information

MDPI

Article

Feature Selection and Recognition Methods for Discovering Physiological and Bioinformatics RESTful Services

Chao Song [1,3], Xinyu Gao [1,2], Yongqian Wang [1,2], Jinhai Li [1,2], Lifeng Fan [1,2], Xiaohuang Qin [1,3], Qiao Zhou [1,3], Zhongyi Wang [1,2,3] and Lan Huang [1,3,*]

1 College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China; songchaodevip@cau.edu.cn (C.S.); gxyatyx@cau.edu.cn (X.G.); cieewyq@cau.edu.cn (Y.W.); lijinhai@cau.edu.cn (J.L.); zgndnd_flfwxj@cau.edu.cn (L.F.); qinxh@cau.edu.cn (X.Q.); bridgezhou@cau.edu.cn (Q.Z.); wzyhl@cau.edu.cn (Z.W.)
2 Key Laboratory of Modern Precision Agriculture System Integration Research, Ministry of Education, Beijing 100083, China
3 Key Laboratory of Agricultural Information Acquisition Technology, Ministry of Agriculture, Beijing 100083, China
* Correspondence: hlan@cau.edu.cn; Tel.: +86-10-62737778

Received: 26 July 2018; Accepted: 28 August 2018; Published: 6 September 2018

Abstract: Many physiology and bioinformatics research institutions and websites have opened their own data analysis services and other related Web services. It is therefore very important to be able to quickly and effectively select and extract features from the Web service pages to learn about and use these services. This facilitates the automatic discovery and recognition of Representational State Transfer or RESTful services. However, this task is still challenging. Following the description feature pattern of a RESTful service, the authors proposed a Feature Pattern Search and Replace (FPSR) method. First, they applied a regular expression to perform a matching lookup. Then, a custom string was used to substitute the relevant feature pattern to avoid the segmentation of its feature pattern and the loss of its feature information during the segmentation process. Experimental results showed in the visualization that FPSR obtained a clearer and more obvious boundary with fewer overlaps than the test without using FPSR, thereby enabling a higher accuracy rate. Therefore, FPSR allowed the authors to extract RESTful service page feature information and achieve better classification results.

Keywords: RESTful services; feature extraction; web page classification; service discovery

1. Introduction

In order to facilitate data sharing and the exchange of analytical methods, and their use by other organizations, many physiological and bioinformatics research institutions and websites have opened their own data analysis services and other related Web services. However, it is still a challenge to find and use these Web services quickly and effectively [1,2]. Unlike traditional Simple Object Access Protocol (SOAP) Web services, it is crucial to effectively extract and select the features of Representational State Transfer or RESTful service pages for the automated identification and discovery of these pages.

Representational State Transfer (REST) is a Web software architecture style that was proposed in 2000 by Dr. Roy Thomas Fielding in his doctoral thesis. After participating in the formulation of the HTTP standard and the URL standard, Fielding put together these two design styles, which he summarized as a new architectural style, namely, the REST architecture style [3]. The purpose of this style was to facilitate the transmission of information between different software programs in

the network (e.g., the Internet). The Web services that conform to the REST design style are called RESTful services.

For the most part, Web services are currently divided into traditional SOAP services and RESTful services. Because of its simple implementation, lightweight quality and extensibility, RESTful services have been widely applied, and have become mainstream in Web service development [4]. Traditional structured SOAP Web services rely on the Web Services Description Language (WSDL) to describe the services. WSDL is a structured language based on XML, and the computer can distinguish and understand its descriptive information [5]. However, the description of RESTful services does not have a unified standard. Many people have extracted many description models [6–8], but these models have not been widely used. The description of RESTful services primarily uses semi-structured HTML documents to describe the functions and interfaces and invokes service methods with natural language [9]. For a computer, the description page of the service is the same as the general document. It makes RESTful service description inundated with massive web pages, and a computer cannot identify RESTful service pages directly, which hinders the automated discovery of RESTful services.

For this paper, the authors proposed the FPSR method to preprocess the RESTful service pages related to physiological and bioinformatics information, and to extract and recognize the features of the RESTful service pages. The method ensured that their features were not disrupted by subsequent segmentation. The experimental analysis showed that this method could effectively improve classification performance. Compared with the traditional method, this method achieved a higher accuracy, precision, recall rate and f-score, thereby extracting RESTful service page features more effectively.

2. Related Work

The recognition of RESTful service pages is a binary classification of web pages. Web classification is different from the traditional plain text classification. Compared to that classification [10], HTML is a semi-structured document that contains information (e.g., labels, images, links, code, and scripts), which makes automatic classification difficult. In recent years, machine learning algorithms, such as Support Vector Machine (SVM), Naive Bayes [11], K-Nearest Neighbors (KNN) [12], and Decision tree [13] have been widely used in web page classification. There are many aspects that are worthy of research and exploration to select and extract features effectively in web pages and use machine learning algorithms for classification.

Kan and Thi [14] extracted feature information from URL and used SVM to carry out fast web page classification. Hu et al. [15] extracted pictures and text information in pages and used multi-kernel learning to classify drug-related websites. Zhao et al. [9] used the Naive Bayes algorithm and SVM to classify RESTful services by extracting features from web page content and document structure information. Rajalakshmi and Xaviar [16] conducted an experimental study on URL-based web page classification feature-weighting techniques and used SVM to perform web page classification. Altay et al. [17] used context-sensitive and keyword density to extract features and used SVM, maximum entropy, and Extreme learning machine for malicious web page detection. Siddiqui et al. [18] used HITEx (Health Information Text Extraction), which is an open-source natural processing software, to extract text features and used SVM and the Naive Bayes algorithm to identify health information web pages. Onan [19] analyzed and compared four kinds of feature selection methods (correlation, consistency, information gain and chi-square-based feature selection) and four different algorithms (Naive Bayes, KNN, C4.5, and FURIA) on the predictive effect of web page classification. Mohamed et al. [13] used Term Occurrence, Term Frequency, and TD-IDF for feature selection and extraction, and used the Naive Bayes, KNN, and Decision tree algorithms for web page classification. Kiziloluk and Ozer [20] selected the HTML pages' tags as features, then used the Parliamentary Optimization Algorithm (POA) for web page classification.

3. Datasets and Tools

3.1. Dependent Libraries

Python and related libraries were used to implement the web crawler program, data processing, feature selection and extraction, model training, and database reading/writing operations. The tools and libraries used are as follows.

Requests: A powerful HTTP Client library for Python. It is very easy to send a custom HTTP request and to retrieve HTML pages using Requests.

Beautiful Soup: A Python library used for parsing HTML or XML, which can parse, query, and modify the document tree. It is convenient to analyze the HTML pages and extract information.

scikit-learn: An excellent machine learning library in Python, which provides many out-of-the-box machine learning algorithms, data preprocessing algorithms, and simple and effective data mining and data analysis tools. Many machine learning algorithms can be easily used through this library.

re: This is a Python module regular expression, which provides regular expression matching operations. Regular expressions use a single string to describe and match the string corresponding to the pattern rules.

pymongo: A Python library for operating MongoDB, providing support for reading, writing, querying, deleting, and other operations of MongoDB.

3.2. Data Acquisition

To obtain datasets for training and testing, the authors wrote a web crawler program using Python's Requests library, and obtained RESTful service pages for biological and physiological information research institutions and organizations, including Kyoto Encyclopedia of Genes and Genomes (KEGG) [21], the European Bioinformatics Institute (EMBL-EBL) [22], German Neuroinformatics Node (G-Node) [23], etc. In addition, to increase the diversity and quantity of datasets, they also retrieved description pages of RESTful services and some other non-RESTful pages from companies such as GitHub, Google, and Facebook. They saved the retrieved URLs and the original web pages to the MongoDB database for subsequent reading and analysis. MongoDB is a NoSQL database which can save the data in JavaScript Object Notation (JSON)-like documents. Because it is schema-free and has scalability, the data storage modal can be easily changed. For saving the file, MongoDB is more suitable than other relational databases.

3.3. Datasets

The size and representativeness of the dataset determined the upper bound of the training model, which played an important role in the training result of the model. The authors obtained 731 RESTful service pages, of which 265 pages were related to physiological information, and 1739 non-RESTful pages, for a total of 2470 samples. The dataset was saved in the local MongoDB database.

4. Implementation Details

4.1. Web Crawler

The authors used the Requests and Beautiful Soup libraries in Python for the implementation of Web Crawler. First, they found all the URLs about the RESTful service using the Beautiful Soup library to parse the directory page of the service. Then, the Requests library was used to send all the HTTP requests for each URL and get all the web pages for each URL, and then each page and URL was saved to MongoDB. The read and write operations of MongoDB required the help of the pymongo library in Python. For some websites, access could not be gained without web proxy. Requests was therefore used to set the proxies argument to access URLs through the parameter setting of proxies.

4.2. Preprocessing of Web Pages

The preprocessing of web pages plays a significant role in the feature extraction and selection and the performance of the classifier.

For this study, the document captured by Web Crawler was in HTML format. HTML is a semi-structured text format, and the tags in a HTML document are used to annotate texts, images, and other contents. JavaScript and Cascading Style Sheets (CSS) can also be used to describe a web page's functionality and presentation, information which was not meaningful for this study. The authors removed it in preprocessing and retained the other information, thereby obtaining the plain text content.

The preprocessing of web pages is shown in Figure 1. During the processing, the Beautiful Soup library was used to finish the HTML parsing, the searching and deleting operation of the target tag, and the extraction and filter of the tag attribute, which ensured efficient processing.

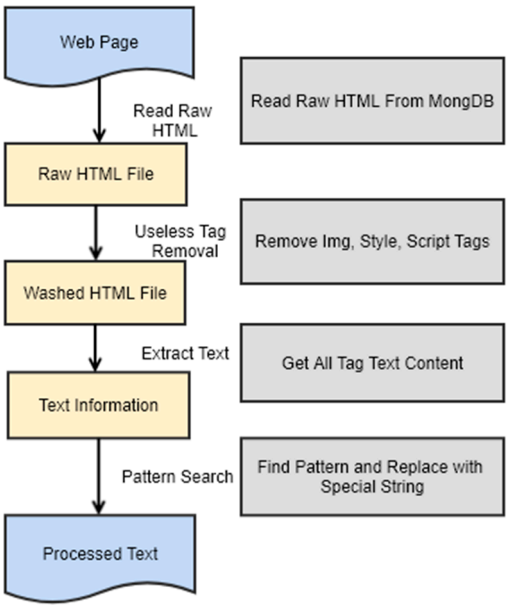

Figure 1. Web page preprocessing.

The web page preprocessing steps are as follows:

Useless tag removal: HTML documents may include some tags such as <img>, <script>, and <style>, which are used to display images, control the behavior of a page, and describe the appearance. These noise or non-sense tags were removed in preprocessing. By using Beautiful Soup (HTML parser), these tags were found by tag name, then removed.

Plain text extraction: The other tags, such as <p>, <title>, <body>, <header>, and many others, include information about the page content description. Beautiful Soup was used to remove the markup information and obtain the plain text.

Feature pattern search and replace: As shown in Figure 2, the authors annotated two types of pattern string in segments from RESTful service pages of KEGG (https://www.kegg.jp/kegg/rest/keggapi.html) and G-Node (http://g-node.github.io/g-node-portal/key_functions/data_api/0.1/api_specification.html). The URL is the base part of the link of a special web service, and the path corresponds to a special web service from a website. Almost all RESTful pages include a similar special pattern. Like a URL, the special path corresponds to a special service and email address. These special patterns in RESTful service pages are more frequent than in other web pages. Depending on different word segmentation methods, however, these feature patterns can be split, or the same pattern can be treated as a different feature. To solve this problem, the authors proposed the Feature Pattern Search

and Replace method. First, they used the regular expression to search and match all the pattern strings. Then, they replaced the same pattern string with a special word. This method helped to avoid the problem of pattern destruction in the tokenizing process and to save the feature pattern information. The details of the feature pattern search and replace are shown in Table 1.

Save the results: Finally, the processed text was saved to MongoDB. Figure 3 is an example of the HTML document preprocessing.

Figure 2. Example of Kyoto Encyclopedia of Genes and Genomes (KEGG) and German Neuroinformatics Node (G-Node) RESTful description.

Table 1. Details of Feature Pattern Search and Replace (FPSR).

Feature Pattern	Regular Expression	Custom String
http://locahost:8080/path1/path2	(https?)://[-A-Za-z0-9+&@#/%?=~_|!:,.;]+ [-A-Za-z0-9+&@#/%=~_|]	httpaddr
username@email.com	([a-zA-Z0-9_.+-]+@[a-zA-RZ0-9-]+.[a-zA-Z]+)	emailaddr
/path1/path2	s(/[w-]+)+/?	pathaddr

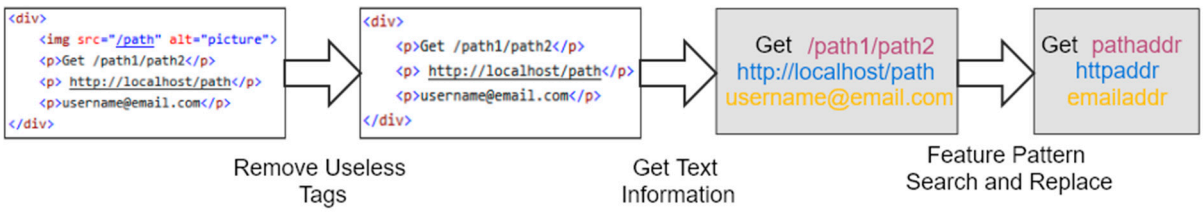

Figure 3. Example of HTML preprocessing.

4.3. Feature Extraction and Selection

BoW (bag of words) is a common document representation method in information retrieval [10]. BoW lists the words with their frequency per document. The frequency of words in a document can be used to represent the document, which allows users to compare documents. BoW ignores the effect of the grammar and the order of words.

Feature extraction is the process of converting raw data to numerical information which a computer can recognize. The computer cannot recognize the raw text information directly. The raw text can be represented by a vector using BoW, followed by a machine learning algorithm used to process the vector.

Feature selection is the process of removing redundant and irrelevant features, that is, to select the most effective feature from the raw features [24]. It can reduce the dimensionality of the datasets and improve the performance of the algorithm. In web page classification, the dimensionality of features is very large as well as the number of samples. Many of the raw features were irrelevant to this study. It is easy to overload the algorithm and create a problem of calculation and memory consumption if feature selection is not performed.

Most machine learning algorithms need input with a fixed length, Because machine learning cannot process the raw text directly. The raw text must be converted to a numerical value with a fixed length, namely, a vector. The document d can be represented as a vector of terms (e.g., $(t_1, t_2, t_3, \cdots, t_i)$), in which t_i is a feature of d.

The process of dictionary initialization is shown in Figure 4:

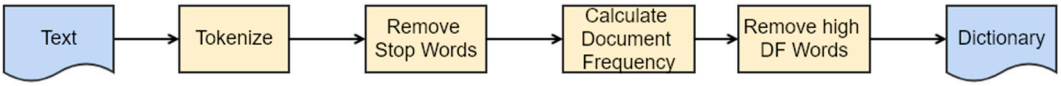

Figure 4. Dictionary initialization.

In the dictionary initialization phase, texts were split by using space and punctuation as delimiters to extract words of at least two letters. The dictionary consisted of the words occurring in documents more than 10 times and in less than 90% of the number of documents. Stop words were removed. The detail of the algorithm is showed in Algorithm 1:

Algorithm 1 The detail process of dictionary initialization

Input: the text T, stop words W_{stop}, the Threshold of DF max_{df}, min_{df}
Output: the set of words W_{dict}
for each text T_i:
lowercase $(T_i) \rightarrow T_i^{low}$
tokenize $(T_i^{low}) \rightarrow W_i$
end
create W_{dict}
for each words array W_i
  W_{dict}+ = set (W_i)
end
$W_{dict} - = W_{stop}$
for each word in W_{dict}, w_i:
  document_frequency $(w_i) \rightarrow df_i$
  if $df_i < min_{df}$ or $df_i > max_{df}$:
    remove w_i from W_{dict}
  endif
end
return W_{dict}

In this process, the authors had to measure the words with their document frequency and the words with their word counts per document. This information was then used to convert a document into a vector.

Finally, term frequency-inverse document frequency (TF-IDF) was used to adjust the weight of words. The more documents the word appeared in, the less valuable that word was as a feature. The words found frequently in all documents were systematically discounted. That left only the frequent and distinctive words as the feature. First, TD-IDF measured the number of times that the words occurred in a given document, and then it adjusted the number by the number of documents in which the words appeared. The formula is as follows:

$$W_{i,j} = tf_{i,j} \times \left[log\left(\frac{N}{df_i}\right) + 1 \right], \quad (1)$$

where tf_{ij} is the number of occurrences of word i in document j, df_i is the number of documents containing i, and N is the total number of documents.

Principal Component Analysis (PCA), a tool for finding patterns in high-dimensional data, is a commonly used linear dimension reduction algorithm. It selects the features with high variance, and improves the calculation speed without the loss of model accuracy [25]. The generated dictionary contained 8974 words. Therefore, the document feature was an 8974-dimensional vector. The number of features was higher than that of the samples. PCA was used to reduce the feature dimension, thereby improving the speed of the model training.

4.4. Classification Method

SVM is a supervised learning model with associated learning algorithms that analyzes data used for classification and regression analysis [26]. SVM belongs to a general linear classifier, and it minimizes empirical errors and maximizes geometric edge regions. To solve the problem of the nonlinear discrimination of datasets in low-dimensional space, a suitable kernel function was selected. Thus, the original finite-dimensional space was mapped into a much higher-dimensional space by nonlinear transformation.

SVM has been used in most of the related works and has proved its success in text classification. SVM chooses the hyperplane to obtain the maximal distance between it and the nearest training-data point, so it has a good generalization ability. With a small amount of instance data to analyze, the SVM algorithm can give a better result. In order to train and test the feature extraction and selection, the authors chose the SVM algorithm of the linear kernel function as the training model of their classifier, with other parameters set to the default value. The linear kernel function has been suggested for text classification [27].

5. Results and Discussion

This section presents the visualization of each document processed with the FPSR method and without the FPSR method, respectively, and the classification results obtained from the different methods.

5.1. Dimensionality Reduction and Visualization Results

The authors conducted the feature extraction of files processed using the FPSR method, and the size of the generated dictionary was 8974. The eigenvectors of each file were reduced to 2D using PCA and visualized in a 2D plane. The result using the FPSR method is shown in Figure 5.

Traditional processing without the FPSR method was also done. When the preprocessing of the web pages was maintained, and the feature extraction and the dictionary initialization were left unchanged, the size of the generated dictionary was 9000. The visualization result is shown in Figure 6 after the same dimension reduction operation.

The number of the features without the FPSR method is higher than the one using the FPSR method, because the FPSR method treats different URLs, paths or email strings as the same pattern. By comparing the visualization results of two methods, it is evident that the boundary of the figure using the FPSR method is clearer and more obvious with fewer overlaps. The classifier presented in this paper would make it easier to distinguish between these two categories of data.

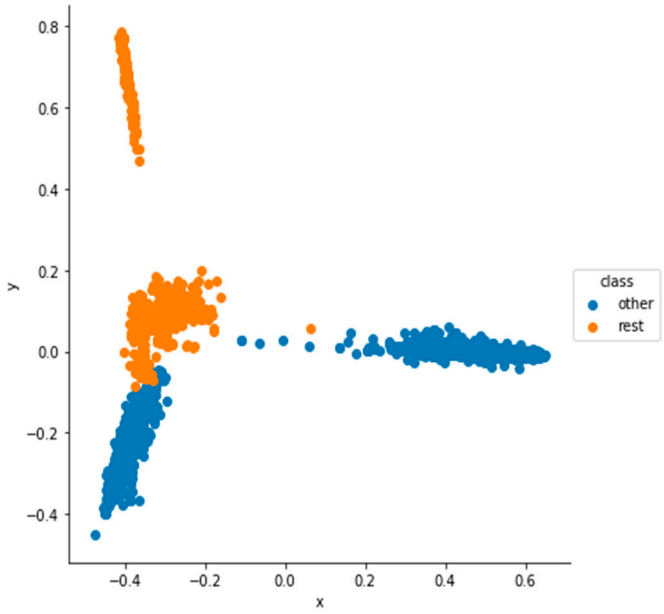

Figure 5. Result using the FPSR method.

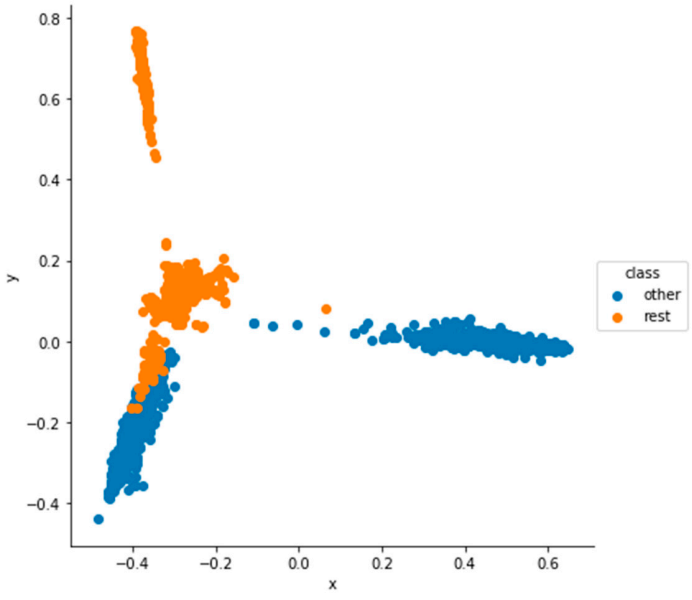

Figure 6. Result without the FPSR method.

5.2. Classification Results

There were 731 RESTful service description pages and 1739 other types of pages in the database. From the data, 80% of each type of data were chosen as the training set, and the remaining 20% was chosen as the test set. First, the dimension was reduced to 2D using PCA, and then a linear kernel function was selected by the SVM algorithm to train the data. The results for precision, recall rate,

and f1-score with and without FPSR are listed in Tables 2 and 3, respectively. The accuracy with FPSR reached 99.4%, whereas that of the traditional method is 94.7%, and the recall rate of the RESTful service pages was only 87.1%.

Table 2. Classification results using FPSR.

	Precision	Recall	f1-Socre
no_restful	0.994	0.997	0.996
restful	0.993	0.986	0.990
avg/total	0.994	0.994	0.994

Table 3. Classification results without FPSR.

	Precision	Recall	f1-Socre
no_restful	0.947	0.980	0.963
restful	0.948	0.871	0.908
avg/total	0.947	0.947	0.947

Figure 7 indicates the confusion matrices of the test result with different preprocessing methods. As shown in Figure 7, the test dataset included 147 RESTful service pages and 348 other pages, and the FPSR method obtained better results, which correctly classified almost all samples with a classification accuracy of 99.4%. The recall score was more important than the precision in this study, because the authors had to identify as many of the RESTful service pages as possible. However, the recall score of the preprocessing without FPSR was only 87.1%. Therefore, the authors concluded that the FPSR method could extract the characteristics of RESTful services more effectively.

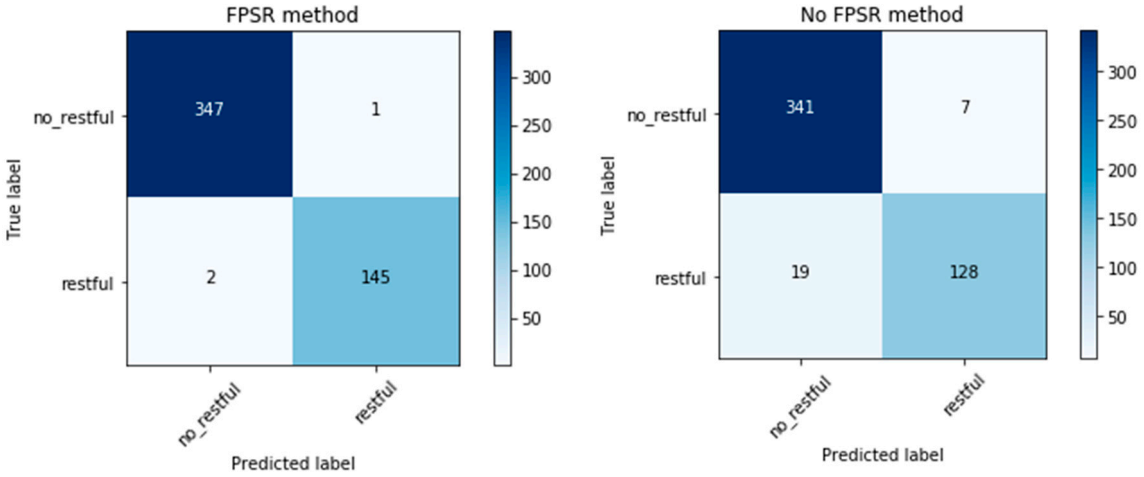

Figure 7. (**a**) The confusion matrix with the FPSR method; (**b**) The confusion matrix without the FPSR method.

6. Conclusions and Future Work

In this paper, the authors proposed a FPSR method for feature extraction and selection of RESTful service description pages related to physiological information. The RESTful service page feature pattern was searched and replaced with a specific string to avoid the destruction of its pattern in the segmentation stage and to retain its feature information. Finally, the results of the dimensionality reduction visualization and classification of the test set showed that this method could more effectively

select and extract RESTful service feature information, thereby achieving a higher classification accuracy and a lower error rate. This will play an important role in the automated discovery of RESTful services related to physiological and bioinformatic information.

In future studies, researchers will have to increase the number of samples in the dataset and enrich its diversity, and then verify the validity of the method on the new dataset. Furthermore, they will have to study the relevant classifier algorithms and evaluate the indicators of each algorithm, such as training time, accuracy, calculation speed, and other aspects, and choose the best classifier algorithm. All this work aims to automatically identify RESTful service pages.

Author Contributions: Conceptualization, C.S. and L.H.; software, C.S.; methodology, C.S. and L.H.; data curation, C.S.; writing–original draft, C.S., L.H., X.G., and Y.W.; writing–review and editing, J.L., L.F., X.Q., Q.Z., Z.W.; visualization, C.S.; supervision, L.H.; project administration, L.H.; funding acquisition, L.H. All the authors have read and approved the final manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (No. 61571443), the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20130008110035), and the National Key Scientific Instrument and Equipment Development Projects (No. 2011YQ080052).

Acknowledgments: The authors would like to thank the Key Laboratory of Agricultural Information Acquisition Technology of the Chinese Ministry of Agriculture for their support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lamine, S.B.A.B.; Zghal, H.B.; Mrissa, M.; Guegan, C.G. An ontology-based approach for personalized restful web service discovery. *Procedia Comput. Sci.* **2017**, *112*, 2127–2136. [CrossRef]
2. Zhang, N.; Wang, J.; He, K.; Li, Z.; Huang, Y. Mining and clustering service goals for restful service discovery. *Knowl. Inf. Syst.* **2018**, 1–32. [CrossRef]
3. Fielding, R.T.; Taylor, R.N. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Thesis, University of California, Irvine, CA, USA, 2000.
4. Haupt, F.; Fischer, M.; Karastoyanova, D.; Leymann, F.; Vukojevichaupt, K. Service composition for REST. In Proceedings of the IEEE 18th International Enterprise Distributed Object Computing Conference (EDOC 2014), Ulm, Germany, 3–5 September 2014; pp. 110–119.
5. Crasso, M.; Rodriguez, J.M.; Zunino, A.; Campo, M. Revising WSDL documents: Why and how. *IEEE Internet Comput.* **2010**, *14*, 48–56. [CrossRef]
6. Sheth, A.P.; Gomadam, K.; Lathem, J. Sa-rest: Semantically interoperable and easier-to-use services and mashups. *IEEE Internet Comput.* **2007**, *11*, 91–94. [CrossRef]
7. Pedrinaci, C.; Domingue, J. Toward the next wave of services: Linked services for the web of data. *J. UCS* **2010**, *16*, 1694–1719.
8. Roman, D.; Kopecký, J.; Vitvar, T.; Domingue, J.; Fensel, D. Wsmo-lite and hrests: Lightweight semantic annotations for web services and restful apis. *Web Semant. Sci. Serv. Agents World Wide Web* **2015**, *31*, 39–58. [CrossRef]
9. Zhao, Y.; Dong, L.; Lin, R.; Yan, D.; Li, J. Towards effectively identifying restful web services. In Proceedings of the 2014 IEEE 21st International Conference on Web Services (ICWS 2014), Anchorage, AK, USA, 27 June–2 July 2014; pp. 518–525.
10. Qi, X.; Davison, B.D. Web page classification: Features and algorithms. *ACM Comput. Surv. (CSUR)* **2009**, *41*, 12. [CrossRef]
11. Asatkar, B.P.; Wagh, K.; Chatur, P. Classification of web pages using naive bayesian approach. In Proceedings of the International Conference on Recent Trends in Engineering Sciences 2017 (ICRTES 2017), Namakkal, India, 25 March 2017; p. 21.
12. Belmouhcine, A.; Benkhalifa, M. Implicit links-based techniques to enrich k-nearest neighbors and naive bayes algorithms for web page classification. In Proceedings of the 9th International Conference on Computer Recognition Systems (CORES 2015), Wroclaw, Poland, 25–27 May 2015; pp. 755–766.
13. Mohamed, T.A.A.A.D.; Abu-Kresha, T.; Bakry, A.N.R. An efficient method for web page classification based on text. *Int. J. Eng. Comput. Sci.* **2017**, *6*, 21854–21860.

14. Kan, M.Y.; Thi, H.O.N. Fast webpage classification using url features. In Proceedings of the 14th ACM Conference on Information and Knowledge Management (CIKM 2005), Bremen, Germany, 31 October–5 November, 2005; pp. 325–326.
15. Hu, R.; Xiao, L.; Zheng, W. Drug related webpages classification using images and text information based on multi-kernel learning. In Proceedings of the SPIE 9th International Symposium on Multispectral Image Processing and Pattern Recognition (MIPPR 2015), Enshi, China, 31 October–1 November 2015.
16. Rajalakshmi, R.; Xaviar, S. Experimental study of feature weighting techniques for url based webpage classification. *Procedia Comput. Sci.* **2017**, *115*, 218–225. [CrossRef]
17. Altay, B.; Dokeroglu, T.; Cosar, A. Context-sensitive and keyword density-based supervised machine learning techniques for malicious webpage detection. *Soft Comput.* **2018**, 1–15. [CrossRef]
18. Siddiqui, A.; Adnan, M.; Siddiqui, R.A.; Mubeen, T. A comparative study of web pages classification methods applied to health consumer web pages. In Proceedings of the 2015 Second International Conference on Computing Technology and Information Management (ICCTIM 2015), Johor, Malaysia, 21–23 April 2015; pp. 43–48.
19. Onan, A. Classifier and feature set ensembles for web page classification. *J. Inf. Sci.* **2016**, *42*, 150–165. [CrossRef]
20. Kiziloluk, S.; Ozer, A.B. Web pages classification with parliamentary optimization algorithm. *Int. J. Softw. Eng. Knowl. Eng.* **2017**, *27*, 499–513. [CrossRef]
21. Kanehisa, M.; Goto, S. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* **2000**, *28*, 27–30. [CrossRef] [PubMed]
22. McWilliam, H.; Li, W.; Uludag, M.; Squizzato, S.; Park, Y.M.; Buso, N.; Cowley, A.P.; Lopez, R. Analysis tool web services from the embl-ebi. *Nucleic Acids Res.* **2013**, *41*, W597–W600. [CrossRef] [PubMed]
23. Herz, A.V.; Meier, R.; Nawrot, M.P.; Schiegel, W.; Zito, T. G-node: An integrated tool-sharing platform to support cellular and systems neurophysiology in the age of global neuroinformatics. *Neural Netw.* **2008**, *21*, 1070–1075. [CrossRef] [PubMed]
24. Uğuz, H. A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. *Knowl. Based Syst.* **2011**, *24*, 1024–1032. [CrossRef]
25. Abdi, H.; Williams, L.J. Principal component analysis. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 433–459. [CrossRef]
26. Wei, Y.; Wang, W.; Wang, B.; Yang, B.; Liu, Y. A method for topic classification of web pages using LDA-SVM model. In Proceedings of the 2017 Chinese Intelligent Automation Conference (CIAC'17), Tianjin, China, 27–29 October 2017; pp. 589–596.
27. Basnet, R.; Mukkamala, S.; Sung, A.H. Detection of phishing attacks: A machine learning approach. In *Soft Computing Applications in Industry*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 373–383.