Automatic Search of Web APIs Exposed in the Open Source Context

Cecilia CIOLOCA, Madalina ZURINI, Mihai GEORGESCU
Department of Economic Informatics and Cybernetics
Bucharest University of Economic Studies
ROMANIA
cecilia_cioloca@yahoo.com, madalina.zurini@gmail.com, mihai.georgescu@europe.com

Abstract: The Service Oriented Architecture is presented in the context of organizations today. The Web APIs are being defined. Architecture for Web APIs aggregation is proposed. The components of the architecture are detailed. A method for automatic feature extraction from Web APIs is proposed in order to generate the input data for k Nearest Neighbor search algorithm. The main directions of kNN algorithm are highlighted along with the advantages and disadvantages appeared. A structure for Web APIs is formed out of the features extracted for a homogeneity approach and the output data is used in a hierarchical searching result. An optimization method is proposed for lowering the time consumption given by the high complexity level, using a grid formed out of cell searching algorithm.

Key-Words: Web API, k Nearest Neighbor, open source, automatic search, SOA, grid cell feature space

1. Introduction

The orientation of organizations and users to have access to more diverse information in a short time led to the introduction and promotion of the concept of SOA, Service Oriented Architecture SOA. SOA was designed with the idea of interoperability; it encourages communication between independent components unlike other models of distributed architectures that focuses on the exchange of objects through the network.

Web API evolved from the well-known web service developed using the service oriented architecture (SOA). While SOA uses mostly SOAP-based web services, Web APIs moved towards RESTful web resources. REST-style architecture consists of both clients and servers, where clients initiate requests and servers process the requests and sends back appropriate responses. Usually the protocol used for communicating with Web APIs is http and return the response in XML or JSON formats. A RESTful web API is a web API implementation using HTTP and REST principles which are defined using the following:

- the set of operations supported by the web API;
- the API must be self-descriptive.

A schematic architecture of a Web API is presented in figure 1. It allows communication with any type of client which is able to understand the communication protocol and the type of resources exposed by the web API.

- Web API using REST
- Collection of methods exposed
- Standardized protocol using HTTP
- Transport of information using JSON or XML

![Figure 1. Architecture of Web API](image-url)
Web API offers many advantages including:

- external database access for other systems;
- convenience for the user who can now consume an web API that already address his business problem;
- security protection that is already embedded in the web API;
- seamless integration with other systems.

In order to have increased productivity and quality while developing the web API, the architect has the option to reuse existing open source components. Integrating open source components is easily achieved through interfaces or existing libraries. The security of the Web API or other functionalities are faster developed using open source.

To allow a system to integrate with an existing Web API and reuse its methods, there are two standards for describing a REST service:

- Web Application Description Language (WADL);
- Web Services Description Language (WSDL).

The WADL standard represents an XML description which models the resources provided by a web service and the relationship between them. Each resource contains param elements to describe the input and method element which describes the request and response of a resource [WWW1].

The WSDL is an XML format which uses the following elements in defining the network services [WWW1]:

- Types- a container for data type definitions using some type system (such as XSD);
- Message- an abstract, typed definition of the data being communicated;
- Operation- an abstract description of an action supported by the service;
- Port Type- an abstract set of operations supported by one or more endpoints;
- Binding- a concrete protocol and data format specification for a particular port type;
- Port- a single endpoint defined as a combination of a binding and a network address;
- Service- a collection of related endpoints.

WSDL and WADL provide a way to consume and aggregate the web API. Search algorithms are used to identify the objects exposed by web API and organize each method by its purpose.

In the present paper, a background of the theme is highlighted in introduction, presenting a schematic architecture of WEB API along with the main standards of describing a REST service, WADL and WSDL. Chapter 2 proposes a Web APIs Aggregator Architecture with its components, an architecture that describes the flow of data, the process of collecting Web APIs, ranking them and selecting the best that matches a searching criterion.

Chapter 3 contains the description, the main steps, the advantages and disadvantages of k Nearest Neighbor algorithm used for automatic search. The kNN algorithm is mapped to the Web APIs search, describing the feature space, characteristic extraction and interpretation of the results. A method for parameter adding is proposed for reaching the structured desired more efficient.

Chapter 4 contains an optimized proposed version of k Nearest Neighbor that minimizes time consumption in the context of high complexity. A grid is formed within the feature space using cells that divide the searching space, lowering the number of sorted objects. Conclusions are drawn and future work is related to the implementation of the proposed architecture in the context of real use in organizations today. Also, the optimization process will be taken into account at the implementation level.

2. Web APIs Aggregator Architecture

SOA architecture is composed of web services, autonomous components that are interacting with each other, as a result the two terms SOA and web services often merge. In [1] the web service is defined as a distributed component, well defined, interoperable that is able to respond to the requests initiated by the consumer. In [2] Web
services are defined as modular business applications, self-described, that expose the business logical using web APIs that can be identified and accomplished via IP. The proposed architecture describes a Web APIs aggregator that exposes an interface to be consumed by third parties web applications. The main role of the system is to provide Web APIs aggregator service based on categories and to route the client application requests to the most efficient web service that meets customer needs. Redirection is performed taking into account a scoring indicator calculated in advance based on the response time and the reliability of the service. Figure 1 presents the architecture components and the relationships between them:

C1. Semantic Web API Search Routine - is a job that runs in the background and that searches and identifies in the cloud new Web APIs based on semantic categories. Once identified the Web APIs are being stored in the system database for future evaluation and usage. The stored data contains web service description, URLs used for calling the web API and information about authentication;

C2. Web API Heart Beat Routine - is a component that can be scheduled when to run. This component extracts the web services configuration list from the database and consumes the web services. The role of this process is to validate that the web services are up and running and to store in the database information about the web services performance. It saves information about response time, failures and downtime for each web service;

C3. Scoring calculator routine - this component runs at a scheduled date and time and its role is to calculate the scoring indicator for each web API in the database. The scoring of a Web API is obtained taking into account the response time measured previously and the failure rate of the service. For those services that are not available anymore and have not responded the last 10 consecutive calls the value of the scoring is set to 0. The scoring interval is between 0 and 1, the more closely to 0 the less confident that Web API is and the chances to use the service decease;

C4. Web API Aggregator Server - is the central component of the architecture, because it contains the whole business logic of the system and is involved in the decision making process for the selection of Web APIs that will be used for redirecting the client request. This component acts as the router between all the components of the system and the web services in the cloud. The selection of the service to redirect the client request is performed based on the scoring indicator. The most reliable web services are those with a higher scoring, closer to value 1. If there are multiple services with the same score, the selection will be completely random to ensure the equity of the process;

C5. Aggregator Web API - this component is the interface exposed by the aggregator system and it is consumed by the client applications. By this interface, web applications send information about the services they want to call and input parameters in order to perform the operation. The client applications will receive the response to their request via this interface;

C6. Web APIs Reports and Administration User Interface - this component exposes a user interface for browsing information and reports on Web APIs in the system. In this interface new categories of web APIs can be added and the system will search for those semantics;

C7. Web APIs - are third party components exposed by service providers and located in the cloud. These web services are being used to redirect client requests by the web API aggregator server;

C8. Web API Consumers - are third-party applications that aim to obtain information through integration with web service providers.

The advantages of the proposed architecture are:

A1. Ensuring availability of information based on the existence of several Web APIs that provide similar functionality. If a web service call fails, the user request will be automatically redirected to another similar web API;
A2. Ensuring minimum response time by calculating the scoring taking into account web services response time and reliability; A3. Reduction of time to market regarding integration with desired Web APIs. Client applications integrate with the web API exposed by the aggregator systems and are able to call multiple services from different categories. In case of not using the web API aggregator, consumers are forced to manage multiple web services integration.

Web APIs aggregator architecture is an original solution to simplify for the consumer the complexity brought by the diversity of web APIs and to maximize the efficiency in using web APIs.

3. Automatic search within WEB APIs using k Nearest Neighbor

K Nearest Neighbor, also known in Computational geometry under the name of Closest pair of points problem, is the algorithm used for searching and computing the nearest k neighbors of a given n dimensional object from a set of points a priori known. This algorithm is used for the process of supervised classification or for automatic searching. In the context of supervised classification, a set of multidimensional objects each labeled with the assigned class is used as a knowledge base. In addition, a new object represented in the same multidimensional space is introduced in the
process of classification as input. The K Nearest Neighbor is run with the single input parameter k, the number of nearest objects to be returned, the set of objects and the analyzed new object. Using a distance function, the algorithm returns the k nearest neighbors in terms of minimization of distance function, from the object given as input.

According to the objectives, the objects returned are then computed using an aggregation function based on vote majority.

For the process of automatic search, the k objects returned aren’t computed, but used as output data. Also, the initial set of multidimensional objects is not a priori classified in classes.

The notations in the context of metric space are given by the initial set of multidimensional objects, \( X \), where

\[
X = \{x_1, x_2, \ldots, x_m\}
\] (1)

\( m \) being the cardinality of the set \( X \). Each object is represented by the values of the analyzed features that are considered to be the axes of the feature space.

\[
x_i = (x_{i1}, x_{i2}, \ldots, x_{in}), \forall i \in \{1, 2, \ldots, m\}
\] (2)

where \( x_{ij} \) is the value of the \( j \) feature for the \( i \) object and \( n \) is the total number of analyzed features.

Let \( x' \) be the new \( n \)-dimensional object given as input for which the algorithm returns the closest objects.

Let \( f_d \) be the distance function used for evaluating the closeness between two objects, with \( f_d: \mathbb{R}^n \times \mathbb{R}^n \rightarrow [0, \infty) \). For the distance function, several functions can be used, mentioning the Euclidian distance, Manhattan, Cosine or Canberra.

For the Euclidian distance, the formula used for calculated the degree of similarity or dissimilarity is:

\[
f_d(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}
\] (3)

For Canberra distance, the function used is:

\[
f_d(a, b) = \sum_{i=1}^{n} \frac{|a_i - b_i|}{|a_i| + |b_i|}
\] (4)

Manhattan distance function is:

\[
f_d(a, b) = \sum_{i=1}^{n} |a_i - b_i|
\] (5)

Further, the steps of the Nearest Neighbor algorithm are presented.

S1. For each object from the initial set of object, the \( f_d \) distance function is calculated as distance between each object to the input object \( x' \).

S2. The distances generated as step 1 are sorted using a sorting algorithm.

S3. The first k objects with the lowest values are given as output.

The complexity of the algorithm is \( O(n \times m + m^2) \).

The disadvantages of the algorithm are those relating to the parameter and distance function selection. Also, the complexity of the code can be reduced while reducing the dimension of the feature space, using Principal Component Analysis, PCA that generates a lower dimensional space with uncorrelated features.

In the context of Web APIs search, the searching object \( x' \) is formed out of key words given as input, along with a weight, the higher the weight, the bigger the importance given to that keyword. The weights are real values that correspond to the number of occurrences of the word in the Web APIs description and methods. For that, the form for \( x' \) object is:

\[
x' = (x'_{1}, x'_{2}, \ldots, x'_{m})
\] (7)

where \( x'_{j} \) is the weight given for the \( j \) feature in describing \( x' \) object.

The feature space is formed out of the total keywords found in the Web APIs database. If a keyword is not entered in the searching phase, then the weight associated to that keyword is equal to 0. If the keyword entered is not found in the collection of keywords, than the closest keyword in terms of correlation matching is used, from the correlation matrix of the features.

Different methods for generating keywords within a Web API are available. We mention the number of occurrences as the main way for discovering the vector form of representation.

In figure 3, in [6], describes a proposed overall process WSQBE for service
discovery. By applying the idea of Query-by-example, in [6], it is proposed to make
easier the task of defining a query for
discovering Web services.

4. Optimization proposal for k Nearest Neighbor

Different optimization algorithms are proposed in [4], [5], [7], [8], for lowering
the searching area using searching optimization, representation and
computation. The problem of optimization is done in the context of high-dimensional
space with large point sets.

Given the high complexity of the automatic algorithm search using k Nearest Neighbor, depending on the
dimension of the feature space, \( n \), and the
cardinality of the initial set of objects, \( m \),
a new method is proposed in order to
reduce the number of objects to which the
object \( x' \) is compared to.

Let we consider the initial feature space,
where the objects are represented. Each
value of each feature is computed using
the formula:

\[
g_f(o_f) = o_f / \dim_f + 1 \quad (8)
\]

where \( g_f(o_f) \) is the result of the function
of grid assignment of feature \( f \) of the
object \( o \) and \( \dim_f \) is the dimension of the
feature \( f \), resulting from the formula:

\[
\dim = \frac{\max_{i=1}^m x_i}{l} \quad (9)
\]

with \( l \) being an input value of separation.

After computing the formulas for each
existing object from the \( X \) set of objects,
a vector of assignment of each object to a
grid cell is formed, \( grid_i \), is the cell from
which object \( i \) belongs to. The objects
being assigned, the input object \( x' \) is
allocated to its cell.

The optimized k Nearest Neighbor
algorithm uses only the objects that are
part from the same cell with the cell of \( x' \)
or the objects part from the cells
neighbors to that cell. A vector of visited
cells is initialized to zero, along with the
parameter \( nr_obj \), the number of objects
found within the closest cells to the cell
from which object \( x' \) is part of.

The pseudo code for the proposed
optimized algorithm for automatic search
has the following steps.

S1. Determines the cell from which \( x' \) is
part from, set it be \( g \).
S2. If the cell isn’t visited yet, the number
of objects that are part from the \( g \) cell are
counted.
S3. If the number of objects identified is
lower than \( k \), the search is moved to all
its closest cells.
S4. Steps 2 and 3 are retaken until the number of objects identified is equal or bigger than \( k \).
For the bi-dimensional space, the label of the cells neighbor to the \( g \) cell are \( g-1 \), \( g+1 \), \( g+1+1 \) and \( g+1-1 \).
The exit point from the algorithm is when the number of identified close objects is bigger than \( k \), let it be \( m' \), with \( m' \ll m \).

5. Conclusion and future work

The present paper highlighted the need of automatic search integration in the process of Web APIs usage. K Nearest Neighbor is the algorithm used to generate the closest k Web APIs that matches with the query search. Each Web API is represented by a set of features that is formed out of the set of key words found in each of them. Because of the high computational consumption, an optimized algorithm is proposed in order to return the desired objects in lower time. Time reduction is direct proportional to the decreasing of the sorting objects.
The introduced architecture for ranking Web APIs is providing faster response time and better stability for the user by redirecting the call to the Web API to the one which is top ranked. The work will continue by adding multiple criteria for ranking the Web APIs among which will be included the feedback from the end user, easiness of use and quality of information provided by the indexed Web API. This will also be presented in an easy to use web interface which allows creation of reports and statistics for each indexed Web API.
By including high quality open source components, the system will be developed faster and will offer even more information and security. The result is a system of stable and fast Web APIs based on an accurate search algorithm that helps the user to find and use the information he needs in his applications.

Acknowledgments

This work was cofinanced from the European Social Fund through Sectoral Operational Programme Human Resources Development 2007-2013, project number POSDRU/107/1.5/S/77213 „Ph.D. for a career in interdisciplinary economic research at the European standards”.

References

[WWW1]W3C, Web Services Description Language (WSDL) 1.1, Available at: http://www.w3.org/TR/wsdl
[WWW2]W3C, Web Application Description Language, Available at: http://www.w3.org/Submission/wadl/