyze the HTML code of the target website and obtain the URL of the image by accurately locating the src attribute of the image element. Subsequently, the crawler will download images in batches based on these URLs, achieving an automated and efficient image collection process. This method not only greatly improves the efficiency of image collection, but also reduces the burden of manual operations, providing strong data support for the design industry and the field of artificial intelligence.

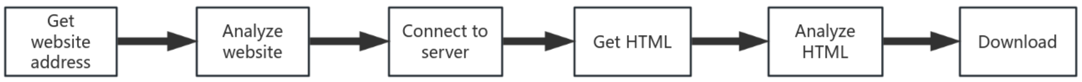


Figure 1: Grasping process

The application of deep learning in web crawling is becoming increasingly widespread, injecting new vitality into this traditional technology field. Web crawlers, as automated tools for collecting network information, have always been a focus of developers' attention on their efficiency and accuracy[3][4][5]. The introduction of deep learning provides a new solution for improving the performance of web crawlers. Deep learning models, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), have demonstrated powerful capabilities in image recognition and text processing[6][7]. In web crawlers, these models can be used to more accurately parse webpage content, identify useful information and images. For example, by training deep learning models to recognize specific elements in web pages, such as images, links, or text blocks, crawlers can more accurately capture target data and avoid downloading irrelevant or redundant information[8]. In addition, deep learning can also help web crawlers better handle complex web page structures[9]. Traditional web crawlers often rely on fixed parsing rules, making it difficult to cope with frequent changes in webpage structure. Deep learning models can automatically extract features and patterns of web pages by learning from a large number of web page samples, thus adapting more flexibly to different web page structures[10]. Overall, the application of deep learning in web crawlers not only improves the efficiency and accuracy of data collection, but also enhances their ability to handle complex web page structures[11][12]. With the continuous development of deep learning technology, we have reason to believe that future web crawlers will become more intelligent and efficient, providing richer network information resources for various industries[13].

2. OBTAIN AND ANALYZE WEBSITE ADDRESSES

When the program starts running, use the CreateDirectory () function to create a folder for storing crawled images. Next, the program will remind the user to enter the URL to be crawled, and then store the URL entered by the user in a string type variable for future use.

Next, load the URL into a char type array and use the strstr() function to check if the input URL is an HTTP protocol URL. If the input URL is confirmed to be an HTTP protocol URL, then the next step is to use the sscanf() function to find the first "/" after "http://". The sscanf() function can find the specified substring in a string and load the front and back parts of the found substring into different variables. By searching for the first "/" after "http://", we can obtain the content of the two parts before and after it. The first part is the host name, and the last part is the resource path.

3. CONNECT TO THE SERVER

Next, we need to connect to the server and send a get request.

We first initialize the socket library and create a socket. Then use gethostbyname() to resolve the host IP and receive its return value using pointer P.

Then declare a socketaddr_in variable, which is a structure encapsulated by the system. The member variable sin_family represents which address family to use, sin_port is used to store port numbers, and sin_dedr stores information about IP addresses. We set its sin_family to AF-INET, indicating the use of TCP/IP protocol; Set its sin_port to htons (80), indicating the use of the default HTTP port number of 80. Then pass the h-addr in pointer p to the sinaddr of the sokaddr_in variable. Finally, the server is connected through the connect() function. The connect() function requests the connection through a socket and the socketaddr_in variable as parameters. If successful, it returns 0, and if the connection fails, it returns an error code.

If the connection is successful, proceed to the next step - send a get request. Here, we need to declare an array of char type to store the get request, and use the sprintf() function to load the requested content into the array; The following content needs to be loaded: "GET% s HTTP/1.1 \ r \ nHost:% s \ r \ nConnection: Close \ r \ n \ r \ n". The first '% s' here is the resource path previously analyzed, and the second is the hostname.

After loading, it needs to be sent through the system call function send(). This function can only be used when the program is in a linked state, so it is placed after connect(), and then sends a get request to the server as a parameter by adding a socket, a char type array containing a get request, and the size of the array calculated by the strlen() function.

4. RETRIEVE AND PARSE HTML

Before crawling the target website, the first step is to determine the collection target, clarify the location information and logical structure of the collection fields [14]. Therefore, after sending the get request, we need to retrieve the HTML and first declare an array and a string to store the HTML. Next, the data returned from the server is received through the recv() function, and a while loop with the return value of the recv() function as the judgment condition is used to determine whether there is still data sent. In each loop, the data in the array is loaded into a string. If the data transmission ends, the loop will terminate, and ultimately we will obtain the complete computer system network and telecommunications HTML of the webpage.

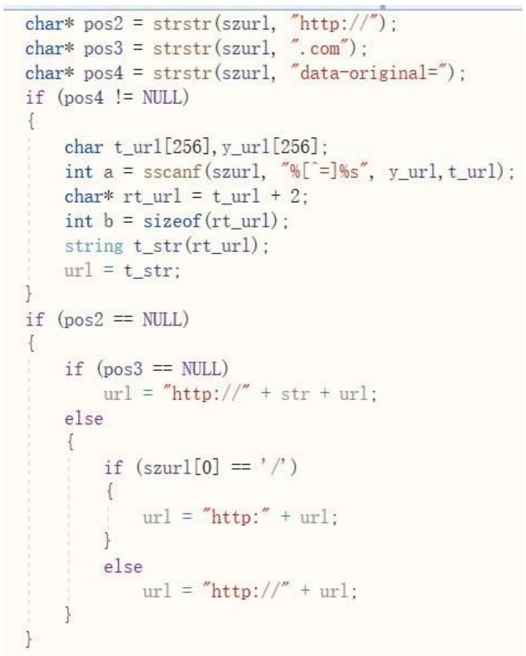
After obtaining the HTML of the webpage, we need to find the src that represents the image and download the image based on the URL provided by src. Therefore, we need a regular expression to make a judgment and find the src that meets the criteria. The content of the regular expression is as follows: "src=\" (. *? \ \. jpg) \ ". In addition, we also need to use a small variable mat to obtain the part that matches the expression.

We need the regex_dearch() function to perform search matching, setting the start and end as the beginning and end of the HTML, and continuously searching through a while loop before putting it into the MAT. In the loop, we will convert the found src into a string variable through mat [1]. First and mat [1]. Second, and pass it as a parameter to the function responsible for downloading.

After the first function call responsible for downloading, we set mat [0]. second as the start, which means setting the beginning of the second part that matches the expression found as the start. The next continuous loop will pass all src that meet the table inversion criteria into the download function for downloading. After the download is complete, setting mat [0]. second as the starting step will narrow the search range of each loop until there are no src that meet the criteria in the search range. At this point, all images will be downloaded. We crawled all the image information of the webpage.

5. DOWNLOAD

After obtaining the src, pass it to the function responsible for downloading, but some src cannot be directly downloaded. We need to complete the src as a URL before proceeding with further operations. The URL needs to have a protocol name and a host name, but some SRCs are missing either or both of these parts.



```
char* pos2 = strstr(szurl, "http://");
char* pos3 = strstr(szurl, ".com");
char* pos4 = strstr(szurl, "data-original=");
if (pos4 != NULL)
{
    char t_url[256],y_url[256];
    int a = sscanf(szurl, "%[^=]%s", y_url,t_url);
    char* rt_url = t_url + 2;
    int b = sizeof(rt_url);
    string t_str(rt_url);
    url = t_str;
}
if (pos2 == NULL)
{
    if (pos3 == NULL)
        url = "http://" + str + url;
    else
    {
        if (szurl[0] == '/')
        {
            url = "http:" + url;
        }
        else
            url = "http://" + url;
    }
}
```

Figure 2: Several situations that need to be completed

In addition, when we obtain the src, we will also obtain the src with data original at the end, for example: "src="//www.yesky. com/TLimages2009/yesky/images/delaypic. gif "data original=" d-pic-image. yesky. com/105 x140/uploadImage/2019/007/05/R09WX0PGB9N_H.jpg ". In this case, we need to take the part after data original=as src to download normally, otherwise the download will fail. If we find data original in the obtained src through the strstr() function, we use the sscanf() function to obtain the part after data original=for further processing.

Some of the SRCs we encountered next have the following situations: first, there is no "http://" or host name; second, there is no "http://" but a host name; and third, there is a host name without "http://" but with the following "//". We will use the strstr() function to identify these situations and complete src accordingly based on the judgment results, so that it becomes a URL that can download images normally.

Subsequently, we will use the URLDownToFile() function to download images, but this function requires the storage path and name of the image, and when the downloaded images have the same name, they will overlap with each other. Therefore, we need to use image storage names that are as unique as possible. I am using the substr() function to extract the image name from src to solve this problem. Firstly, I use the find_1ast_of() function to determine the location of the last "/", and then pass the location to the substr() function. This function will automatically extract the content after the location and receive the extracted part as the save name of the image through a string type variable. Finally, adding the save name and the path of the previously created folder together becomes the final save path for the image.

The required URL and save path in the URLDownToFile() function require wide character strings, which is not the type of URL and save path we currently use. Therefore, we need to use the function MultiByteToWideChar() to convert these two strings. This function requires the size of the buffer for converting characters as a parameter, but we do not know this value beforehand. Therefore, we need to first set the second to last parameter of the function to empty and the first to last parameter to 0, so that the function's function will return the required size of the buffer for converting characters. Afterwards, we will call the function again, this time the second to last variable of the function will be set to a wide character variable, and the first to last variable will be the size of the buffer obtained previously. Only we will use this method to convert both the URL and storage path into wide character strings.

After the above preparations are completed, we will start using the URLDownToFile() function for downloading. This function has five parameters, the first of which is the interface address of the ActiveX component, which we do not need here. We will set it to empty; The second parameter is the URL, and the third parameter is the storage path; The fourth one is a reserved value, set to 0; The fifth one is the address of the ibindstatuscallback interface, which can be set to empty here. Afterwards, the function will download the image to the specified path based on the given URL and save path. The return value of the function will determine whether the download was successful and return a prompt. By parsing the while loop in HTML and continuously passing the obtained src into the download function for processing and downloading, we can obtain all the images of the webpage.

6. DEEP CRAWLING

The application of neural networks in capturing images on web pages is a new field that has emerged in recent years with the development of artificial intelligence technology[15]. Traditional web image capture methods often rely on fixed rules or templates, making it difficult to cope with complex and ever-changing web page structures and content. Neural networks, especially deep neural networks, provide new solutions to this problem with their powerful feature extraction and pattern recognition capabilities. In web image capture, neural networks can be used to analyze the HTML code and CSS styles of web pages, identifying the positions and attributes of image elements[16]. By training a large number of web page samples, neural networks can learn the common features and distribution patterns of image elements in web pages, thereby more accurately locating and capturing images. In addition, neural networks can also process the content of the image itself for further screening and classification. For example, convolutional neural networks (CNNs) can be used to extract and classify features from captured images, identifying information such as objects, scenes, or text in the images, thereby helping users find the desired image resources more quickly[17][18]. Overall, the application of neural networks in web image capture not only improves the accuracy and efficiency of capture, but also provides users with a more intelligent and personalized image search experience. With the continuous development of neural network technology, we have reason to believe that future web image capture will become more intelligent, efficient, and accurate, providing richer image information resources for various industries and promoting the widespread application and development of artificial intelligence technology[19][20].

The above process is the process of crawling all images within a webpage, but only the webpage where the user inputs the URL can be crawled. When we have a larger demand for images, we hope that the crawler can also crawl images from other related webpages. Therefore, based on this requirement, we can declare a queue and insert a step after obtaining HTML and before parsing SRC to achieve deep crawling of other related web pages.

After obtaining the HTML, we use regular expressions in the same way as searching for src to find the 'ref' in the HTML. The expression content can be in two situations: 'ref=\' (. *? \ \. html) \ "" and 'ref=\' (. *? \ \. shtml) \ ". These are all web pages related to the input URL, but some of them also lack protocol names or host names like src. Here, we still use the method of processing src to complete the missing parts, and then we store the completed URL in a queue. After the content crawling on a page is completed, the program will automatically retrieve the URL from the queue and then perform the above cycle, continuously crawling images.

When there is a high demand for images, this method allows you to crawl down all relevant webpage images with only one URL input, and obtain a large number of images in a short period of time.

7. CONCLUSION

Crawlers can efficiently extract valuable information, reduce technical costs, and improve business efficiency. [3] This article elaborates on the design ideas and implementation process of a web crawler based on C++for the HTTP protocol. The program has a clear structure and clear ideas, making it suitable for beginners who are new to web crawling and C++network programming to learn.

At the same time, the project has a small amount of code and low development costs; It also has a large renovation space. When encountering situations that were not considered in this project, the scope of application of the project can be increased by adding situations that did not appear in the original program.

REFERENCES

[1] Chen, H., Shen, Z., Wang, Y., & Xu, J. (2024). Threat Detection Driven by Artificial Intelligence: Enhancing Cybersecurity with Machine Learning Algorithms.
[2] Xu, Y., Gao, W., Wang, Y., Shan , X., & Lin, Y.-S. (2024). Enhancing user experience and trust in advanced LLM-based conversational agents. Computing and Artificial Intelligence, 2(2), 1467. https://doi.org/10.59400/cai.v2i2.1467
[3] Teller, & Virginia. (2000). Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition daniel jurafsky and james h. martin (university of colorado, boulder) upper saddle river, nj: prentice hall (prentice hall ser. Computational Linguistics, 26(4), 638-641.
[4] He, C., Yu, B., Liu, M., Guo, L., Tian, L., & Huang, J. (2024). Utilizing Large Language Models to Illustrate Constraints for Construction Planning. Buildings, 14(8), 2511. https://doi.org/https://doi.org/10.3390/buildings14082511
[5] Tian, Q., Wang, Z., Cui, X. Improved Unet brain tumor image segmentation based on GSConv module and ECA attention mechanism. arXiv preprint arXiv:2409.13626.
[6] Ren, Z. (2024). Semantic Transformation Network: Improving Dialogue Summarization Through Contrastive Learning and Attention. Journal of Theory and Practice in Engineering and Technology, 1(3), 1 – 8. Retrieved from https://woodyinternational.com/index.php/jtpet/article/view/59
[7] Wang, Z., Chu, Z. C., Chen, M., Zhang, Y., & Yang, R. (2024). An Asynchronous LLM Architecture for Event Stream Analysis with Cameras. Social Science Journal for Advanced Research, 4(5), 10-17.
[8] Tennant, & Harry. (1981). Natural language processing: an introduction to an emerging technology.
[9] Zheng, H., Wang, B., Xiao, M., Qin, H., Wu, Z., & Tan, L. (2024). Adaptive Friction in Deep Learning: Enhancing Optimizers with Sigmoid and Tanh Function. arXiv preprint arXiv:2408.11839.
[10] Ren, Z. (2024). Adaptive Multi-Scale Fusion for Infrared and Visible Object Detection in YOLOv8. Journal of Theory and Practice of Engineering Science, 4(09), 28 – 34. https://doi.org/10.53469/jtpes.2024.04(09).04
[11] Shen, Z., Ma, Y., & Shen, J. (2024). A Dynamic Resource Allocation Strategy for Cloud-Native Applications Leveraging Markov Properties. International Journal of Advance in Applied Science Research, 3, 99-107.
[12] Li, L., Gan, Y., Bi, S., & Fu, H. (2024). Substantive or strategic? Unveiling the green innovation effects of pilot policy promoting the integration of technology and finance. International Review of Financial Analysis, 103781.
[13] Xu Y, Shan X, Guo M, Gao W, Lin Y-S. Design and Application of Experience Management Tools from the Perspective of Customer Perceived Value: A Study on the Electric Vehicle Market. World Electric Vehicle Journal. 2024; 15(8):378. https://doi.org/10.3390/wevj15080378
[14] Xie, Y., Li, Z., Yin, Y., Wei, Z., Xu, G., & Luo, Y. (2024). Advancing Legal Citation Text Classification A Conv1D-Based Approach for Multi-Class Classification. Journal of Theory and Practice of Engineering Science, 4(02), 15 – 22. https://doi.org/10.53469/jtpes.2024.04(02).03
[15] Bethard, S. , Jurafsky, D. , & Martin, J. H. . (2008). Instructor's Solution Manual for Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (Second Edition).
[16] Jurafsky, D. , & Martin, J. H. . (2007). Speech and language processing: an introduction to speech recognition, computational linguistics and natural language processing. Prentice Hall PTR.

[17] Yao, J. (2024). The Impact of Large Interest Rate Differentials between China and the US bn the Role of Chinese Monetary Policy -- Based on Data Model Analysis. Frontiers in Economics and Management, 5(8), 243-251.
[18] Yao, J. (2024). The Impact of Large Interest Rate Differentials between China and the US bn the Role of Chinese Monetary Policy -- Based on Data Model Analysis. Frontiers in Economics and Management, 5(8), 243-251.
[19] Wang, Z., Zhu, Y., Chen, M., Liu, M., & Qin, W. (2024). Llm connection graphs for global feature extraction in point cloud analysis. Applied Science and Biotechnology Journal for Advanced Research, 3(4), 10-16.
[20] Nadkarni, P. M. , Ohno-Machado, L. , & Chapman, W. W. . (2011). Natural language processing: an introduction. Journal of the American Medical Informatics Association Jamia, 18(5), 544.