# Implementation of data parsing technology using neural network and web driver

*Pavel* Egarmin[1], *Roman* Panov[1], *Farid* Akhmatshin[1], *Sergey* Mikitchak[1], and *Alena* Egarmina[2]

[1] Siberian University of Science and Technology, 660037, Krasnoyarsk, Russia
[2] Siberian Federal University, 660041, Krasnoyarsk, Russia

**Abstract.** As a rule, data parsing is used to quickly obtain information from various web resources for further study and use. For parsing, you can use both specialized online services and desktop applications. Unfortunately, existing parsing technologies have some limitations. For example, it is often difficult to parse dynamic web pages and classify information obtained through parsing. New approaches are needed in implementing data collection and analysis - using language models and software (web driver) that simulate human actions when working with websites. The web driver assists in accessing data from dynamically updated sites, while artificial intelligence technologies help correctly recognize and classify data. This technology can be used to create parsers for real estate agencies, employment services, university admission committees, advertising campaigns, and financial organizations.

## 1 Introduction

Currently, collecting and analyzing data from web pages is part of many tasks in the fields of business (modeling and forecasting financial indicators) [1-2], sociology (studying public opinion) [3], marketing (promoting and advertising goods and services) [4].

Practice has shown that classical parsing technologies, based on a detailed analysis of the structure of a web page, cannot be used for the following purposes [5]:

– parsing websites with dynamically updated elements. Dynamic pages are the basis of any website. Large clusters of data are "invisible" to the parser until a certain event occurs: a mouse click, using the page scroll bar;

– structuring information. As a rule, existing parsing technologies are oriented towards pre-defined data representation structures, which reduces their effectiveness in subsequent data classification.

Creating a parser based on artificial intelligence technology and special software will help:

– ensure access to information initially hidden on dynamically updated websites (filling out forms, clicking on elements, scrolling the page), thus providing access to data that is not available through a regular HTTP request;

– to perform processing of data with complex structure. Parsing combined with the capabilities of neural networks [6-7] can be used for image retrieval, deep analysis of texts written in natural language. Neural network can help in extracting meaningful features of the

collected data.

## 2 Materials and Methods

To create a data parser, the following tools were used: WebDriverIO [8], the object-oriented language C Sharp, and LLaMA neural network [9-10]. Table 1 shows the characteristics of the software used.

WebDriverIO: a custom implementation of Selenium's WebDriver protocol for Node.js, which can be used with C Sharp. WebDriverIO has the following functionality:

- tools for working with dynamically updated pages;
- compatibility with all popular web browsers;
- convenient Application Programming Interface;
- compatibility with popular programming languages and development systems.

LLaMA is an open language model created in 2023, trained on over 14 trillion tokens of natural language. The main features of LLaMA include:

- the model is built on deep learning and, in terms of its characteristics, is on par with other popular language models, including ChatGPT. It can be used for deep data analysis with subsequent classification;
- supports over 20 languages, has a transformer architecture, and is constantly evolving.

To implement the parser, it is recommended to use the C#, which has the following advantages:

- it supports working with WebDriverIO;
- it has a wide range of libraries for implementing work with neural networks.

**Table 1.** Software for implementing a parser

| Software | License | Features |
|---|---|---|
| WebDriverIO | MIT License | Compatibility with C#, working with popular browsers |
| LLaMA neural network | Llama 2 Community License Agreement | Data structuring |
| C Sharp | MIT License | Support for neural networks, .NET ecosystem |

## 3 Results and Discussion

The subject area chosen for implementing the parser was a real estate organization specializing in apartment sales. The following functionality of the software product was implemented:

- searching for information in advertisements posted on websites or in social network communities, based on the following categories (sale or rent, type of renovation), by the advertisement topic;
- saving data in the form of files with the following structure: detailed description of the advertisement, images attached to the advertisement;
- determining advertisement characteristics: type of housing, number of rooms, presence of images;
- grouping of files by key characteristics of the ad.

Figure 1 shows the algorithm of parser operation.

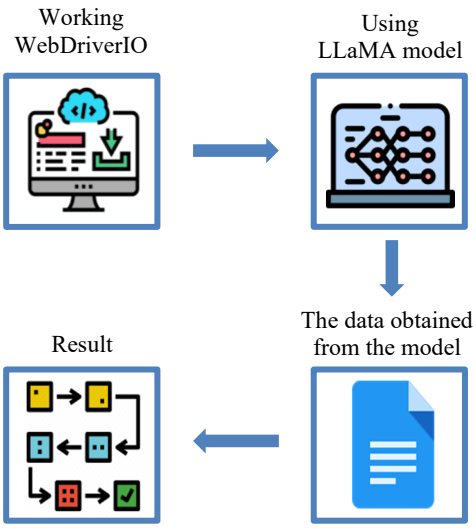


**Fig. 1.** Implemented technology.

First, the parser addresses the selected page of the site, then using the web driver it simulates the work of a person: pressing buttons, scrolling pages, selecting elements, following hyperlinks. Further, having gained access to dynamic data, the parser collects information (Fig. 2). Access to HTML code occurs using CSS and XPath technology. An example of the parser's operation when scrolling a page is shown in Figure 3.

```
// initializing list of components with text using web river variable
var topicElements = _driver.FindElements(By.ClassName("media-
text_cnt_tx"));
foreach (var topicElement in topicElements)
{
   // creating a list of links to ads on social networks
   IWebElement linkElement =
topicElement.FindElement(By.ClassName("media-text_a"));
   topicUrls.Add(linkElement.GetAttribute("href"));
}
```

**Fig. 2.** A code fragment extracting the textual content of an advertisement.

```
private void ScrollToTheBottom()
{
// Creating an object to execute JavaScript using a web driver
    IJavaScriptExecutor jsExecutor = (IJavaScriptExecutor)_driver;
// Saving the page scroll height value
    long scrollHeight = (long)jsExecutor.ExecuteScript("return
Math.max(document.documentElement.scrollHeight,
document.body.scrollHeight);");
// Saving the page scroll height value
    while (true)
    {
        jsExecutor.ExecuteScript("window.scrollTo(0,
document.documentElement.scrollHeight);");
// Updating the scroll height
        Thread.Sleep(2000);
        long newScrollHeight = (long)jsExecutor.ExecuteScript("return
Math.max(document.documentElement.scrollHeight,
document.body.scrollHeight);");
// Checking to reach the end of the page
        if (newScrollHeight == scrollHeight) break;
        scrollHeight = newScrollHeight;
    }
    Thread.Sleep(2000);
}
```

**Fig. 3.** Method containing the logic of the web driver when listing a page.

At the next step, the parser addresses the language model. Its task is to structure the obtained data into categories, in our example these are: the presence of repairs in the apartment, cost, floor, price, material of the house. The data received by the neural network has the following paired structure: «initial HTML code» – «assumed information». The model analyzes the uploaded HTML code or the result of primary parsing and finds the relevant elements to be extracted (Figure 4).

```
// Creating a list of identifiers based on regular expressions
private List<Tuple<string, string>> typePatterns = new List<Tuple<string,
string>>()
{
    Tuple.Create("_Cottage",@"[Cc]ottage"),
private string ExtractType(string text)
{
    for (int i = 0; i < typePatterns.Count; i++)
    {
// Checking whether a line of text matches a regular expression
        Match match = Regex.Match(text, typePatterns[i].Item2,
RegexOptions.IgnoreCase);
        if (match.Success)
        {
            string result = typePatterns[i].Item1;
            if (result.Contains('_'))
            {
                result = result.Replace("_", "");
                return result;
            }
            else return "Detected";
        }
    }
    return "Not detected";
}
```

**Fig. 4.** Code fragment for determining the type of housing in an advertisement.

Figure 5 shows a code fragment demonstrating the operation of the parser in structuring data: the language model sets pairs of tags; the parser reads them and classifies them according to the specified categories. At the same time, both text and images are subject to classification.

```
// Connecting a neural network
var RequestData = new Request()
{
    ModelId = "gpt-3.5-turbo",
    Messages = messages
};
// Executing an HTTP POST request
using var response = await httpClient.PostAsJsonAsync(endpoint,
requestData);
// Reading the response from the neural network in JSON format
ResponseData? responseData = await
response.Content.ReadFromJsonAsync<ResponseData>();
// Determining property values Housing type and Number of rooms
// Return of the Ad object
public Advertisement CreateAdvertisement(string neuroformattedData)
{
    ad.HousingType = ExtractType(neuroformattedData);
    ad.RoomCount = ExtractType(neuroformattedData); return ad;
}
```

**Fig. 5.** Structuring data.

Thus, parsing data using artificial intelligence technology and a web driver allows obtaining data not only from static but also from dynamic web pages. In the given example, the parser analyzes real estate sale announcements posted in social network communities, and using a language model, classifies the obtained information according to specified characteristics. The presented technology can be used in other subject areas.

## 4 Conclusion

The implemented technology has advantages over traditional parsing methods. Firstly, it allows access to data located on dynamically complex websites. Second, the use of neural networks allows achieving high accuracy when processing and classifying large amounts of data.

However, there are some limitations in doing so. Neural network models integrated into client applications are often paid, require constant training and updating. The use of web drivers can slow down the parsing process due to the need to fully display and interact with the web page.

## Acknowledgements

The parsing technology was implemented as part of the grant support of the Krasnoyarsk Regional Fund of Science.

## References

1. Min Choi, Hye Jin Lee, Soh Hyung Park, Sung Whan Jeon, Sungzoon Cho. Stock price momentum modeling using social media data // Expert Systems with Applications. 2024. Vol. **237**. DOI: 10.1016/j.eswa.2023.121589

2. Abbas Yazdinejad, Ali Dehghantanha, Hadis Karimipour, Gautam Srivastava, Reza M. Parizi. *An efficient packet parser architecture for software-defined 5G networks* // Physical Communication. 2022. Vol. **53** DOI: 10.1016/j.phycom.2022.101677
3. Raj Agnihotri, Khashayar Afshar Bakeshloo, Sudha Mani. *Social media analytics for business-to-business marketing* // Industrial Marketing Management. 2023. Vol. **115**. DOI: 10.1016/j.indmarman.2023.09.012
4. Igor Dejanovic. *Parglare: A LR/GLR parser for Python* // Science of Computer Programming. 2021. Vol. **214**. DOI: 10.1016/j.scico.2021.102734
5. Angelo Borsotti, Luca Breveglieri, Stefano Crespi Reghizzi, Angelo C. Morzenti. *General parsing with regular expression matching* // Journal of Computer Languages. 2022. Vol. **2.2.** DOI: 10.1016/j.cola.2022.101176
6. Ande Quintiliano Bezerra Silva, Wesley Nunes Gonçalves, Edson Takashi Matsubara. *DESCINet: A hierarchical deep convolutional neural network with skip connection for long time series forecasting* // Expert Systems with Applications: An International Journal. 2023. Vol. **228**. DOI: 10.1016/j.eswa.2023.120246
7. Mengtian Yin, Llewellyn Tang, Chris Webster, Jinyang Li, Haotian Li, Zhuoquan Wu, Reynold C.K. Cheng. *Two-stage Text-to-BIMQL semantic parsing for building information model extraction using graph neural networks* // Automation in Construction. 2023. Vol. **152**. DOI: 10.1016/j.autcon.2023.104902
8. Boni Garcia, Carlos Delgado Kloos, Carlos Alario-Hoyos, Mario Munoz-Organero. *Selenium-Jupiter: A JUnit 5 extension for Selenium WebDriver* // Journal of Systems and Software. 2022. Vol. **189**. DOI: 10.1016/j.jss.2022.111298
9. Ruopeng An, Yuyi Yang, Fan Yang, Shanshan Wang Mlwa. *Use prompt to differentiate text generated by ChatGPT and humans* // Machine Learning with Applications. 2023. Vol. **14**. DOI: 10.1016/j.mlwa.2023.100497
10. Biao Zhao, Weigiang Jin, Javier Del Ser, Guang Yang Neucom. *Exploring potentials of ChatGPT on cross-linguistic agricultural text classification* // Neurocomputing. 2023. Vol. **557**. DOI: 10.1016/j.neucom.2023.126708