Journal of Information Technology and Computer Science
Volume 4, Number 1, June 2019, pp. 33-41
Journal Homepage: www.jitecs.ub.ac.id

Audit System Development for Government Institution Documents Using Stream Deep Learning to Support Smart Governance

Imam Cholissodin[1], Arief Andy Soebroto[2], Sutrisno[3]
[1,2]Faculty of Computer Science, Computer Science, Brawijaya University, Malang, Indonesia
{[1]imamcs@ub.ac.id, [2]ariefas@ub.ac.id, [3]trisno@ub.ac.id}

Received 27 November 2018; accepted 16 May 2019

Abstract. Document audit system is a means of evaluating documents on the results of delivering information, administrative documentary evidence in the form of texts or others. Currently, these activities become easier with the presence of computer technology, smartphones, and the internet. One of the examples is the documents created by various government institutions whether local, city and central government. The instance is online-published documents that are shaded by certain government institutions. Before the documents are published or used as an archive or authentic evidence for reporting or auditing activities, the documents must go through the editing stage to correct if there are errors and deficiencies such as spelling errors or incomplete information. In the editing process, however, a person may not be able to escape from making mistakes that result in the existence of writing errors after the editing process before the submission. Word spelling mistakes can change the meaning of the conveyed knowledge and cause misunderstanding of information to the readers, especially for assessors or the audit team. Based on the problem, the researcher intends to assist the work of the audit preparation team in document analysis by proposing a system capable of detecting word spelling errors using the Dictionary Lookup method from Information Retrieval (IR) and Natural Language Processing (NLP) science combined with Stream Deep Learning algorithms. Dictionary Lookup method is considered effective in determining the spelling of words that are true or false based on Lexical Resource. In addition, String Matching method that has been developed can correct word-writing errors correctly and quickly.
Keywords: spelling mistake detection, dictionary lookup, audit of government institution documents, stream deep learning

1 Introduction

Document is the result of writing as a form of delivery, and communication and information in human's daily activities that has been carried out long ago, even before the Christ by using certain symbols on very primitive media, for example carving on stones, palm fronds, leaves, and so on [1]. As time went on from time to time, writing also underwent improvements. Today, in the era of sophisticated technological developments where many people use computers, smartphones, and the internet, the art of writing also undergoes improvements and continues to grow. The existence of these technologies that develop rapidly is widespread in the community so increasingly that it encourages an increasing number of people who are able to write documents well in various aspects [2]. In government institutions, many writings in the form of documents

are the results of policies, activities and the others for audit and archiving needs that are published online. Before the publication, each document should be required to go through the editing stage to check if there are deficiencies and misspellings of words or incomplete information. A person, however, in doing his assignment in writing documents cannot escape from accidental typing errors as well. Although several checks have been done manually by several people from the editing team, errors in word writing can still be rediscovered after the editing stage due to negligence or lack of knowledge of the spelling of words that are up to date and in accordance with KBBI (Indonesian Dictionary). The editorial team usually focuses on major errors related to the substance of the document so that the process of detecting the misspellings of words that seem simple but many can escape checking. Yet, small things like word writing errors can affect the information and knowledge presented in the document. Furthermore, it is also related to the limited time of checking and the number of existing documents with various number of pages, which allows editors to be less thorough or sometimes tend to be hasty in analyzing the documents. Therefore, there is a requirement for a system that is able to work automatically to assist the work of the editing team so that they can detect if there are typing errors in the documents.

Several studies have been performed on the detection of word writing errors, for example in the form of *automatic spell checker* applications. Automatic spell checkers have been carried out since the 1960s and have been developed to date [3]. The aim is to improve and revise the quality of detection results, and utilize and develop knowledge related to what methods are appropriate to make the system works. A research was conducted by Faili (2010) which focused on the detection and correction of writing errors for Persian using the *Bayesian rule* probability method [4]. The test data used from IRNA corpus amounts to 100 sentences in which each sentence is not more than 15 words. From the test results, the good evaluation value of precision value is 80.5% and recall is 87%. The number of researches on Indonesian word-writing errors and corrections is still small and the use of the method is still limited to a few simple methods. In the research on detection and correction of non-word errors of Indonesian words, Soleh and Purwarianti (2011) used a detection method of a dictionary lookup with several other compared methods. The results showed that the *Probability of Similarity* method was better than the *Forward Reversed Dictionary* with accuracy values of 98.55% and 97.59% respectively [5].

Based on several background problems above and the previous studies, as well as the author's analysis on the combination of problem-based and some of the previous studies, the ones of which showed the small number and too simple methods of researches on Indonesian-language documents, this study proposes making an intelligent system to audit government institution documents using *stream deep leaning*. It is expected that the results of this system can help and support smart governance, and can add the insight especially in the fields of IR and NLP [6][7].

2.Method

2.1 Smart Governance and Audit Documents

Smart Governance is the activity of developing a government system with the adoption of the latest technology for process automation, and the main thing is how appropriate the technology adoption by government institutions is as shown in Figure 2.1.

Auditing documents in government institutions is related to how to evaluate any forms of text documents, from correspondence documents to form documents in preparation for ISO and so on, automatically using a computerized system, in which there are some of the following cases (District Government/ City Government Service, 2018):

a. Automated evaluation of government institution documents that will be or have been archived in the database to provide the percentage of typo errors.
b. Automated evaluation of government institution documents that will be or have been archived in the database to provide the results of the classification of document quality, for example, rated 1 to 3 (starting from not very good to very good class/level) in accordance with the rules used in processing text documents.
c. Automated evaluation of government institution documents that will be or have been archived in the database to provide the results of the percentage of similarity of with other documents outside the government institution.
d. Automated evaluation of government institution documents that will be or have been archived in the database to provide the results of the percentage of suitability with the templates (rules of law, standard format, etc.) that become the reference documents.

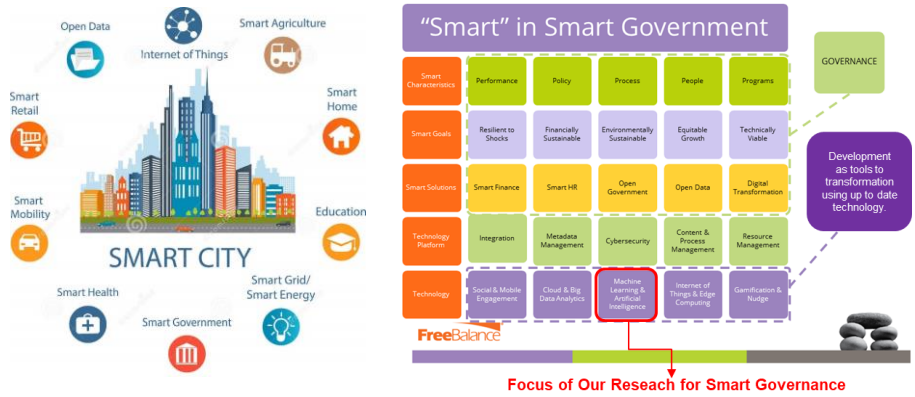

Figure 2.1 Smart City and "Smart" in Smart Government

2.2 Matlab, Frontend and Backend Python Django

Matlab is a programming language that is quite reliable for making prototypes and final versions of program code while Django is one of the many frameworks that use python language. Django has dozens of advantages that can be employed to handle the needs in making general applications such as handling user authentication, content administration, RSS and so on. Django is also very concerned about security, for example, securing applications from SQL Injection, cross-site scripting, cross-site request forgery, and clickjacking. Django in terms of interface also frees users to determine the desired design [8].

2.3 Propose Method: Stream Deep Learning

AI on the system, which is related to the detailed intelligence system process when it is run so that the results of evaluation values can be obtained in the form of accuracy values. Stream Deep Learning, which is a contribution to this research, combines ELM-based Simplified Deep Learning (SDL-ELM) from convolution neural network (CNN) and Extreme Learning Machines training speed with data processing that utilizes Matlab, backend (Python Django) and frontend (PHP) Web App [9]. In Figure 2.2, SDL-ELM structure consists of an input layer, an output layer and several hidden layers arranged as a convolution layer of a single unit, followed by a pooling layer. The number of convolution and pooling layers depends on the complexity of the case. The convolution layer consists of several feature groups and the pooling layer consists of a precis of

several feature groups. The following is the detailed steps of Stream Deep Learning for the classification process:

1. Create a folder in "Document Audit System Application (SuMe App)" by combining Convolution, Sig/ReLU, Pooling, and Fully Connected process, as in Figure 2.2.

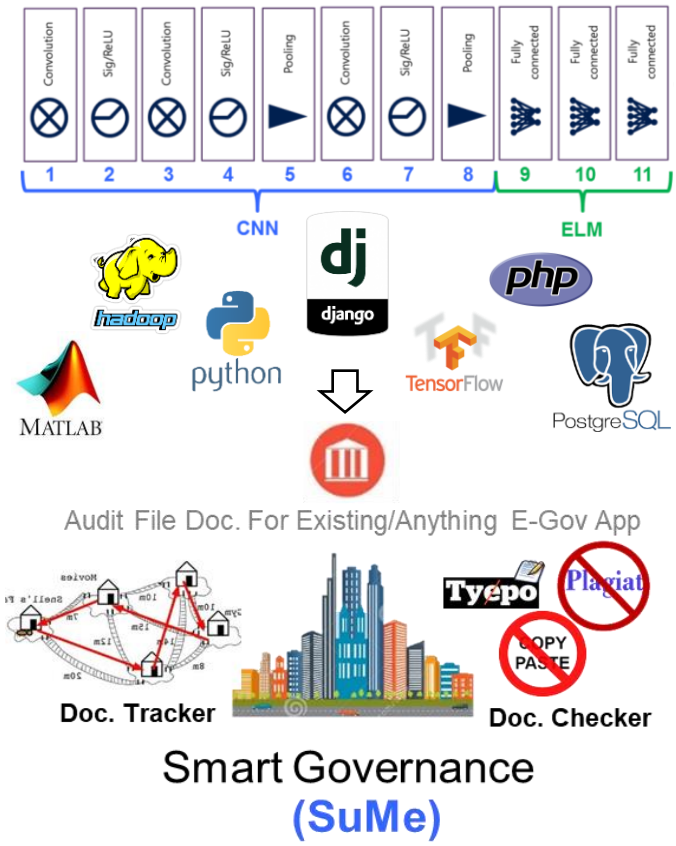

Figure 2.2 Map Stream Deep Learning of SuMe App

2. Determine the parameter value.
   a. For the normalization process of feature values.
   b. For the convolution process. Set, for example with three types of filters: in which, 1st (conv11): average filter, 2nd (conv12): max filter, and 3rd (conv13): std filter, std (standard deviation).
      numFilter = 3; and,% of the amount of padding (*k*), filter matrix size (*k* x *k*), for example *k* = 3;
   c. For the pooling process.
3. Perform the training Process
   a. Preprocessing
      [numData, ... numFeature, target, norm]=FnPreProses ('datatrainClassify', ...
      mac, mic, mao, mio);

- 1. Load training data, get numData and numFeature.
- 2. Create "image matrix" for each initial data (only the feature value is taken) from the dataset, which is using Repmat technique.
- 3. Normalization of all "image matrix" data.
  norm{*i*}=(((a{*i*}-mic)./(mac-mic))*(mao-mio))+mio;
  in which a{*i*} is each *i*-th data matrix element, and norm {*i*} defines it as a matrix with a size [numFeature x numFeature]

b. Feature Abstraction with CNN (based on Figure 2.2).

- 1. Convolution Init.
  hC=FnConvDL(norm,numData,k);
  if *k*=3, then expand the edge norm image matrix (padding) with zero value as much as pad size = (*k*-1)/2 = (3-1)/2=1, where *k* is an odd number ≥ 3, and *f* (*i*,*j*) represents the data value.

  The equation of filter 1st: average filter

$$hC\{1\}\{..\}_{[4x4]} = \frac{1}{k.k}\sum_{i=1}^{k}\sum_{j=1}^{k} f(i,j) \quad (1)$$

  The equation of filter 2nd: max filter

$$hC\{1\}\{..\}_{[4x4]} = Max\left(\bigcup_{i=1}^{k}\bigcup_{j=1}^{k} f(i,j)\right) \quad (2)$$

  The equation of filter 3rd: std filter

$$hC\{1\}\{..\}_{[4x4]} = STD\left(\bigcup_{i=1}^{k}\bigcup_{j=1}^{k} f(i,j)\right) \quad (3)$$

- 2. **Sigmoid**/ReLU
  hA=FnSigDL(hC,numFilter,numData);
- 3. Convolution In.
  hC=FnConvInDL(hA,numData,k,numFilter);
- 4. **Sigmoid**/ReLU
  hA=FnSigDL(hC,numFilter,numData);
- 5. Pooling
  hP=FnPoolDL(hA,windows_size,numFilter,numData);
- 6. Convolution In.
  hC=FnConvInDL(hP,numData,k,numFilter);
- 7. **Sigmoid**/ReLU
  hA=FnSigDL(hC,numFilter,numData);
- 8. Pooling
  jika ukuran (hA{i}{j}) = [2 x 2], maka set windows_size = 1
  hP=FnPoolDL(hA,windows_size,numFilter,numData);

c. Fully connected to **ELM** (Based on Figure 2.2).

- 9. Fully connected 1st
  E.g., num_neuron_hidden_layer=5;
  [hFC11,W11,Bias11,Beta11]=FnELMtrainClassify(hP,target,...
  num_neuron_hidden_layer,numData,numFilter);
- 10. Fully connected 2nd
  Eg, num_neuron_hidden_layer=7;
  [hFC12,W12,Bias12,Beta12]=FnELMtrainClassify(hP,target,...
  num_neuron_hidden_layer,numData,numFilter);
- 11. Fully connected 3rd

```
Eg, num_neuron_hidden_layer=4;
[hFC13,W13,Bias13,Beta13]=FnELMtrainClassify(hP,target,...
   num_neuron_hidden_layer,numData,numFilter);
```

4. Perform the Testing Process
   a. Preprocessing

```
[numData2,...
numFeature2,target2,norm2]=FnPreProses('datatestClassify',...
   mac, mic, mao, mio);
```

   b. Feature Abstraction with **CNN** (Based on Figure 2.2).
      - 1. Convolution Init.

```
hC2=FnConvDL(norm2,numData2,k);
```

      - 2. **Sigmoid**/ReLU

```
hA2=FnSigDL(hC2,numFilter,numData2);
```

      - 3. Convolution In.

```
hC2=FnConvInDL(hA2,numData2,k,numFilter);
```

      - 4. **Sigmoid**/ReLU

```
hA2=FnSigDL(hC2,numFilter,numData2);
```

      - 5. Pooling

```
hP2=FnPoolDL(hA2,windows_size,numFilter,numData2);
```

      - 6. Convolution In.

```
hC2=FnConvInDL(hP2,numData2,k,numFilter);
```

      - 7. **Sigmoid**/ReLU

```
hA2=FnSigDL(hC2,numFilter,numData2);
```

      - 8. Pooling

```
if size (hA2{i}{j}) = [2 x 2], then set windows_size = 1
hP2=FnPoolDL(hA2,windows_size,numFilter,numData2);
```

   c. Fully connected to **ELM** (Based on Figure 2.2).
      - 9. Fully connected 1st

```
[Accuracy1,classPredict1,Ytest_predict1]=...
   FnELMtestClassify(hP2,target2,...
   W11,Bias11,Beta11,numData2,numFilter);
```

      - 10. Fully connected 2nd

```
[Accuracy2,classPredict2, Ytest_predict2]=...
   FnELMtestClassify(hP2,target2,...
   W12,Bias12,Beta12,numData2,numFilter);
```

      - 11. Fully connected 3rd

```
[Accuracy3,classPredict3,Ytest_predict3]=...
   FnELMtestClassify(hP2,target2,...
   W13,Bias13,Beta13,numData2,numFilter);
```

   d. Voting to obtain the final classification result

```
% the voting process of some "Fully connected" ELM testing by
comparing the actual class (target) with the prediction class
CompareKelas=[target2' classPredict1 classPredict2 classPredict3];
% of the frequency or appearance calculation for voting material
AllClassPredict=CompareKelas (:,2:end)
classPredictVoting=mode(AllClassPredict')';
num_data_test = numData2;

% of the final accuracy calculation from the voting results
numCorrect=numel(find(target2'-classPredictVoting==0));
accuracy=( numCorrect/num_data_test)*100
```

3 Results and Discussion

In Figures 3.1 and 3.2, there are frontend and backend results, which are related to the user interface validation or whether the interface is appropriate or not.

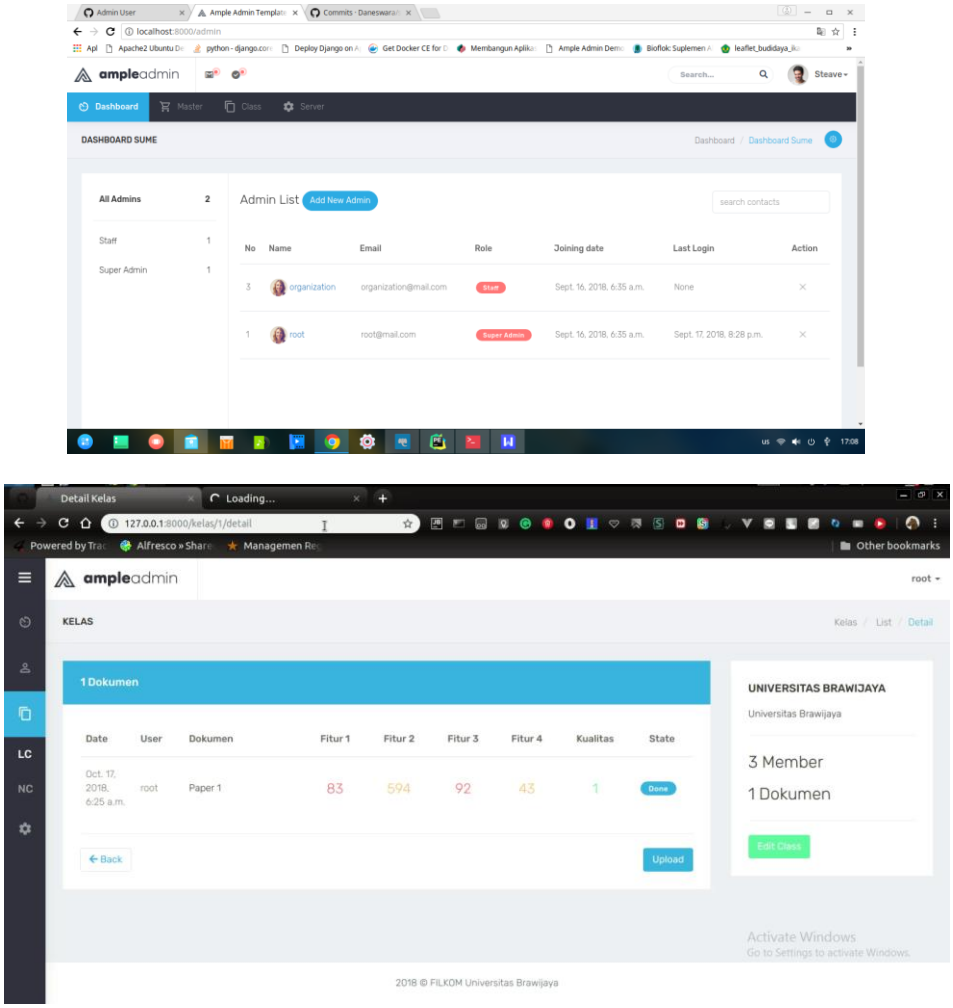

Figure 3.1 Frontend Result

The test results in Figure 3.3 show that the more training data used compared to the test data, the better the accuracy. This is because the system can learn more knowledge from training data. Sehinnga when testing new data, even though it has never been known before, the system results still provide high accuracy. And conversely, if the training data is getting smaller, then the system is less able to recognize the patterns that exist in the data, so that the knowledge gained from the training process is still not fully optimal. But sometimes there are special conditions, namely even though the training data is less than the test data, but provides better accuracy results

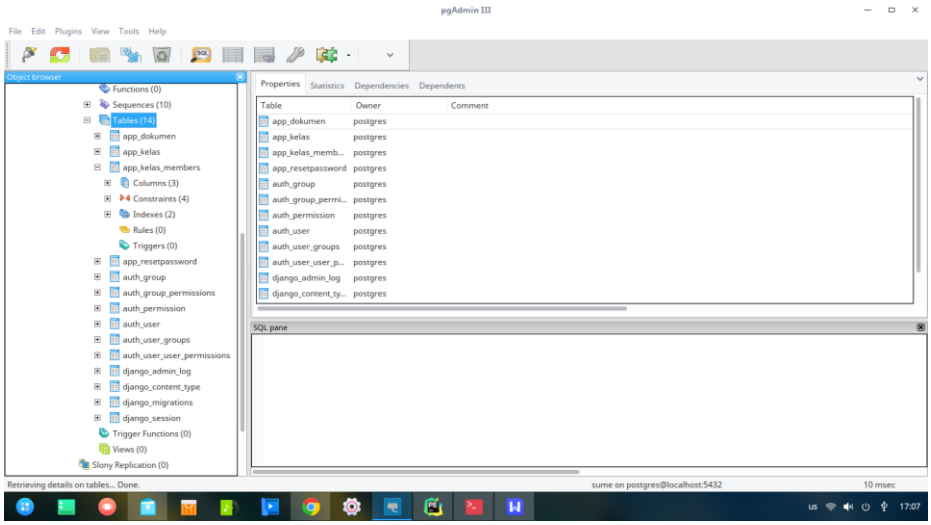

Figure 3.2 Backend Result

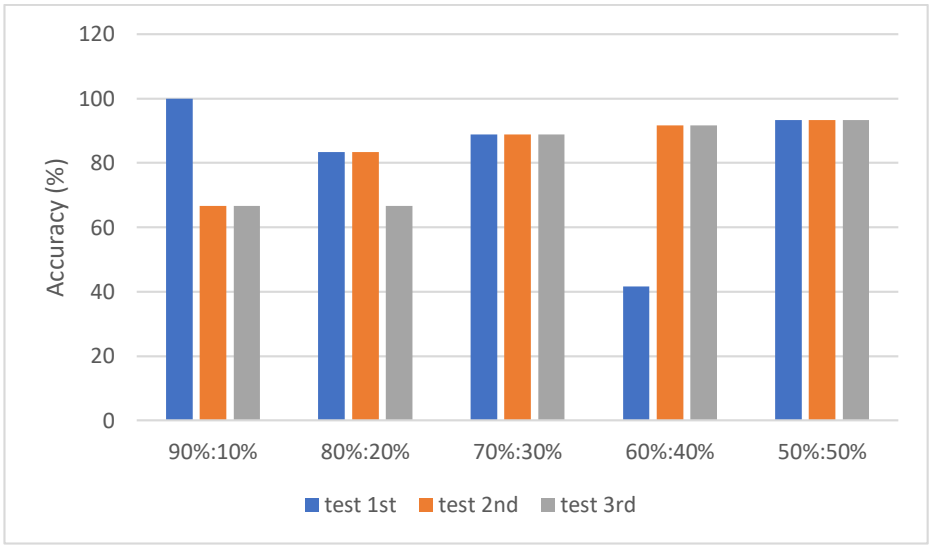

Figure 3.3 Result of Testing data

. This condition seems to be said that the quality of the training data is very good. Therefore, it is necessary to look for how much training data should be used when it is known that the test data is a certain amount and the quality of the training data is also good. Therefore, it is necessary to look for how much training data should be used when it is known that the test data is a certain amount. After obtaining ideal training data (from the results of the search process as much as possible), the system is ready to be tested on arbitrary data, because it is considered to have sufficient knowledge.

4 Conclusion

Based on the result of the basic implementation of the backend and frontend, it shows that the Stream Deep Learning can and is ready to be well-used for the Development of Audit System for Government Institution Documents to support Smart Governance.

This can be observed from the results of the design on the database, the AI planted on the system, to the interface design for the end user and the result of testing process achieved higher accuracy namely 100%. Based on the conclusions, the results obtained from the accuracy 100% achievement can be carried out for a test to the user as an initial trial of the system in the form of product application prototype or Web App. The trial was utilized to measure the readiness of the system while at the same time obtaining user feedbacks about the things that were necessary to be developed in order to improve the application. Thus, when testing the system performance by few parameters, everything can be better prepared and the results were able to meet of the targets to be achieved from both parties, from the application developers and government institutions that tested the document audit system professionally.

References

1. Muradmaulana, 2014. "Sejarah Tradisi Tulis: Dari Masa ke Masa". Available at: http://www.muradmaulana.com/2014/06/sejarah-tradisi-tulis-menulis-dari-masa.html [Accessed 27 November 2018]
2. N, A. R., Kamayani, M., Reinanda, R., Simbolon, S., Soleh, M. Y., & Purwarianti, A. 2011. "Application of Document Spelling Checker for Bahasa Indonesia", 978–979.
3. Ahmed, F., Luca, E. W. De, & Nürnberger, A. 2009. "Revised N-Gram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness".
4. Faili, H. 2010. "Detection and Correction of Real-Word Spelling Errors in Persian Language".
5. Soleh, M. Y., & Purwarianti, A. 2011. "A Non Word Error Spell Checker for Indonesian using Morphologically Analyzer and HMM".
6. Manning, C. D., Raghavan, P., & Schütze, H. 2009. "An Introduction to Information Retrieval". Cambridge University Press.
7. Mishra, R., & Kaur, N. 2013. "A Survey of Spelling Error Detection and Correction Techniques", 4, 372–374.
8. Nawaz, S. 2018. "Backend vs Frontend", sumber: https://medium.com/@shahroznawaz/best-backend-frameworks-to-build-your-next-web-application-2f89f08f34e3 [Accessed 4 October 2018]
9. Cholissodin, I., Sutrino, S., 2018. "Prediction of Rainfall using Simplified Deep Learning based Extreme Learning" Journal of Information Technology and Computer Science (JITeCS) Volume 3, Number 2, 2018, pp. 120-131.