www.bircu-journal.com/index.php/birci
Budapest International Research and Critics Institute-Journal (BIRCI-Journal)
Humanities and Social Sciences
ISSN 2615-3076 Online)
ISSN 2615-1715 (Print)
Budapest Institute

Context-Aware for Natural Language Processing Services using Microservices Architecture

Abdullah Aziz Sembada[1], Dhomas Hatta Fudholi[2], Raden Teduh Dirgahayu[3]
[1,2,3]*Faculty of Industrial Technology, Universitas Islam Indonesia*
18917102@students.uii.ac.id, hatta.fudholi@uii.ac.id, teduh.dirgahayu@uii.ac.id

Abstract

The development of the industrial era 4.0 and big data has made Natural Language Processing much needed, especially when preprocessing data. With the Natural Language Processing service in order to make it easier for researchers to conduct research because some of their needs have been provided. Before providing services Natural Language Processing first does the system design. The system is built using a microservice architecture, microservice was chosen because it has the characteristics of being flexible, safe, isolated errors, making it very easy to develop the system.feature is added Context-Aware to make it easier for users to process data. The purpose of the research is to be able to integrate Context-Aware into the Natural Language Processing service system so that the system is able to provide recommendations for the most suitable algorithms from the data owned by the user. System test results show that Natural Language Processing service can shorten research on natural language processing. These results cannot be separated from the context-awareness that can determine the type of file or data inputted by the user, thus the user is directly directed by the system to process the file or data using a clustering or classification. The implementation of microservices is also very helpful in development, especially service or algorithms that will not interfere services with existing.

Keywords
microservice; natural language processing; context-aware

I. Introduction

Natural Language Processing is a branch of AI science to make computers understand natural language processing. *Natural Language Processing* is increasingly needed along with the development of text data every day. The development of text data makes research using Natural *Language Processing* increasingly needed, but to conduct research using *Natural Language Processing* must be able to *program*, of course this is a separate obstacle, especially for those who do not have an information technology background.

In this case, many companies already provide *Natural Language Processing service*. Like CloudFactory and prosa.ai, but the services offered by the user do not know the algorithm used or not change the algorithm used. Because of the service that is used on a B2B (*business to business*) basis so there is less support for researchers.

Natural Language Processing also hinders researchers from choosing what method to use.feature was added *Context-Aware*. *Context-Aware* defined as information that can be sensed by the application from the environment that can be used to describe the situation. With *Context-Aware* the system is able to read the file entered by the user and can understand the type of data from the file, whether it is *categorical* or *uncategorical*. If *uncategorical* will be trained algorithm *clustering* and if *categorical* will be trained with a

classification algorithm, then the system will recommend an algorithm with the highest accuracy. The system will only recommend algorithms in *Natural language processing services*. Language is one of the most important things in the life of every human being (Purba, N. et al. (2020).

Microservice is an architectural style consisting of a series of small, independent applications, each running on a single VM and interacting between applications using HTTP APIs or *asynchronous messaging. Microservices* are now increasingly recognized as a promising architectural style for building complex or large-scale systems. Architecture *microservice* is *flexible* and easy to develop, and its independent nature makes it easy to carry out system development. In addition to *microservices* architecture *Monolithic*, but Applications built using *Monolithic* have a disadvantage when the application codebase grows large and changes have to be made quickly. Small scaling is also not possible with *Monolithic*, as the entire application needs to be deployed all the time.

From research it is concluded that *microservice* greatly facilitate system development, in research *microservices* have been implemented in *Natural language processing*. So in this study, we will implement *microservices* for the *Natural language processing services* system, which is expected to facilitate system development and also add *Context-Aware* features to provide recommendations for data processing using *clustering* or *classification*.

Contribution to this research is the implementation of *Context-Aware* and *microservices* for *Natural Language Processing Services.* To be able to explain the above contributions and the position of this research plan, the following describes *the state of the art* previous research on *Natural Language Processing Services, Context-Aware, Microservices*.

The first is research on *Natural Language Processing Services*. Proposed Natural Language Processing for a system that classifies opinions about products and services from the perspective of consumers and tweets for recommendations, this system is also useful for all products and services making it easier for customers. Floating the ITU Turkish NLP Web Service, a system that provides sophisticated research tools, is also mentioned because it provides convenience in performing natural language processing, this system is able to attract a large number of users from various universities in Turkey.

The second research - research on Context-Aware. Developed a system to detect important conversations on cellular phones, it is also mentioned that the system is able to show information about the context of the conversation using a purely content-based approach. Developed a neural machine translation (NMT) system, which is a machine translation equipped with *Context-Aware* so that NMT is able to detect ambiguous pronouns.

Last research on *Microservice*. Choosing *microservices* as the architecture of the *smart city*, so that the system becomes *flexible* so that it is easy to add *services* or replace outdated services outdatedIt is also reported to use *cloud* which provide scalability and cost-effectiveness advantages. *Microservices* are applied to build *Internet of Things* (IoT) / *Mobile*, as reported and also.

From this explanation, it is concluded that there is no research that has implemented *Context-Aware* and *microservices* with case studies of *Natural Language Processing (NLP) Services*. To handle communication between services, we use a RESTful API as has been done.

II. Research Method

Natural Language Processing Services built using *microservices* with a system architecture as shown in Figure 1. *Microservices* are run using docker, docker is recommended because it can facilitate system management for *microservices*. *Natural Language Processing Services* are used to perform text data processing that can be accessed via the UI or API.

The development of *Natural Language Processing Services* is carried out in five steps. The steps taken are data collection, *source code refactoring*, *Dockerize*, *context-aware*, and development and evaluation. The steps of system development can be seen in Figure 1.

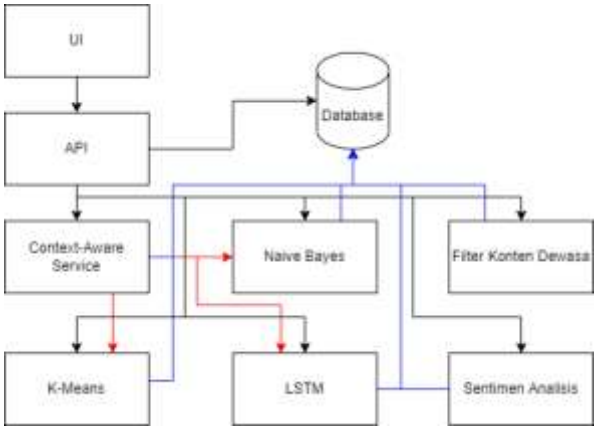

Figure 1. *System Architecture Natural Language Processing Services*

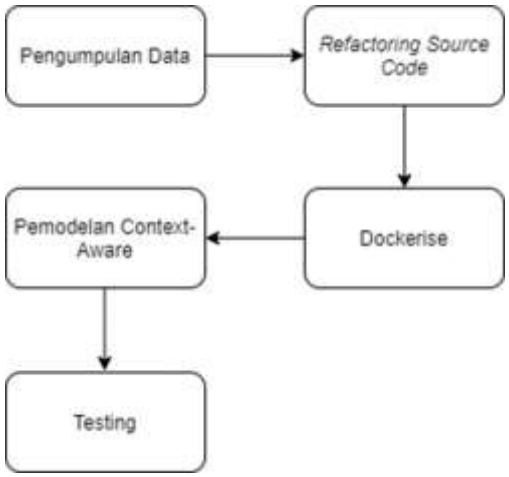

Figure 2. *Research steps*

2.1 Data Collection

The *source code* used to develop the *Natural Language Processing Services* is the *source code* for *Natural Language Processing* written using the Python programming language. Python was chosen because it is a complete programming solution, the language is easy to learn, its flexibility also makes Python a fast-growing language and there are many *libraries*. Especially *the data analytics library* which makes Python a recommended language for data analysis. Python is also simpler than other programming languages such as Ruby, Perl, Java, C++, etc.

2.2 Refactoring Source Code

Refactoring is required to adapt *source code* with *Natural Language Processing Services* so that *source code* can be integrated. *Refactoring* is done by taking the algorithm code, as well as adjusting *input* and *output*. Then the algorithm code is installed on the *controller* Django *Rest Framework*. Django was chosen because it has proven to be *powerful* in code reuse and functional extensibility and fast development, and Django can also improve digitization and *data sharing.*

2.3 Dockerize

Dockerize is needed so that the *source code* can be run as a docker *container*, dockerize is done in three steps as in Figure 3 the first is the creation of the Dockerfile, the Dockerfile serves to define *root* file system *image* container *environment variables*, changes to be made to the *image base*, adding some applications and *ports* and specifying the commands to run after the container is started second *build docker image* and third run *container*.

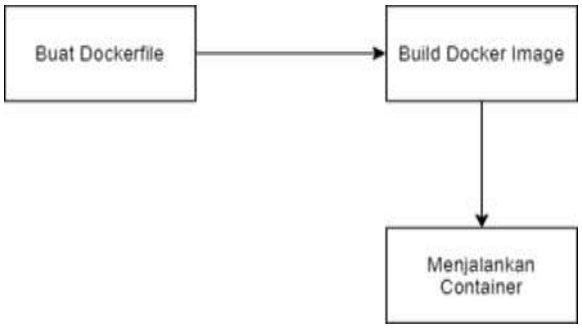

Figure 3. *Dockerize*

2.4 Steps Modeling *Context-Aware*

To provide analytical recommendations, the system reads user-uploaded excel or csv files. If the number of columns in the file is more than two, then the column with the least number of unique data will be used as a label and the column with the most unique number will be used as a feature. If the number of unique data in the feature is less or equal to three (<=3) then the file is categorized as *categorical* and if it is more than three then it is categorized as *uncategorical*. This condition was chosen because the less unique data in the feature, the easier it is to determine the category in each row of data. If the *input is like* Figure 4, the system will recognize the file as categorical because the number of unique data in the feature column is three. After the file category is known, the system recommends an algorithm according to its category. For example, if the category is categorical the system recommends using the *naive Bayes*.

Figure 4. *Input*

2.5 *Testing*

Testing will be done manually on *the context-aware service* , namely by inputting files, then the results from the *context-aware service are* matched with the file categories that have been categorized manually before.

III. Results and Discussion

3.1 Data Collection

Workflow from projects *Source code Adult content classification on* Indonesian *tweets using* LSTM *neural network* and *Long Short-Term Memory* is the user provides *input* form of *text* via *frontend* or API and the *output* from the backend is the result of the analysis of the *input*.

Source code programming language *python* written using google colab *input* of *source code* is an excel file and the output is text.

Source code text preprocessing already uses the Django *rest framework* and can be accessed via the API. *Input* and *output* are JSON.

3.2 Refactoring Source Code

Refactoring is carried out from the *source code* obtained differently for *Adult content classification on Indonesian tweets using* LSTM *neural network* taken *backend adjustments database* and *input* and .

For *source code* of Sentiment Analysis, Naïve Bayes and K-Means, code implementation to the Django *rest framework* was carried out which also made adjustments to *the database* as well as *input* and *output*.

As well as for *source code text preprocessing* and *Long Short-Term Memory* are ready to be implemented in *natural language processing services* because these two *source codes* were created using the Django *rest framework*. *Recode* is done in the *settings* connection *database*

3.3 Dockerize

As shown in Figure 3 there are 3 steps to do to dockerize, namely creating a dockerfile, building *a* docker *image* and running a *container*. Figure 5 is the result of the dockerize process, there are 12 *containers.* Of the 12 *containers* used for *context-aware services,* there are 5 *containers*, namely textpreprocessing_app, lstm-service_app, naïve-bayes_app, k-means_app and context-awarw_app.

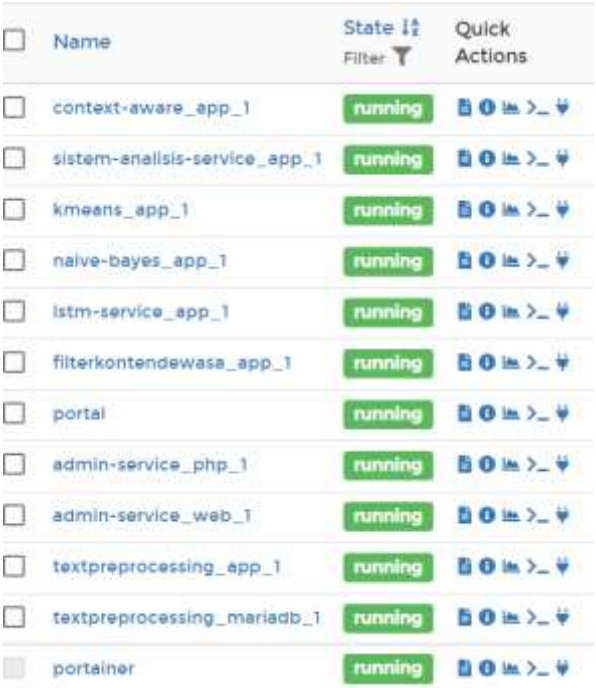

Figure 5. *List of Containers*

3.4 Context-aware Modeling

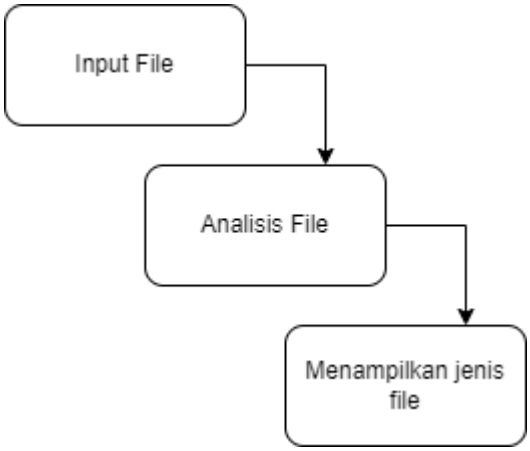

Figure 6. *Context-aware process*

Process flow *context-aware* that consists of 3 stages. First the user uploads a csv/xlsx file, then the file is converted to JSON and sent to *the context-aware service*. Both *services* determine whether the file type is *categorical* or *uncategorical*. The three *services* display whether the file type is *categorical* and *uncategorical*.

The context-aware service can identify whether a file is *supervised* or *unsupervised* by calculating the uniqueness of the data in each column of the file. From Figure 7, it can be seen that the results of 53 datasets tested were able to detect 28 *unsupervised* and 25 *supervised*.

Figure 7. *Context-Aware*

Because *unsupervised* cannot be compared, *the context-aware service* only displays recommendations for the *unsupervised*.algorithm *supervised* the service calculates the accuracy of each algorithm to compare the level of accuracy.

Algorithms *supervised* from 25 dataset files that are categorized as *supervised*, there are only 3 files with a higher accuracy level of *Bayes* while for LSTM there are 8 files that have a higher accuracy level than *Naive Bayes* and the remaining 14 files both have a higher the same accuracy.

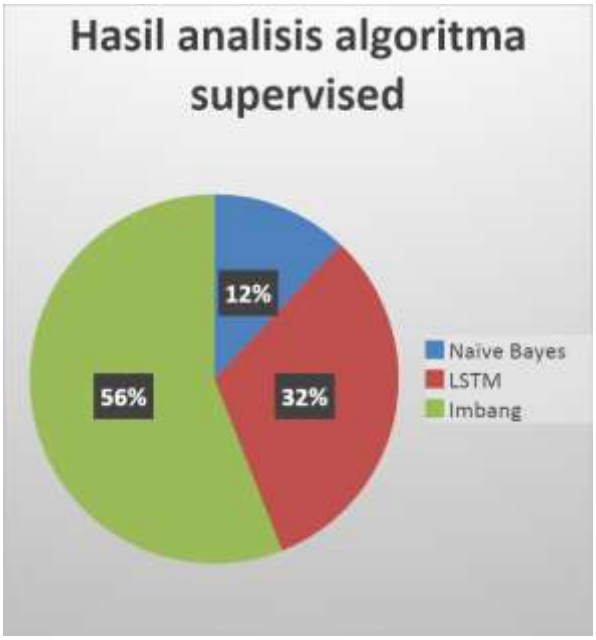

Figure 8. *Algorithm Analysis*

Natural Language Processing Services can be accessed using the API with examples of usage as shown in Figure 9 JSON format to access the Naive Bayes API, Figure 10 JSON format for accessing the LSTM API, Figure 11 JSON format for accessing the adult content filter API and Figure 8. figure 12 JSON format to access the *Text Preprocessing*.

API access is expected to help to access *Natural Language Processing Services* from the notebook or the application itself. To be able to access the API, users must first register with *Natural Language Processing Services* then create a token and each user will get an access_token and access_token_secret.

Figure 9. *Service Naive Bayes Access*

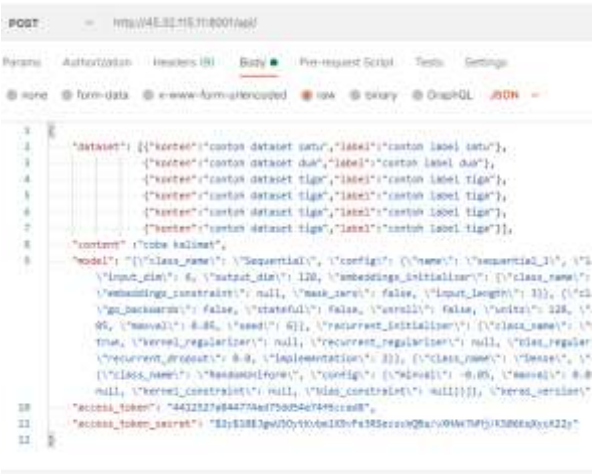

Figure 10. *LSTM Service API Access*

Figure 11. *Adult Content Filter Service API Access*

Figure 12. *Text Preprocessing Service API Access*

IV. Conclusion

This research can shorten research on *natural language processing*, especially in text data research. Because the system is able to show the right algorithm to be used in the file being tested on the system.

The implementation of *microservices* is also very helpful in development, especially *service* or algorithms that will not interfere with existing services with existingWith this, it is also possible to use algorithms with different programming languages because microservices allow applications with different programming languages to communicate with each other.

Context-aware services can recognize files and categorize them as *supervised* or *unsupervised* as shown in Figure 7. From the results in Figure 8, it can be concluded that overall LSTM or Naïve Bayes have the same accuracy. However, in tests with different results, LSTM is superior to naive *Bayes*.

Suggestions for further research are to add algorithms and compare the same algorithm with different languages.

References

A. A. Sembada, "Long Short-Term Memory (LSTM) API," *github*, 2020. https://github.com/azizsembada/lstm-api (accessed Jan. 08, 2021).

A. A. Sembada, "Text Preprocessing," *github*, 2020. https://github.com/azizsembada/text-preprocessing-api (accessed Jan. 08, 2021).

A. F. Hidayatullah,

A. F. Hidayatullah, "Sentiment Analysis Example," *github*, 2019. https://github.com/fathanick/Sentiment-Analysis-Example (accessed Jan. 08, 2021).

A. F. Hidayatullah, A. M. Hakim, and A. A. Sembada, "Adult content classification on Indonesian tweets using LSTM neural network," *2019 Int. Conf. Adv. Comput. Sci. Inf. Syst. ICACSIS 2019*, pp. 235–240, 2019, doi: 10.1109/ICACSIS47736.2019.8979982.

A. Lee, "Why NLP is important and it'll be the future — our future," *https://towardsdatascience.com/*, 2019. https://towardsdatascience.com/why-nlp-is-important-and-itll-be-the-future-our-future-59d7b1600dda (accessed Aug. 10, 2020).

A. Nagpal and G. Gabrani, "Python for Data Analytics, Scientific and Technical Applications," *Proc. - 2019 Amity Int. Conf. Artif. Intell. AICAI 2019*, pp. 140–145, 2019, doi: 10.1109/AICAI.2019.8701341.

C. Santoro, F. Messina, F. D'Urso, and F. F. Santoro, "Wale: A dockerfile-based approach to deduplicate shared libraries in docker containers," *Proc. - IEEE 16th Int. Conf. Dependable, Auton. Secur. Comput. IEEE 16th Int. Conf. Pervasive Intell. Comput. IEEE 4th Int. Conf. Big Data Intell. Comput. IEEE 3*, no. Vmm, pp. 776–784, 2018, doi: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00135.

E. Gossett *et al.*, "AFLOW-ML: A RESTful API for machine-learning predictions of materials properties," *Comput. Mater. Sci.*, vol. 152, no. February, pp. 134–145, Sep. 2018, doi: 10.1016/j.commatsci.2018.03.075.

E. Voita, P. Serdyukov, R. Sennrich, and I. Titov, "Context-aware neural machine translation learns anaphora resolution," *arXiv*, 2018, [Online]. Available: https://arxiv.org/abs/1805.10163

G. Biegel and V. Cahill, "A framework for developing mobile, context-aware applications," *Proc. - Second IEEE Annu. Conf.*

G. Eryiğit, “ITU Turkish NLP Web Service,” pp. 1–4, 2015, doi: 10.3115/v1/e14-2001.
G. Kousiouris *et al.*, “A microservice-based framework for integrating IoT management platforms, semantic and AI services for supply chain management,” *ICT Express*, vol. 5, no. 2, pp. 141–145, Jun. 2019, doi: 10.1016/j.icte.2019.04.002.
G. Kousiouris *et al.*, “A microservice-based framework for integrating IoT management platforms, semantic and AI services for supply chain management,” *ICT Express*, vol. 5, no. 2, pp. 141–145, 2019, doi: 10.1016/j.icte.2019.04.002.
I. F. Siddiqui and N. Babar, “NLP Based Service Recommendation System,” *Int. Conf. Internet*, no. January, pp. 321–324, 2020, [Online]. Available: https://www.researchgate.net/publication/338711391_NLP_BASED_SERVICE_RECOMMENDATION_SYSTEM
J. Vanderplas, “Python Data Science Handbook,” *https://github.com/*, 2018. https://github.com/jakevdp/PythonDataScienceHandbook (accessed Feb. 12, 2021).
L. Bao, C. Wu, X. Bu, N. Ren, and M. Shen, “Performance modeling and workflow scheduling of microservice-based applications in clouds,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 9, pp. 2101–2116, 2019, doi: 10.1109/TPDS.2019.2901467.
M. J. Kargar and A. Hanifizade, “Automation of regression test in microservice architecture,” *2018 4th Int. Conf. Web Res. ICWR 2018*, pp. 133–137, 2018, doi: 10.1109/ICWR.2018.8387249.
M. Kalske, N. Mäkitalo, and T. Mikkonen, “Challenges When Moving from Monolith to Microservice Architecture,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10544 LNCS, pp. 32–47, 2018, doi: 10.1007/978-3-319-74433-9_3.
M. Krämer, S. Frese, and A. Kuijper, “Implementing secure applications in smart city clouds using microservices,” *Futur. Gener. Comput. Syst.*, vol. 99, pp. 308–320, 2019, doi: 10.1016/j.future.2019.04.042.
NLP-Course-UII,” *Github*, 2021. https://github.com/fathanick/NLP-Course-UII/tree/master/Preprocessing (accessed May 01, 2021).
P. L. Lokapitasari Belluano, P. Purnawansyah, B. L. E. Panggabean, and H. Herman, “Sistem Informasi Program Kreativitas Mahasiswa berbasis Web Service dan Microservice,” *Ilk. J. Ilm.*, vol. 12, no. 1, pp. 8–16, 2020, doi: 10.33096/ilkom.v12i1.492.8-16.
P. Merson and J. Yoder, “Modeling Microservices with DDD,” *Proc. - 2020 IEEE Int. Conf. Softw. Archit. Companion, ICSA-C 2020*, pp. 7–8, 2020, doi: 10.1109/ICSA-C50368.2020.00010.
P. Prashant, A. Tickoo, S. Sharma, and J. Jamil, “Optimization of cost to calculate the release time in software reliability using python,” *Proc. 9th Int. Conf. Cloud Comput. Data Sci. Eng. Conflu. 2019*, pp. 470–474, 2019, doi: 10.1109/CONFLUENCE.2019.8776620.
Purba, N. et al. (2020). Language Acquisition of Children Age 4-5 Years Old in TK Dhinukum Zholtan Deli Serdang. *Linglit Journal: Scientific Journal of Linguistics and Literature*. P.19-24
Q. Yu and W. Yang, “The Analysis and Design of System of Experimental Consumables Based on Django and QR code,” *Proc. - 2019 2nd Int. Conf. Saf. Prod. Informatiz. IICSPI 2019*, pp. 137–141, 2019, doi: 10.1109/IICSPI48186.2019.9095914.
R. S. Trofin, C. Chiru, C. Vizitiu, A. Dinculescu, R. Vizitiu, and A. Nistorescu, “Detection of astronauts’ speech and language disorder signs during space missions using

natural language processing techniques," *2019 7th E-Health Bioeng. Conf. EHB 2019*, pp. 1–4, 2019, doi: 10.1109/EHB47216.2019.8969950.

R. Wang, M. Imran, and K. Saleem, "A microservice recommendation mechanism based on mobile architecture," *J. Netw. Comput. Appl.*, vol. 152, no. October 2018, p. 102510, 2020, doi: 10.1016/j.jnca.2019.102510.

S. Liu, Y. Li, G. Sun, B. Fan, and S. Deng, "Hierarchical RNN Networks for Structured Semantic Web API Model Learning and Extraction," *Proc. - 2017 IEEE 24th Int. Conf. Web Serv. ICWS 2017*, pp. 708–713, 2017, doi: 10.1109/ICWS.2017.85.

S. P. Ma, C. Y. Fan, Y. Chuang, W. T. Lee, S. J. Lee, and N. L. Hsueh, "Using Service Dependency Graph to Analyze and Test Microservices," *Proc. - Int. Comput. Softw. Appl. Conf.*, vol. 2, pp. 81–86, 2018, doi: 10.1109/COMPSAC.2018.10207.

S. Roca, J. Sancho, J. García, and Á. Alesanco, "Microservice chatbot architecture for chronic patient support," *J. Biomed. Inform.*, vol. 102, p. 103305, 2020, doi: 10.1016/j.jbi.2019.103305.

W. Hasselbring and G. Steinacker, "Microservice architectures for scalability, agility and reliability in e-commerce," *Proc. - 2017 IEEE Int. Conf. Softw. Archit. Work. ICSAW 2017 Side Track Proc.*, pp. 243–246, 2017, doi: 10.1109/ICSAW.2017.11.

Z. Yu, J. Han, T. Zhao, N. Tian, and J. Wang, "Research and implementation of online judgment system based on micro service," *Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS*, vol. 2019-Octob, pp. 475–478, 2019, doi: 10.1109/ICSESS47205.2019.9040684.