The Data Extraction Using Distributed Crawler Inside the Multi-Agent System

Karel TOMALA[1], *Jan PLUCAR*[2], *Patrik DUBEC*[2], *Lukas RAPANT*[3], *Miroslav VOZNAK*[1]

[1]Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB–Technical University of Ostrava, 17. listopadu, 708 33 Ostrava-Poruba, Czech Republic
[2]Department of Computer Science, Faculty of Electrical Engineering and Computer Science, VSB–Technical University of Ostrava, 17. listopadu, 708 33 Ostrava-Poruba, Czech Republic
[3]Department of Applied Mathematics, Faculty of Electrical Engineering and Computer Science, VSB–Technical University of Ostrava, 17. listopadu, 708 33 Ostrava-Poruba, Czech Republic

karel.tomala@vsb.cz, jan.plucar@vsb.cz, patrik.dubec@vsb.cz, lukas.rapant@vsb.cz, miroslav.voznak@vsb.cz

Abstract. *The paper discusses the use of web crawler technology. We created an application based on standard web crawler. Our application is determined for data extraction. Primarily, the application was designed to extract data using keywords from a social network Twitter. First, we created a standard crawler, which went through a predefined list of URLs and gradually download page content of each of the URLs. Page content was then parsed and important text and metadata were stored in a database. Recently, the application was modified in to the form of the multi-agent system. The system was developed in the C# language, which is used to create web applications and sites etc. Obtained data was evaluated graphically. The system was created within Indect project at the VSB-Technical University of Ostrava.*

Keywords

Class diagram, multi-agent system, Twitter, web crawler.

1. Introduction

We have faced the problem of data mining from social networks, such as Twitter. Data mining is the methodology of obtaining non trivial hidden and potentially useful information from data. It is used in the commercial sector and scientific research, but also in other areas. In our case, we have used the data mining methods to extract keywords from content downloaded by web crawlers [1]. Social network has great potential for obtaining data and information about relationships between groups of people [2].

1.1. Web Crawler

A Web crawler (also known as a web spider or web robot) is a computer program or automated script which browses the World Wide Web in a methodical, automated manner or in an orderly fashion. This process is called Web crawling or spidering [3].

Users are browsing websites through a series of links from one page to another. This activity can be simulated and performed by robots. Browsing the code of web pages, gathering the information found in the code and search links to other websites is the most common task of robots. In principle, this type of robots are divided into two groups according to the size of the search area:

- Robots which browse websites on the pre-specified domain or a finite set of several domains.
- Robots browsing across large environment of WWW.

To start browsing website the robot needs the initial URL or a list of URLs (seed). Then it starts to browse the web pages from the given URLs and searches links leading to other sites (crawl frontier), which will crawler need to visit. This procedure is then repeated recursively. It also shows that the robot can visit only those web pages that can be accessed by following links leading from the initial web page. Some robots can simultaneously visit multiple web pages and browse them in parallel. Other robots move to the next web page after processing the current web page [4]. Architecture of crawler is shown in Fig. 1.

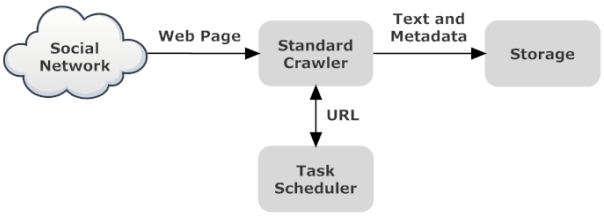

Fig. 1: Architecture of crawler.

1.2. Multi-Agent System

Multi-agent system (MAS) is a simulated environment with the network character, in which there is interaction between certain types of actors (agents) to each other and / or with the environment in which they are located (Fig. 2). These agents collectively solve problems that go beyond the capabilities and skills of each of them [4].

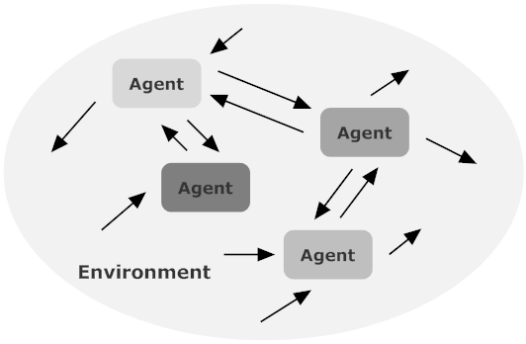

Fig. 2: Multi-agent system.

2. Used Technology and Methodology

First, we have created a standard crawler, which browsed through a predefined list of URLs and gradually download page content of each of the URLs. Page content is then parsed and important text and metadata are stored in a database.

Crawler, demonstrated in Fig. 1 was working properly, but in terms of the volume of downloaded data was not sufficient. It was necessary to run the crawler on multiple machines and somehow coordinate these instances of crawler. We have decided to create a multi-agent system, which would provide support for the creation of agents and the organization of work between them. Figure 3 demonstrated the top level of the multi-agent system. This is a hierarchical structure in which two types of agents exist:

- Master agent: this agent creates new agents, maintains a database of addresses that are to be crawled, and distributes tasks to individual agents.
- Agent: this agent contains a module that is responsible for downloading the web page content.

2.1. Distributed Web Crawling

Distributed web crawling is a distributed computing technology. Distributed web crawling is a basic element for any of the decentralized search applications. The web content collected by a distributed crawler can be indexed by decentralized search infrastructures, or archived using a permanent storage infrastructure. The distributed crawler uses the excess bandwidth and computing resources of clients to crawl the websites. Such systems may allow for users to offer their own computing and bandwidth resources for crawling web pages. Distributing the load of these tasks across many computers saves the cost that would otherwise be spent on maintaining large computing clusters [5].

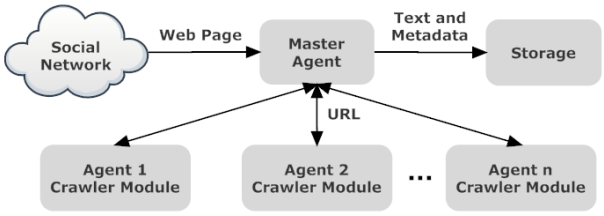

Fig. 3: Distributed web crawling.

The query is executed in a distributed manner as follows. Each crawler node starts the query engine and is responsible for crawling of different web pages. At first, the query is sent to all the crawler agents, and set to run according to exit criterion (crawling time, depth of crawl etc.). The crawl is started by publishing a set of URLs to each crawler, which is then responsible for downloading data from this given set. Once the crawler finishes crawling, it contacts manager crawler that provides more URLs to crawl [6].

2.2. Twitter Search API

The Twitter Search API is used for running searches against the real-time index of recent Tweets. There are several important facts that you need to know before using the Twitter Search API [7]. Limitations:

- The Search API is not complete index of all Tweets, but is only an index of recent Tweets. Index includes between 6-9 days of Tweets.
- Complex queries can be limited and Search API will respond to such query with the error: "er-

ror":"Sorry, your query is too complex. Please reduce complexity and try again.".

- Search does not support authentication in which case all queries are anonymous.
- Search is focused in significance and not completeness. This means that some Tweets may be missing from search results. If you want to match for completeness you can use the Streaming API instead.
- The near operator cannot be used by the Search API. You need to use the geocode parameter.
- Queries are limited to 1000 characters in length.
- When performing geo-based searches with a radius, only one thousand distinct subregions will be considered when evaluating the query.

The Rate Limits for the Search API are other than for the REST API. Using the Search API you are not limited to a certain number of API requests per hour, but instead by the complexity and frequency. As requests to the Search API are anonymous, the rate limit is determined against the requesting client.

In order to prevent abuse the rate limit for Search API is not published. Should the rate limit is restricted. The Search API will respond with an HTTP 420 Error. "error":"You have been rate limited. Enhance your calm.". Sample result:


```
{
"created_at": "Tue, 15 Nov 2011 20:08:17
+0000",
"from_user": "fakekurrik",
"from_user_id": 370773112,
"from_user_id_str": "370773112",
"from_user_name": "fakekurrik",
"geo": null,
"id": 136536013832069120,
"id_str": "136536013832069120",
"iso_language_code": "en",
"metadata": {
"result_type": "recent"
},
"profile_image_url":
"http://a1.twimg.com/profile_images//
phatkicks_normal.jpg",
"source": "&lt;a href=&quot;http://
twitter.com/&quot;&gt;web&lt;/a&gt;",
"text": "@twitterapi, keep on keeping it
real",
"to_user": "twitterapi",
"to_user_id": 6253282,
"to_user_id_str": "6253282",
"to_user_name": "Twitter API"
}
```


3. Web Crawler Implementation

Web crawler itself is started within every agent instance. Multi-agent system is able to encapsulate any application that needs to be run inside the multi-agent system. This has been accomplished by following FIPA standards [8].

There are several types of the multi-agent system architectures. In our case, the most suitable architecture is the hierarchical one, in which the hierarchy of agent's roles is the most essential element. Central management element is also part of this architecture, but it does not perform all tasks itself. It may delegate part of communication and management tasks to the control elements in lower levels of the hierarchy. Such architecture can be represented in the form of a tree (Fig. 4). Leaves of the tree represent discrete and finite agents. The advantage of this architecture is scalability and robustness. Adding additional control elements supports load balancing management. New element can be simply added as a child of its parent agent and there is no need to change the implementation of the system.

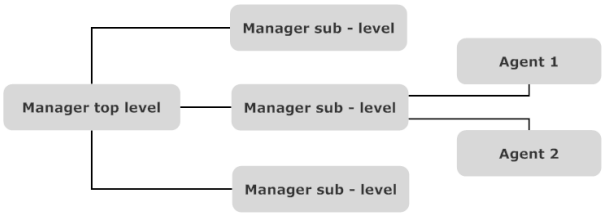

Fig. 4: Architecture of system.

Our MAS implementation offers two solutions: distributed implementation (agents distributed throughout network) or undistributed (MAS is started on one local work station).

In distributed version, every agent starts and runs both TCP client and TCP server in order to be able to receive and send messages. Both TCP server and TCP client of each agent are launched in their own thread so that running them does not oppress the main computational thread of the agent.

Message object is a communication element that uses XML format for the socket communication. The Message consists of Header and Content. The Header contains information about a sender and a receiver (IP address, port, etc.). The Content contains specific information which is the subject of the communication. A message can be encrypted before it is sent and decrypted when received. The MD5 algorithm is used for this encryption and it may as well be replaced by any other encryption algorithm. Finally, the message is converted into a binary form and sent.

Multi-agent system is running set of task simultaneously. Workflow of the system is depicted in the Fig. 5 and description below. The basic concept is to crawl URLs, download content of the URLs and analyze it using data mining processor. New set of URLs is created from this analysis. Example given from testing during Olympics in London 2012:

- Initial set of URLs contained search phrase: "Summer Olympics 2012".
- Suggested keywords for new search were: "London, Traffic jam, Accommodation, Tickets, etc.".

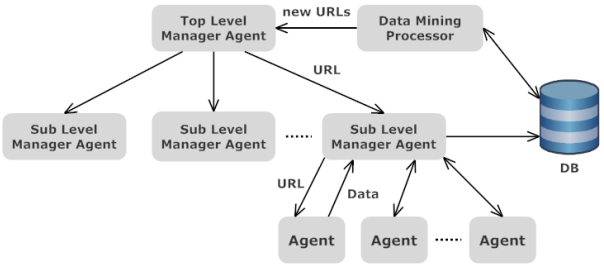

Fig. 5: System concept overview.

- Top level manager agent is started with the list of initial URLs. These URLs are divided as a task to sub level manager agents.
- Sub level manager agents create a number of agents to crawl Twitter social network.
- Every agent downloads target URL content and sends it to the manager.
- Manager saves content into the DB and informs top level manager about task completion.
- Content stored in DB is continuously passed and analyzed by data mining processor.
- Data mining processor creates a list of important keywords that shall be crawled for. This list is passed to top level manager.
- Top level manager agent divides task between sub level manager agents. New agents are created if necessary.

4. Results

In order to test system efficiency, we have set up series of tests that have been executed during summer Olympics in London 2012. Please note that results are measured just for one sublevel segment: one sublevel manager agent and set of agents belonging under this manager. To scale the system and balance the load, we can let the system create hundreds of these segments. Before we do so, we need to find out optimal size of the segment - meaning the number of agents working in one segment.

Figure 6 shows the dependence between the volume of downloaded data and the number of running crawler instances. It is natural to expect that higher number of agents will be able to process higher number of requests.

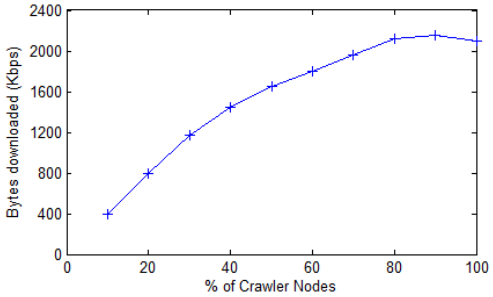

Fig. 6: Crawling load.

However, according to our experiments we have discovered that running about 30 – 40 crawlers is lowering the number of request that single crawler processes. This is caused by manager agent not being able to handle all requests. These requests are divided between inbound and outbound, inbound being data returned from crawler and outbound being URLs to be crawled. This phenomenon could be observed when running about 90 crawlers, where manager agent is overwhelmed with inbound requests and is not able to distribute new URLs to be crawled.

We were looking for a way to increase the number of agents which be fully served by manager. Critical point was communication with the database. Therefore, we focused on the performance of the database layer, which has been programmed using ADO.NET, Entity framework and LINQ to SQL. These three approaches were tested and compared mutually and results are described in Tab. 1.

Tab. 1: Non-transactional insertion - results.

	Time (ms)	CPU Time (ms)	Memory (B)
LINQ to SQL	52333	46819	2970942
Entity Framework	23091	12425	619352
ADO.NET	14333	7736	4418

Database layer using ADO.NET excelled in the means of completion time. When connecting to SQL Server, BULK INSERT method was added for quick data insertion. BULK INSERT method uses specific properties of the database [9]. Due to this implementation ADO.NET layer greatly exceeded the speed of Link to SQL and Entity Framework. Use of ADO.NET

layer is suitable for applications with low requirements of abstraction and high requirements of performance. This could be for example a data pumps, import and export modules. The absence of a full conceptual layer is balanced by the speed of ADO.NET layer. LINQ to SQL returned worst results in terms of memory complexity and processing time. The results are caused by a complicated layering of LINQ to SQL. The LINQ to SQL seeks to provide a conceptual layer.

5. Conclusion

Contribution of this work lays in the creation of an open-source tool that will be usable in the scientific sphere. We have demonstrated the applicability of multi-agent approach enabling distributed crawling. Based on measurements at the laboratory of VSB we have tested and subsequently performed optimization of tool for downloading data. The obtained results show that the tool manages to download large amounts of data (we have to take into account that the data size depends on the size and number of tweets on Twitter). During the testing, we have found that the optimal number of concurrent crawlers varies between 30 and 40. Using more than 40 crawlers, relative performance per crawler is decreased. This is due to the fact that the manager can't handle processing requests for saving data into the database and simultaneous assignment of new tasks. This phenomenon is evident from Figure 5. When running 90 plus crawlers system loses total efficiency. Thanks to this test, we have set the maximum number of crawlers per segment to 65. If more than 65 crawlers are needed, another segment is automatically created and tasks are balanced between old and new segments.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 218086. This work was supported by the Grant Agency of the Czech Republic - GACR P103/13/08195S and project, reg. no. CZ.1.07/2.3.00/20.0072.

References

[1] LIU, B. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications*. Springer, 2011. ISBN 978-3642194597.

[2] RUSSELL, M. A. *Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites*. O'Reilly Media, 2011. ISBN 978-1449388348.

[3] MARMANIS, H. and D. BABENKO. *Algorithms of the Intelligent Web*. New York: Manning Publications, 2009. ISBN 978-1933988665.

[4] SCHRENK, Michael. *Webbots, Spiders, and Screen Scrapers: A Guide to Developing Internet Agents with PHP/CURL*. San Francisco: No Starch Press, 2012. ISBN 978-1449388348.

[5] BEER, M., M. FASLI and D. RICHARDS. *Multi-Agent Systems for Education and Interactive Entertainment: Design, Use and Experience*. Hershey: IGI Global, 2010. ISBN 978-1609600808.

[6] SHKAPENYUK, V. and T. SUEL. Design and implementation of a high-performance distributed Web crawler. In: *Proceedings 18th International Conference on Data Engineering*. New York: IEEE, 2002, pp. 357–368. ISBN 0-7695-1531-2. DOI: 10.1109/ICDE.2002.994750.

[7] MAKICE, K. *Twitter API: Up and Running: Learn How to Build Applications with the Twitter API*. Sebastopol: O'Reilly Media, 2009. ISBN 978-0596154615.

[8] GOMAA, H. *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. Cambridge: Cambridge University Press, 2011. ISBN 978-0521764148.

[9] MEHTA, V. P. *Pro LINQ Object Relational Mapping in C# (Expert's Voice in .NET)*. New York: Springer, 2008. ISBN 978-1-59059-965-9.

About Authors

Karel TOMALA was born in 1984. In 2007, received a Bachelor title in VSB–Technical University of Ostrava, Faculty of Electronics and Computer Science, Department of Telecommunications. Two years later he received the M.Sc. title focused on Telecommunications in the same workplace. Currently in the doctoral study he focuses on Voice over IP technology and Speech Quality in VoIP.

Jan PLUCAR was born in 1987. In 2011, received a Master title in VSB–Technical University of Ostrava, Faculty of Electronics and Computer Science, Department of Computer science. Jan Plucar is currently working on his Ph.D. in the field of computer security and bio inspired computations.

Patrik DUBEC was born in 1986. In 2011, received a Master title in VSB–Technical University of Ostrava, Faculty of Electronics and Computer Science, Department of Computer science. Patrik Dubec is currently working on his Ph.D. in the field of computer security and data mining methods.

Lukas RAPANT was born in 1986 in Ostrava. In 2009, he received the bachelor's degree in applied mathematics from VSB–Technical University of Ostrava, Faculty of Electronics and Computer Science, Department of Apllied Mathematics. In 2011, he received the master degree in the same field from the same university. Currently, he is undergoing his doctoral study on VSB–Technical University of Ostrava, where he focuses on applied staticstics and graph algorithms.

Miroslav VOZNAK is an associate professor with Department of Telecommunications, VSB–Technical University of Ostrava. He received his M.Sc. and Ph.D. degrees in telecommunications, dissertation thesis "Voice traffic optimization with regard to speech quality in network with VoIP technology" from the Technical University of Ostrava, in 1995 and 2002, respectively. Topics of his research interests are Next Generation Networks, IP telephony, speech quality and network security. He was involved in several FP EU projects. At present, he is also working for the Czech National Centre of Excellence IT4I.