Design and Development of REST-based Instagram Spam Detector for Indonesian Language

Antonius Rachmat Chrismanto, Willy Sudiarto Raharjo, Yuan Lukito
Duta Wacana Christian University: Department of Informatics, Faculty of Information Technology
Yogyakarta, Indonesia
anton@ti.ukdw.ac.id, willysr@ti.ukdw.ac.id, yuanlukito@ti.ukdw.ac.id

***Abstract*—Instagram, the most popular picture-based social media, has been well known for spam comments, mostly for public figures, including Indonesian actors and actress. Spam comments may lead to misleading information and confusing to other followers. Our previous research on Indonesian language spam comment detector has concluded that K-Nearest Neighbor yields the best result to detect Instagram spam comments. We continue previous our work by designing the REST-based web service to detect Indonesian language spam comments for Instagram. We are using *supervised learning* combined with Instagram dataset gathered from previous work. This system is built on top of Amazon Web Service (AWS) to provide a robust, resilience, and scalable system using Java Jersey library. The result of this work is a preliminary design and implementation of REST-based Instagram spam detector for Indonesian language that has an average response time of 1678.133 ms with deviation standard of 1178.59 ms. The current service still has limitations in terms of number of features and k-d tree data structure and it still needs some optimization before it can be used in production environment.**

Keywords: REST web services, Instagram, spam comments, Indonesian Language

I. INTRODUCTION

Instagram is a very popular picture-based social media service among Indonesian actors and actress. A lot of them used Instagram to boost their popularity and to share their activities with their fans. By following their Instagram accounts, their fans are able to follow their idol visually and interact with fans. Unfortunately, it came at a cost that more and more spam comments appearing as their popularity increased. Spam comments can be a source of information misleading [1] as people might have thought that a post is full of comments, but in reality, most of them are spam. Spam also makes it harder for readers to track information in a post and also causing information becomes irrelevant against the original post by the author. Currently, there is no automatic spam remover provided by Instagram, leaving the user to clean them manually or leave them be polluting their postings.

Eskelinen [2] and Tamir [3] concluded that Instagram users need to spend extra effort to manually verify and remove all spam comments. While Instagram already provided a feature to report spam, it is not the best solution as it still requires users to remove them manually. Another possible solution is to set the account as private, but it will not be beneficial for them as it would decrease their exposure to the public. The last solution was to enable Instagram feature to remove comments that contain user-provided words that are considered as spam. None of the viable solution provided can be done automatically and works for the Indonesian language specifically.

This work is the continuation of our previous 3 years roadmap. In our previous work [4], we have gathered posting and comments data from Indonesian actor/actress with more than 10 million followers and we have collected around 17.000 data. We used the posting and comments as training data in a supervised learning system to detect Indonesian spam comments. We concluded that the best algorithm to be used for spam detection (sorted by their relevance) are K-Nearest Neighbor (k-NN) [1], Support Vector Machine [5], and Naïve Bayes [4]. We continued our work onto the next step, which is designing a service that implements the best algorithm in form of a web service. This work will be a bridge for our final goal which is to make an applied product in form of a browser plugin that can be used by all Instagram users.

II. THEORETICAL REVIEWS

A. Previous Works

Research on spam detection has been conducted in many areas. Some worked on a specific platform, such as Twitter [6] [7] [8] while others are working on various social media networks [9] [10] [11]. We are focusing on text-based spam detection that appears on comments. Spam may appear as hyperlinks, trackbacks, and pingback spam in people's posting [12]. Hines described several ways to detect spam comments by looking at the relevance, posting specific comments, links provided, the validity of the name and email address used, and whether multiple email address are used. Several techniques to reduce the number of spam is to use registration before posting any comments, using CAPTCHA, not allowing HTML code, IP address blacklisting, and to limit repetitive comment in a certain period of time [13].

In our previous work, we conducted an experiment with three different methods against a dataset of 10 Indonesian actress that has more than 10 million followers. We tested Naïve Bayes, Support Vector Machine (SVM), and k-Nearest Neighbor (k-NN) and achieve an accuracy of 75.5% [4], 78.5% [5], and 88.4% [1].

We continue our work by designing REST-based web service to detect Indonesian language spam comments for Instagram using k-NN. This system is built on top of Amazon

This research is sponsored by LPPM Duta Wacana Christian University

Web Service (AWS) to provide a robust, resilience, and scalable system [14] as a foundation for our future work, which is a browser plugin.

We are using REST-based web services since it is simple, low usage on resources, and using HTTP protocol which is widely supported in many cloud service providers. We will conduct testing using assertion as described by Wenhui et al [15].

B. *Amazon Web Service*

Amazon Web Services (AWS) is a business subsidiary of Amazon Inc. providing cloud computing services that were started in 2006 and has become a market leader in cloud computing area. Recently, AWS was declared as a leader in "Magic Quadrant for Cloud Infrastructure as a Service" for 7 consecutive since 2010 [16].

AWS has a number of products that are divided into several categories: Compute, Storage, Database, Migration, Networking & Content delivery, Machine Learning, etc. AWS global architecture is depicted in Fig. 1.

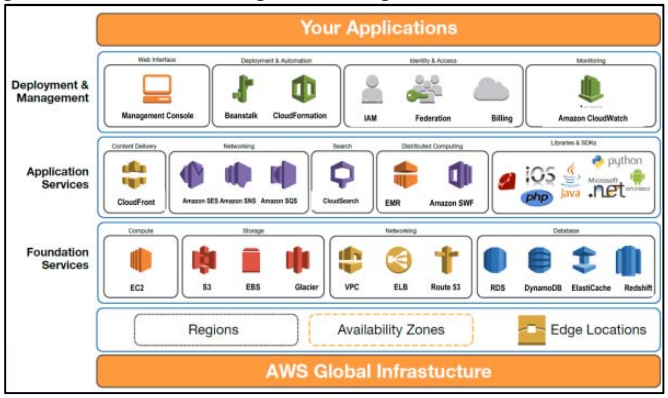

Fig. 1. AWS Architecture [17]

C. *Web Services (REST)*

Web service is a web-based service containing functions that represents the specific task. Web service uses one of the following technology:

1. Remote Procedure Call (RPC)
2. Simple Object Access Protocol (SOAP)
3. Representational State Transfer (REST)

REST Web Services is an architectural design for communication between applications through Internet using HTTP protocol [18]. Several HTTP operations used in REST are GET to fetch resources, PUT to update data, POST to create new resources, and DELETE to delete resources.

Since REST is running on top of the stateless protocol (HTTP), it can gain several advantages, especially related to cloud computing technology. Stateless components can be substituted easily in cloud computing since it does not require any data synchronization. Transactions or requests from clients are not required to be saved for future transactions. REST has several advantages against SOAP, which are better throughput and response time. REST is also more flexible compared to SOAP. [19].

D. *k-NN (k-Nearest Neighbor)*

K-Nearest Neighbor (k-NN) is a method that utilizes statistic principle to find the closest neighbor in the data. In supervised learning, k-NN is often called lazy learner because it does not learn anything from the data in the beginning, but it learns directly while doing classification process. K-NN works by finding some adjacent "k" data object or patterns based on the input and then choosing a class with the highest number of pattern in those k patterns. In short, k-NN using the voting principle to classify a pattern as in Fig. 2. The closest "k" pattern is decided by distance, similarity or dissimilarity, depending on its attributes.

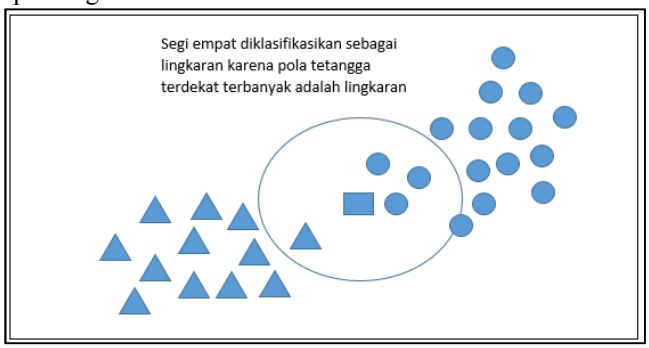

Fig. 2. K-NN Algorithm

K-NN has some advantages [20]:

1. Simple and easy to implement
2. K-NN works a locally by predicting k data until it matched with the locally clustered set of data

But it also has some disadvantages [21]:

1. Not efficient in terms of time because it always performs comparison operation without saving the information, thus not efficient for larger scale of data.
2. K values cannot be decided mathematically, thus further manual observation is still needed.
3. An even value of k also makes system cannot make a good decision. Lower k value will cause generalization, while higher k value will cause overfitting.

III. METHODOLOGY

We started our work by reviewing the AWS global architecture and look for the documentation about what we needed to build and deploy our applications into the AWS environment. Next, we started to design our architecture as in Fig. 3. We are using the compute and storage service mainly in this work, but we are adding more services in the future to ensure the reliability of the system under a production scenario. Our work is using a Java-based REST web services framework called Jersey. It is deployed on top of Apache Tomcat 8 as our main web server.

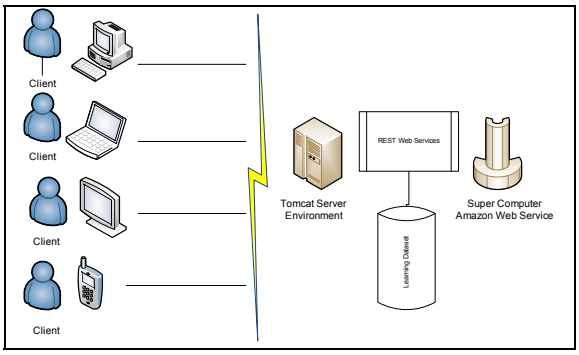

Fig. 3. Draft of Web Service Architecture

Next, we create the learning dataset by using our previous work. The format is in CSV file and it contains several attributes, such as actress ID, user, other attributes that are defined as features, and class (category). The records in CSV file are the weight per document according to their attributes. Table 1 and 2 shows the dataset and training data containing TF-IDF in form of CSV which will be uploaded into the AWS storage service (AWS S3).

TABLE 1. COMMENT DATASET IN CSV FORMAT

id_artist	user	comments	category
522969993	andreasnik oplak	butuh followers instagram atau likes instagram kamu sedikit yuk cek instagram unicorn apps stores melayani dengan ramah	Spam
522969993	nasardigu nawan	Indonesia followers com udah tau tuh bisa nambahin followers	Spam
522969993	agvirasala mi	mirip bapaknya cantik	Notpsam
…dst	…dst	…dst	…dst

TABLE 2. SYSTEM TRAINING DATA

Id	boikot	pacaran	pasif	payudara	…	Kategori
1	0.00505481	0	0	0.00505481	…	Spam
2	0.00607481	0.01705481	0	0	.	Nonspam
3	0.430001	0.00345	0	0	…	Nonspam

When designing the REST web service, we decided to use JSON format as the data-interchange format. We designed to build 5 functions:

- **classify**
  It is used to classify a document / new data in POST method. The input of this function is a document that the user wants to classify and a token.
- **version**
  It is used to show the web service version; implemented in GET method. This function does not require any input.
- **url**
  It is used to show the URL of the learning dataset currently being used; implemented in GET method. This function does not require any input.
- **file**
  It is used to retrieve an active dataset file currently being used in a text format; implemented in GET method. This function does not require any input.
- **dataset**
  It is used to replace dataset file number used as system training data; implemented in GET method. This function requires 1 input, which is dataset number, ranging from 1 – 8.

The last step is to implement the web service and deploy it to the AWS. We used Jersey framework (https://jersey.github.io/) while the k-NN is implemented using Kd-tree Java library (https://home.wlu.edu/~levys/software/kd/). We use the Eclipse equipped with AWS Toolkit plugin as our main IDE.

IV. RESULTS AND DISCUSSIONS

The result of this work is as follow:

1. **A working architecture deployed on top of AWS service**
   The final result of our architecture is depicted in Figure 4.

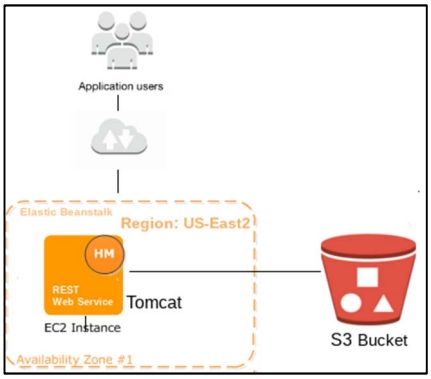

Fig. 4. Web Service Architecture

We deployed Elastic Beanstalk in the Ohio region (US-East2) and we used a Tomcat as our main server. All the dataset are kept in the S3 Bucket storage for durability and performance reason. There is only one instance at this point, but we are expecting to add more instances placed on multiple availability zones and region and setup a load balancer to distribute the load to several instances when the system is ready for production environment.

2. **Supervised Learning Dataset**
   We are using k-NN algorithm for supervised learning classification; thus, a training dataset is required. We have generated 8 training datasets using our own application that was built using PHP and

RapidMiner for this purpose. This enables users to select which datasets being used by using noDataset parameter in the request parameter.

Kd-tree Java library is selected for the k-NN implementation but we found that it has a limitation of handling a maximum of 1024 attributes in one dataset. This drawback forced us to limit every dataset to contain 1000 attributes at maximum. The details for each training datasets is described in Table 3.

TABLE 3. TRAINING DATASET

Code	Dataset	Description
1	PHP-unbalanced-non stem (16000 x 1000)	Number of data: 16000 Number of attributes: 1000 Generated manually by PHP-based system. This dataset contains unbalanced number of spam and non-spam data and they are not yet stemmed.
2	PHP-unbalanced-stem (16000 x 1000)	Number of data: 16000 Number of attributes: 1000 Generated manually by PHP-based system. This dataset contains unbalanced number of spam and non-spam data and they have been stemmed.
3	PHP-balanced-non stem (16000 x 1000)	Number of data: 16000 Number of attributes: 1000 Generated manually by PHP-based system. This dataset contains balanced number of spam and non-spam data and they are not yet stemmed.
4	PHP-balanced-stem (16000 x 1000)	Number of data: 16000 Number of attributes: 1000 Generated manually by PHP-based system. This dataset contains balanced number of spam and non-spam data and they have been stemmed.
5	Rapidminer-unbalanced-non stem (16000 x 1000)	Number of data: 16000 Number of attributes: 1000 Generated by RapidMiner. This dataset contains unbalanced number of spam and non-spam data and they are not yet stemmed.
6	Rapidminer-unbalanced-stem (16000 x 1000)	Number of data: 16000 Number of attributes: 1000 Generated by RapidMiner. This dataset contains unbalanced number of spam and non-spam data and they have been stemmed.
7	Rapidminer-balanced-non stem (16000 x 1000)	Number of data: 16000 Number of attributes: 1000 Generated by RapidMiner. This dataset contains balanced number of spam and non-spam data and they are not yet stemmed.
8	Rapidminer-balanced-stem (16000 x 1000)	Number of data: 16000 Number of attributes: 1000 Generated by RapidMiner. This dataset contains balanced number of spam and non-spam data and they have been stemmed.

3. Java Jersey REST-Based Web Service Implementation

The following are the endpoints for the functional web services:

1. version: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/version (GET)
2. no: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/no (GET)
3. url: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/url (GET)
4. file: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/file (GET)
5. dataset: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/dataset (GET)
6. classify: http://tomcat.zdbgmwikga.us-east-2.elasticbeanstalk.com/service/classify (POST)

All endpoints have been tested and the results can be seen in Fig. 5-10.

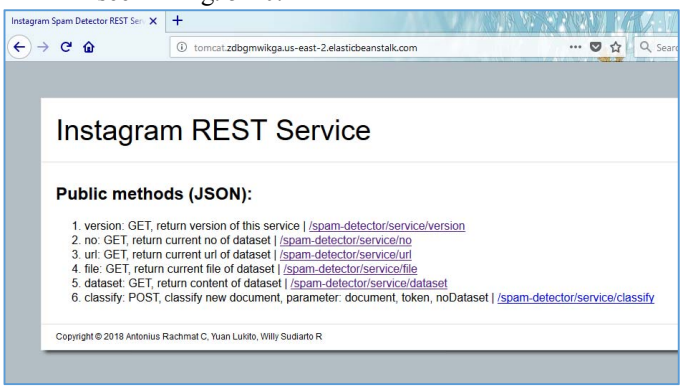

Fig. 5. Web Service Interface

The result of version endpoint using REST GET method is depicted in Fig. 6.

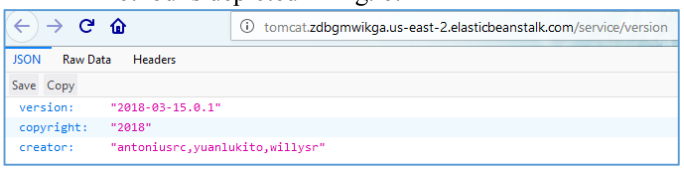

Figure 6. version() function result

The returned data is in form of JSON format as follows: `{"version":"2018-03-15.0.1","copyright":"2018","creator":"antoniusrc,yuanlukito,willysr"}`.
The result of noDataset endpoint using REST GET method is depicted in Fig. 7.

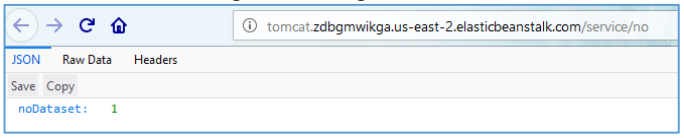

Figure 7. noDataset() function result

The returned data is in form of JSON format as follows: `{"noDataset":1}`.
The result of url endpoint using REST GET method is depicted in Fig. 8.

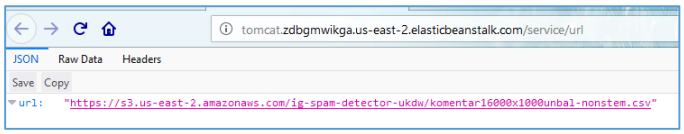

Fig. 8. URL() function result

The returned data is in form of JSON format as follows: `{"url":"https://s3.us-east-2.amazonaws.com/ig-spam-detector-ukdw/komentar16000x1000unbal-nonstem.csv"}`.
The result of getDataset using REST GET method is depicted in Fig. 9.

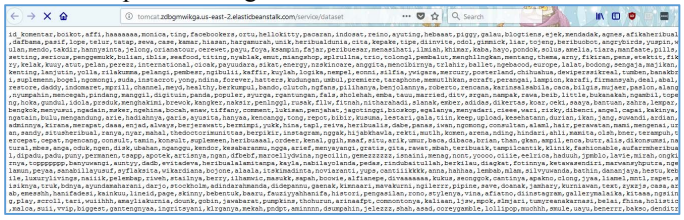

Fig. 9. getDataset() function result

The returned data from classify endpoint using REST POST method with comments, token, and noDataset parameters is depicted in Fig. 10.

TABLE 4. RESPONSE TIME

Test	Response Time (ms)					
	F1	F2	F3	F4	F5	F6
1	448	700	734	684	17000	744
2	330	624	622	628	10704	1672
3	332	484	1054	338	3726	640
4	694	780	636	690	3978	2554
5	624	652	620	1272	4172	802
6	600	628	1604	800	3680	620
7	322	598	1720	792	3412	804
8	602	800	486	334	3490	614
9	678	1628	646	642	3696	4986
10	328	668	786	702	4472	612
AVG	**495.8**	**756.2**	**890.8**	**688.2**	**5833**	**1404.8**
ST DEV	**158.47**	**319.26**	**433.68**	**262.08**	**4490.40**	**1407.63**

The overall average for response time is 1678.133 ms with deviation standard of 1178.6 ms.

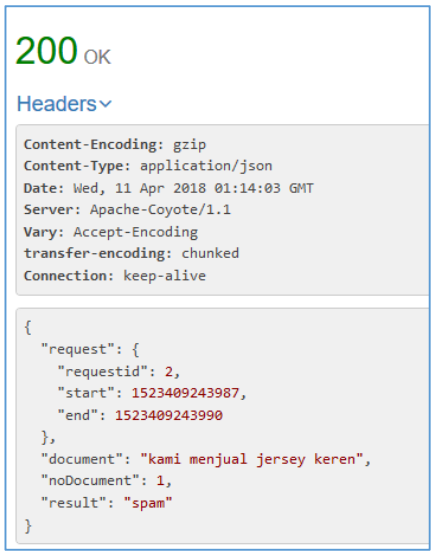

Fig. 10. classify function result

4. Web Services testing

After all web service endpoints are completed, we conducted performance test to see the response time in each service (F1 – F6). We are using RESTClient Firefox plugin (https://addons.mozilla.org/en-US/firefox/addon/restclient/) to conduct the testing and measure the results. The results of the response time tests are displayed in Table 4. Since AWS has many regions available, deploying on different regions might give a different result. Another possible solution to improve performance is to deploy AWS CloudFront, a global content delivery network that can deliver data with lower latency and higher transfer speed since requests will be redirected to closest edge location.

V. CONCLUSION

In this work, we have designed and developed a web service built using Java Jersey on top of Amazon Web Service (AWS). We tested the performance of this web service and the average response time is 1678.133 ms with deviation standard of 1178.6 ms.

We realized that further optimization is still required along with improvement to the number of attributes the system able to handle, improvement to kd-tree data structure, improvement of accuracy against a variety of provided datasets, and response time measurement when the environment is deployed in multiple regions or with CloudFront.

VI. ACKNOWLEDGMENT

This research has been funded by UKDW Research and Community Service (LPPM) unit under grant no. 072/D.01/LPPM/2018. The author also wished to thank Faculty of Information Technology Duta Wacana Christian University for their support.

VII. REFERENCES

[1] A. R. Chrismanto and Y. Lukito, "KLASIFIKASI KOMENTAR SPAM PADA INSTAGRAM BERBAHASA INDONESIA MENGGUNAKAN K-NN," in *Seminar Nasional Teknologi Informasi Kesehatan (SNATIK) 2017*, Yogyakarta, 2017.

[2] M. Eskelinen, "How to get rid of spam comments on Instagram," Mervi Eskelinen, 21 May 2013. [Online]. Available: http://merviemilia.com/blog/2013-05-how-to-get-rid-of-spam-comments-on-instagram. [Accessed 9 August 2017].

[3] D. Tamir, "How To Protect Yourself From Instaspam," 2015. [Online]. Available: http://readwrite.com/2015/04/15/instagram-spam-instaspam-how-to-avoid/. [Accessed 24 January 2018].

[4] A. Rachmat and Y. Lukito, "Deteksi Komentar Spam Bahasa Indonesia Pada Instagram Menggunakan Naive Bayes," *Ultimatics,* vol. 9, no. 1, 2017.

[5] A. R. Chrismanto and Y. Lukito, "Identifikasi Komentar Spam Pada Instagram," *Lontar Komputer,* vol. 8, no. 3, pp. 219-231, 2017.

[6] A. H. Wang, "Don't follow me: Spam detection in Twitter," in *Proceedings of the 2010 International Conference in Security and Cryptography (SECRYPT)*, Athens, Greece, 2010.

[7] A. H. Wang, "Detecting Spam Bots in Online Social Networking Sites: A Machine Learning Approach," in *IFIP Annual Conference on Data and Applications Security and Privacy*, Berlin, Heidelberg, 2010.

[8] M. McCord and M. Chuah, "Spam Detection on Twitter Using Traditional Classifiers," in *International Conference on Autonomic and Trusted Computing*, Banff, Canada, 2011.

[9] E. Tan, L. Guo, S. Chen, X. Zhang and Y. Zhao, "UNIK: unsupervised social network spam detection," in *CIKM '13 Proceedings of the 22nd ACM international conference on Information & Knowledge Management* , San Francisco, California, USA, 2013.

[10] J. Xin, J. Luo, C. X. Lin and J. Han, "A Data Mining-based Spam Detection System for Social Media Networks," in *Proceedings of the VLDB Endowment*, Seattle, Washington, 2011.

[11] X. Zhang, S. Zhu and W. Liang, "Detecting Spam and Promoting Campaigns in the Twitter Social Network," in *2012 IEEE 12th International Conference on Data Mining (ICDM)*, Brussels, Belgium, 2012.

[12] K. Hines, "How to Identify and Control Blog Comment Spam," 2012. [Online]. Available: https://blog.kissmetrics.com/control-blog-comment-spam/.

[13] G. Mishne, C. D and L. R., "Blocking Blog Spam with Language Model Disagreement," in *Proceedings Of The First International Workshop On Adversarial Information Retrieval On The Web (Airweb)*, 2005.

[14] C. Radford, "Challenges and solutions protecting data within Amazon Web Services," *Network Security,* vol. 2014, no. 6, pp. 5-8, 2014.

[15] H. Wenhui, H. Yu, L. Xueyang and X. Chen, "Study on REST API Test Model Supporting Web Service Integration," in *2017 IEEE 3rd International Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, Beijing, China, 2017.

[16] L. Leong, R. Bala, C. Lowery and D. Smith, "Gatner," 15 June 2017. [Online]. Available: https://www.gartner.com/doc/reprints?id=1-2G2O5FC&ct=150519&st=sb&aliId=1221201968.

[17] J. Townsend, "Introduction to Amazon Web Services (AWS)," 20 Juy 2015. [Online]. Available: http://vmtoday.com/2013/07/introduction-to-amazon-web-services-aws/.

[18] R. Fielding, Architectural Styles and the Design of Network-based Software, California: University of California, 2000.

[19] S. Kumari and S. K. Rath, "Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration," Kochi, India, 2015.

[20] Suyanto, Data Mining untuk Klasifikasi dan Klasterisasi Data, Bandung: Informatika, 2017.

[21] S. Jiang, G. Pang, M. Wu and L. Kuang, "An improved K-nearest-neighbor algorithm for text categorization," *Expert Systems with Applications,* vol. 39, no. 1, pp. 1503-1509, 2012.