ICESTI2017Spelling

by Viny M

Submission date: 08-Nov-2020 03:03PM (UTC+0700)
Submission ID: 1439409397
File name: matecconf_icesti2018_01047.pdf (1,002.64K)
Word count: 4613
Character count: 23599

MATEC Web of Conferences **164**, 01047 (2018) https://doi.org/10.1051/matecconf/201816401047
ICESTI 2017

Spelling Correction for Text Documents in Bahasa Indonesia Using Finite State Automata and Levinshtein Distance Method

Viny Christanti Mawardi*, *Niko* Susanto, and *Dali* Santun Naga

Faculty of Information Technology, Tarumanagara University, Letjend S. Parman No.1, West Jakarta 11440, Indonesia

Abstract. Any mistake in writing of a document will cause the information to be told falsely. These days, most of the document is written with a computer. For that reason, spelling correction is needed to solve any writing mistakes. This design process discuss about the making of spelling correction for document text in Indonesian language with document's text as its input and a .txt file as its output. For the realization, 5 000 news articles have been used as training data. Methods used includes Finite State Automata (FSA), Levenshtein distance, and N-gram. The results of this designing process are shown by perplexity evaluation, correction hit rate and false positive rate. Perplexity with the smallest value is a unigram with value 1.14. On the other hand, the highest percentage of correction hit rate is bigram and trigram with value 71.20 %, but bigram is superior in processing time average which is 01:21.23 min. The false positive rate of unigram, bigram, and trigram has the same percentage which is 4.15 %. Due to the disadvantages at using FSA method, modification is done and produce bigram's correction hit rate as high as 85.44 %.

Key words: Finite state automata, Levenshtein distance, N-gram, spelling correction

1 Introduction

Language is one of the most important component in human life, which could be expressed by either spoken word or written text. As a text, language became an important element in document writing. Any mistake in document writing will cause the information told falsely. Nowadays, most of the document is written with a computer. In its writing process, there could be some mistakes happened because of human error. The mistake or error can be caused by either the letters from the adjacent keyboard keys, errors due to mechanical failure or slip of the hand or finger. Error when writing a document is often occurred. Especially nowadays, people's awareness to pour his idea into the article, scientific journals, college assignment or other documents have increased. For that reason, spelling correction is needed to solve any writing mistakes. This research aim to objectify spelling correction on Indonesian text documents, to overcome non-word error. This system helps

* Corresponding author: viny@untar.ac.id

the user to overcome document text writing errors, input on this system is a text document and its output is new document text which error of writing have been corrected.

FSA method is used to determine which letter caused error in a word. Levenshtein distance method is used to calculate the difference between the word error and the word suggestion. The word suggestion sequence is determined by the probability of N-gram.

2 Literature review

In this case, the Finite State Automata, Levenshtein distance and N-gram methods are used to create application spelling correction for document text.

2.1 Pre-processing

Pre-processing is the initial stage in processing input data before entering the main stage process of spelling correction for Indonesian text documents. Pre-processing consists of several stages. The pre-processing stage is done case folding, eliminating numbers, punctuation and characters other than the alphabet letter, tokenization.

The pre-processing step are explained below:

(i) Case folding

Case folding is the stage that changes all the letters in the document into lowercase. Only the letters 'a' to 'z' are accepted. Case folding is done for inputs, because the input can be capitalized letter or non capitalized letter. For example from the input "i just Want to Eat ", we'll get output "i just want to eat ".

(ii) Eliminating numbers, punctuation and characters other than the alphabet

Eliminating numbers, punctuation and characters other than the alphabet is done because the characters are considered as delimiters and have no effect on text processing. There are exceptions to the eliminating punctuation because it will affect the process of building N-grams, the examples is ". (dots) "and", (comma) ".

(iii) Tokenization

Tokenization is commonly understood as "splitting text into words." using white space characters as delimiters and isolate punctuation symbols when necessary [1]. For example from input text "i just want to eat ", we can get tokens "i", "just", "want", "to", "eat ".

2.2 Finite state automata (FSA)

Finite State Automata (FSA) is a mathematical model that can accept inputs and outputs with two values that are accepted and rejected. The FSA has a limited number of states and can move from one state to another by input and transition functions. The FSA has no storage or memory, so it can only remember the current state [9].

There are two types of FSA, the first is Deterministic Finite State Automata (DFA) expressed by five tuples:

$M = (Q, \Sigma, \delta, S, F)$

Q = Finite set of states

Σ = Finite set of input symbols called the alphabet

δ = Transition function $\delta : Q \times \Sigma$

S = Initial or start state, $S \in Q$

F = Final state, $F \subseteq Q$

The second is the Nondeterministic Finite State Automata (NFA) which differs from the Deterministic Finite State Automata (DFA) only in the transition function $Q \times (\Sigma \cup \{\varepsilon\})$.

That means, for each state there may be more than one transition, and each may lead to a different state. Finite State Automata is a good data structure for representing natural language dictionaries. FSA is used to represent dictionaries and word errors to facilitate Levenshtein distance calculations [3].

2.3 Levenshtein distance

Levenshtein distance, could also be referred to as edit distance, is a measurement (metric) by calculating number of difference in the two strings. The Levenshtein distance from two strings is the minimum number of point mutation required to replace a string with another string. Point mutations consist of:

(i) Substitutions: substitutions is an operation to exchange a character with another character. For example the author writes the string "eat" become "eay", in this case the character "t" is replaced with the character "y".

(ii) Insertions: insertions is the addition of characters into a string. For example the string "therfor" becomes "therefore" with an addition of the character "e" at the end of the string. The addition of characters is not restricted at the end of a word, but can be added in the beginning or middle of the string.

(iii) Deletions: deletions is deleting the character of a string. For example the string "womanm" become "woman", in this operation the character "m" is deleted [4].

Distance is the number of changes needed to convert a string into another string. The following equation or algorithm used to calculate the Levenshtein distance between two words $a = a_1 \ldots a_n$ and $b = b_1 \ldots b_m$ are:

$$d_{i0} = \Sigma_{k=1}^{i} w_{del}(b_k), \quad \text{for } 1 \le i \le m \tag{1}$$

$$d_{0j} = \Sigma_{k=1}^{j} w_{ins}(a_k), \quad \text{for } 1 \le j \le m \tag{2}$$

$$d_{ij} = \begin{cases} d_{i-1,j-1} & \text{for } 1\, a_j = b_j \\ \min \begin{cases} d_{i-1,j} + w_{del}(b_k) \\ d_{i,j-1} + w_{ins}(a_j) \\ d_{i-1,j-1} + w_{sub}(a_j, b_j) \end{cases} & \text{for } 1 \le i \le m, 1 \le j \le n \\ & \text{for } a_j \ne b_j \end{cases} \tag{3}$$

Where : a = Words represented as rows
b = Words represented as columns
i = Letter index of word a
j = Letter index of word b
n = Letter length of the word a
m = Letter length of the word a
w = The value of deletions, insertions, or substitutions (w = 1)

This algorithm start from the top left corner of a two-dimensional array that has filled in a number of initial sring characters and target strings with cost. The cost at the lower right end becomes the levenshtein distance value, that describes the number of differences between the two strings.

2.4 N-gram

N-gram is a probabilistic language model to predict the next item in sequence. The N-gram model is widely used in probability, communication theory, linguis computing (e.g., statistical natural language processing), biological computing (e.g., biological sequence analysis), and data compression. There are two benefits of the N-gram model that is simple and scalability [5].

The number of N-grams in a sentence can be calculated using the following equations:

$$\text{N-gram}_k = X-(N-1) \tag{4}$$

Where: X = Number of words in a sentence k
N = Number of N-grams

The easiest way to calculate the probability is with Maximum Likelihood Estimation (MLE), by taking the number of corpus and dividing to produce interval [0,1]. For example, to calculate the probability of bigram the word y against x is by calculating amount number of bigrams c (xy) of the corpus and dividing by the number of corpus x. Here is the equation used to calculate the probability bigram:

$$P\langle W_n \,|\, W_{n-1}\rangle = \frac{c(W_1 W_n)}{c(W_{n-1})} \tag{5}$$

Where: P = N-gram probabilities
w = Word
n = Index
c = Word frequency

For the general case of MLE N-gram parameter estimation :

$$P\left\langle W_n \,\middle|\, W_{n-N+1}^{n-1}\right\rangle = \frac{c\left(W_{n-N+1}^{n-1} / W_n\right)}{\left(W_{n-N+1}^{n-1}\right)} \tag{6}$$

The probability of N-gram can be zero, due to the limitations of the corpus. Therefore modification of Maximum Likelihood Estimation (MLE) is required, the modification is called smoothing.

2.5 Add-one smoothing

There are many smoothing techniques for Maximum Likelihood Estimation (MLE) that can done from the simplest (add-one smoothing) to the sophisticated smoothing techniques, such as Good-Turing discounting or back-off dels. Some of these smoothing methods work by determining a distribution value of N-grams and using Bayesian inference to calculate the probability of N-grams produced. However, more sophisticated smoothing methods usually does not use the method but through independent considerations instead [6].

Spelling correction for document text using add-one smoothing, because add-one smoothing is the simplest method to implement. Add-one smoothing is a very simple

smoothing algorithm, which adds a value of one at all frequencies and normalizes the denominator because of this addition. The unigram probability equation is as follows :

$$p(W_i) = \frac{c_i}{N} \tag{7}$$

Where c_i is the number of words i and N is the total number of words. The unigram probability equation after smoothing using the add one method is as follows :

$$P_{addone}(W_i) = \frac{c_i + 1}{N + V} \tag{8}$$

Where V is the number of words (unique). Bigram probability equation after smoothing :

$$P_{addone}\langle W_n | W_{n-1} \rangle = \frac{c(W_1 W_n) + 1}{c(W_{n-1}) + V} \tag{9}$$

2.6 Perplexity

The correct way to evaluate the performance of a language model N-gram is to embed it in an application. There are some methods to evaluate this performance. One of these methods is in vivo evaluation. But this method is very wasteful in time which may take hours or even days. Perplexity (PP) is the most common evaluation metric for N-gram language models.

The intuition of the perplexity depends on two probabilistic models. The better model is the one that has tighter fit to the test data also predict better details on the test data. A better prediction depends on the probability. If the probability of test data is high the model is good. If the probability of the word sequence is high, then the perplexity is then low. Minimizing the perplexity is equal to maximizing the probability [7].

The mathematical concept of the perplexity of language model on a test set is function of probabil normalized by the number of words. The perplexity of a test set W=w1 w2…wN is in Equation 12:

$$pp(\mathrm{W}) = p(w_1 w_2 ..... w_N)^{-\frac{1}{N}} \tag{10}$$

$$pp(\mathrm{W}) = \sqrt[N]{\frac{1}{p(w_1 w_2 ..... w_N)}} \tag{11}$$

$$pp(\mathrm{W}) = \sqrt[N]{\Pi_{i=1}^{N} \frac{1}{p(w_1 w_2 ..... w_N)}} \tag{12}$$

Where: W = Test set (sentence)
N = Number of words in W
P = w probabilities
i = Indeks

2.7 Correction hit rate and false positive rate

Correction hit rate, it is the ratio between the number of typos corrected and the total number of typos [8].

$$\text{Correction hit rate} = \frac{\Sigma \text{typos corrected}}{\Sigma \text{typos}} \times 100\% \quad (13)$$

The higher the percentage of correction hit rate, the higher the accuracy of spelling correction is done. In addition to looking for a correction hit rate that indicates the accuracy of the designed spelling correctio system, it also determines the false positive rate. False positive rate, it is the ratio between the number of false positives (i.e., correct words that are wrongly detected as typos) the total number of correct words in the tested sentences. The lower, the better [9].

$$\text{False positive rate} = \frac{\Sigma \text{false positive}}{\Sigma \text{correct words}} \times 100\% \quad (14)$$

3 Planning and implementation

3.1 System plan

The designed system is a web-based spelling correction for text documents. The system is designed to allow users to manually and automatically perform the correction. Users can input text documents in the spelling correction system, then the system will display the contents of the document and mark the location of the error word with red color.

Manual correction can be done by clicking on a word that has a mark and the system will give word suggestions for the error word, where the user can choose the word suggestion that he/she consider as the correct word. Automatic correction can be done only by clicking a button, then manual or automatic correction will result in the output of a text document whose error has been corrected.

Spelling correction system is designed using System Development Life Cycle (SDLC). Systems development methods are methods, procedures, job concepts, rules that will be used as guidelines for how and what to do during this development. Method is a systematic way to do things. A step-by-step list of instructions for solving any instance of the problem is known as algorithm [10].

3.1.1 Initiation

Initiation is the initial planning stage for the designed system. At the stage of the initiation described the data needed in the design of spelling correction for text documents system .

There are two data that must be prepared first before step into the design of spelling correction system text documents, the first is the dictionary and the second is news articles. The data dictionary (Kamus Besar Bahasa Indonesia or KBBI) obtained manually with pdf form. Specification of data dictionary used can be seen in Table 1.

Table 1. Dictionary data specification.

Description	Specification
Source	http://jurnal-oldi.or.id/public/kbbi.pdf
Edition	Fourth (2008)
Document size	8.71 MB
Number of pages	1 490
Word count	426 774

We used news articles as training data that is consist of 5 000 Indonesian. News articles obtained from online news www.kompas.com. The news articles consist of various categories, the categories are regional, megapolitan, national, entertainment, soccer, economics, international, automotive, travel, property, technology, health, female, science and education.

3.1.2 Analyzing system

This section discusses the specifications or devices that will be used in the design of this system, the device is hardware and software. Hardware specifications (laptop) used in this design are :

1. Model : Acer Aspire E5-473G
2. Processor : Intel i5 7200U 2.5 GHz
3. Memory : RAM 8GB
4. Hard Disk : 1 TB
5. Graphics Card : NVIDIA GeFOrce 940MX 2GB

The software specifications used in this design are:

1. Operating System : Windows 10
2. Web Browser : Google Chrome
3. Text Editor : Padre and dreamweaver
4. Programming Languages: Perl and PHP

3.1.3 System design

The next step after analyzing system is system design. Design stage is a system workflow designed. This stage is divided into several sections. They consist of process and data flow design and application program interface design.

3.1.3.1 Design process

The design process describe the process of application program communicating with the System and how the system runs. The design of the process and the flow of data in the design is explained with flowchart and state transition diagram. Flowchart spelling correction system designed can be seen in Figure 1.

The state transition diagram illustrates how the system works through state. The state transition diagram illustrates the actions performed due to a particular event. Figure 2 shows the state transition diagram of the designed spelling correction system.

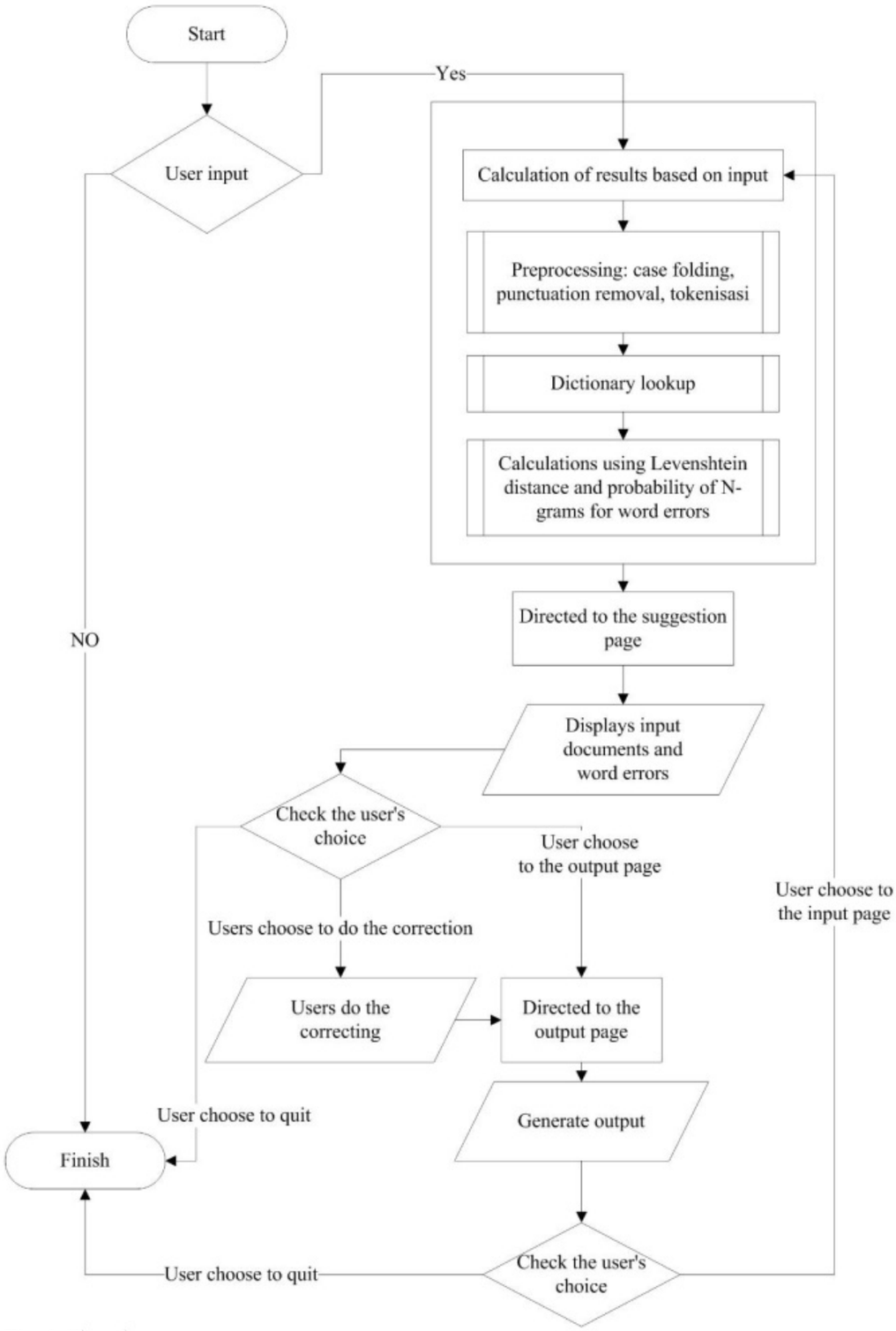

Fig. 1. Flowchart system.

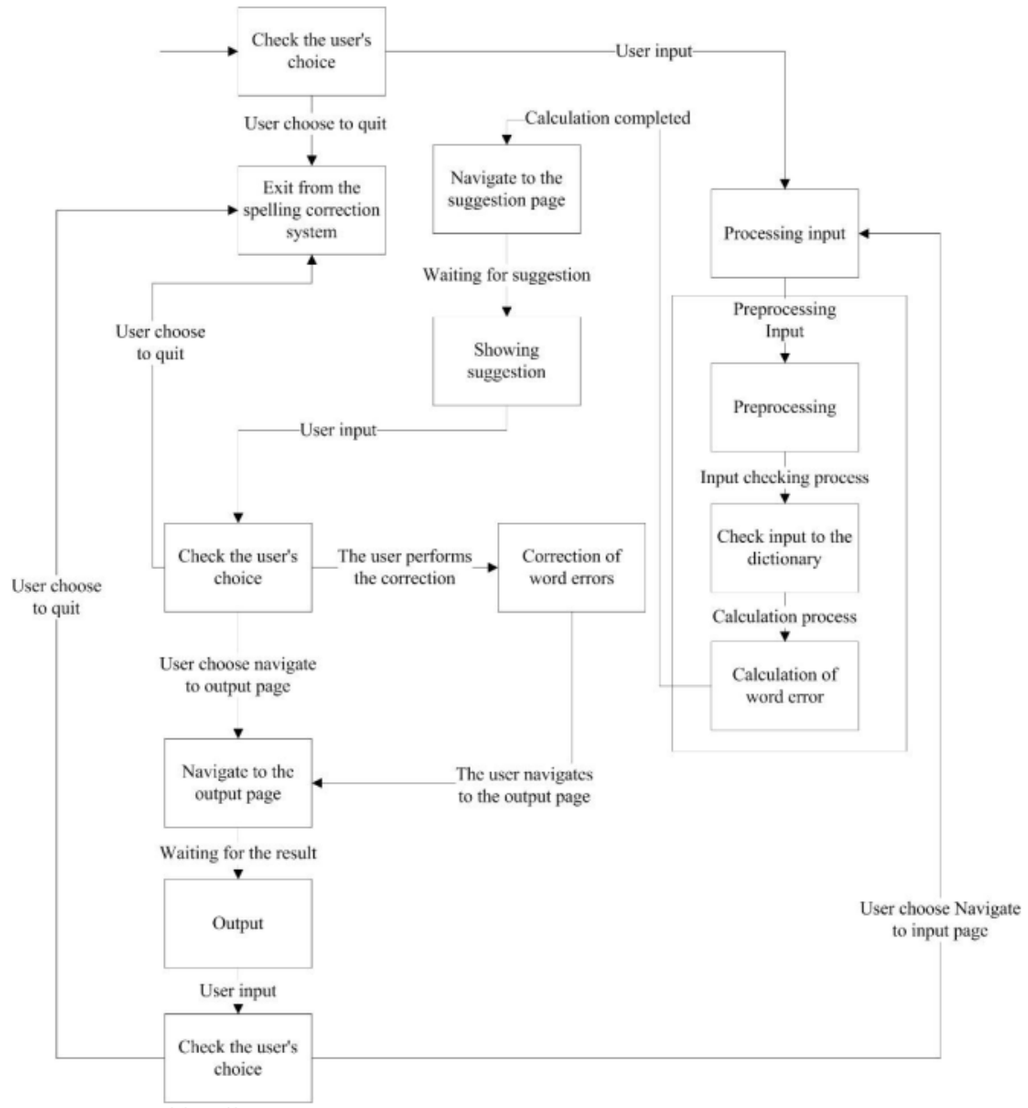

Fig. 2. State transition diagram.

3.1.3.2 Dialog design

In this section described what dialogues exist in the designed system. There are three dialogues on the designed system, the first is the front page dialogue, the second is the suggestion page dialogue, and the last is the output dialogue. More details can be seen in Figure 3 which shows the hierarchy of the designed spelling correction system.

Fig. 3. Hierarchy diagram program.

3.1.3.3 Interface design

Interface Design is a media that connects users and systems which designed to communicate. In this spelling correction system there are four modules:

(i). Front Page Module
This module is the main module of the web-service. User can input the text document which he/she want to check in this module.

(ii). Suggestion Page Module
This module displays the contents of the input document which already marked for word errors. User can perform both automatic or manual correction.

(iii). Manual Output Module
This module displays the correction results manually which could be downloaded by the user.

(iv). Manual Automatic Module
This module displays the correction results automatically which could be downloaded by the user.

3.1.3.4 Implementation system

The next stage is the system implementation. System implementation is a software adjustment with the system design that has been made before. The hardware and software of the server is required in the system implementation. Server acts as a mediator during the process of data transfer using internet connection. Implementation of the system will use a hosting server.

(iv) Result and discussion

4.1 Testing method

Testing process are performed on the website, the method used, and the results of the correction. The tests are performed by the developer to determine how well the method is used in the system, and the output results.

4.1.1 Website testing

Website testing are done on every module on the website. Testing performed for each component contained in the module to test whether it is running properly or not. The tests include the following modules:

4.1.1.1 Front page module

In the middle of the front page there is a "Pilih File" button to select a text document and next to the "Pilih File" button there is a textbox that displays the name of the text document. At the bottom of "Pilih File" there is "UPLOAD" button to upload the file.

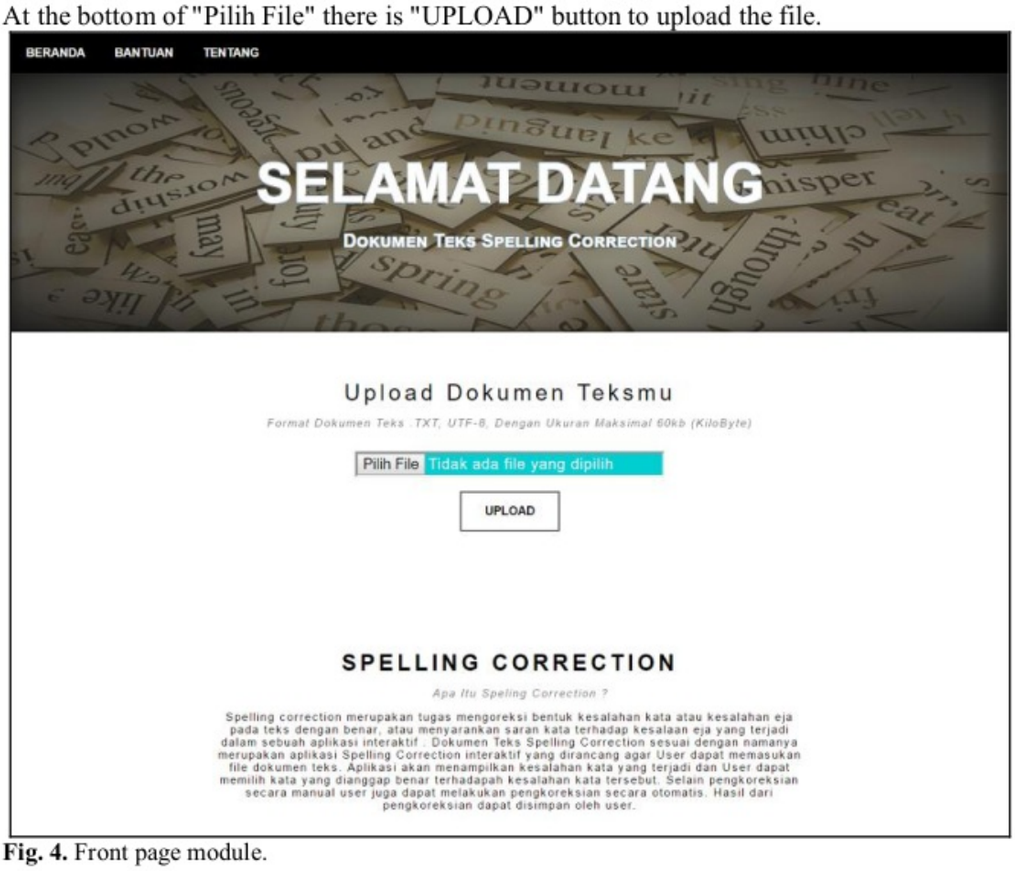

Fig. 4. Front page module.

4.1.1.2 Suggestion page module

There is a text area that displays the contents of a text document that has been uploaded by the user where the error word will be highlighted in red. If a highlighted word is clicked, it will create a suggestion word pop-up. The user then choose the word that is considered correct from the pop-up. When the word suggestion is selected, the highlighted word will be replaced with the suggested word that has been selected. Below the text area is a "KOREKSI MANUAL" button to navigate user to the manual output page module. Below the "KOREKSI MANUAL" button is the "KOREKSI OTOMATIS" button to navigate user to the automatic output page module.

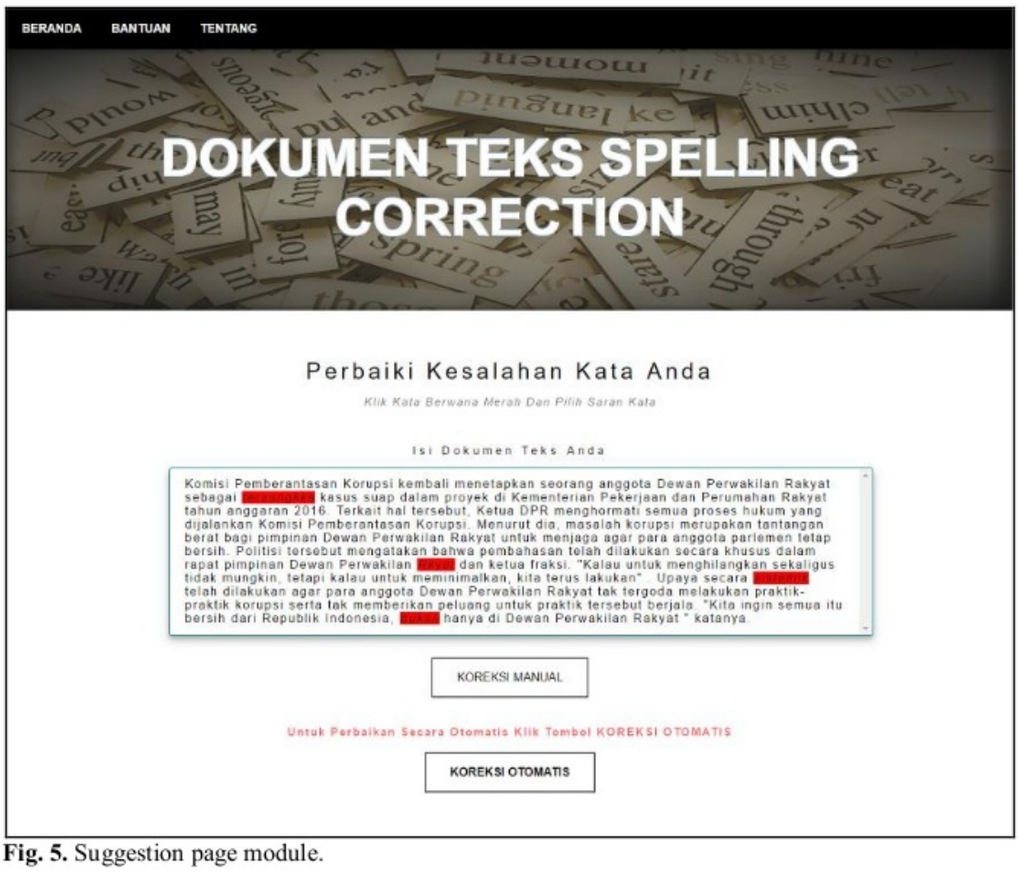

Fig. 5. Suggestion page module.

4.1.1.3 Manual output page module

The manual output page module contains the results of the manually performed correction. The result are displayed inside the text area, where the word error with a red highlight in the previous module will turn green as a mark that the word has been corrected. In this module, there is a "DOWNLOAD" button to download the result of manual correction into .txt file. After the "DOWNLOAD" button is clicked, it will create a thank you pop-up with an "OK" button to return to the front page module.

Fig. 6. Manual output page module.

4.1.1.4 Automatic output page module

The automatic output page module contains the results of the automatic correction. The results are displayed inside the text area. The highlight will turn into green if the error word has been successfully corrected, otherwise it will remains red. Below the text area there are "<" and ">" buttons. The ">" button is used to navigate to next corrected document's page and the "<" button is used to navigate to previous corrected document's page Below the "<" and ">" buttons is a "DOWNLOAD" button to download the results of automatic correction into .txt file. After the "DOWNLOAD" button is clicked, it will create a thank you pop-up with an "OK" button to return to the front page module.

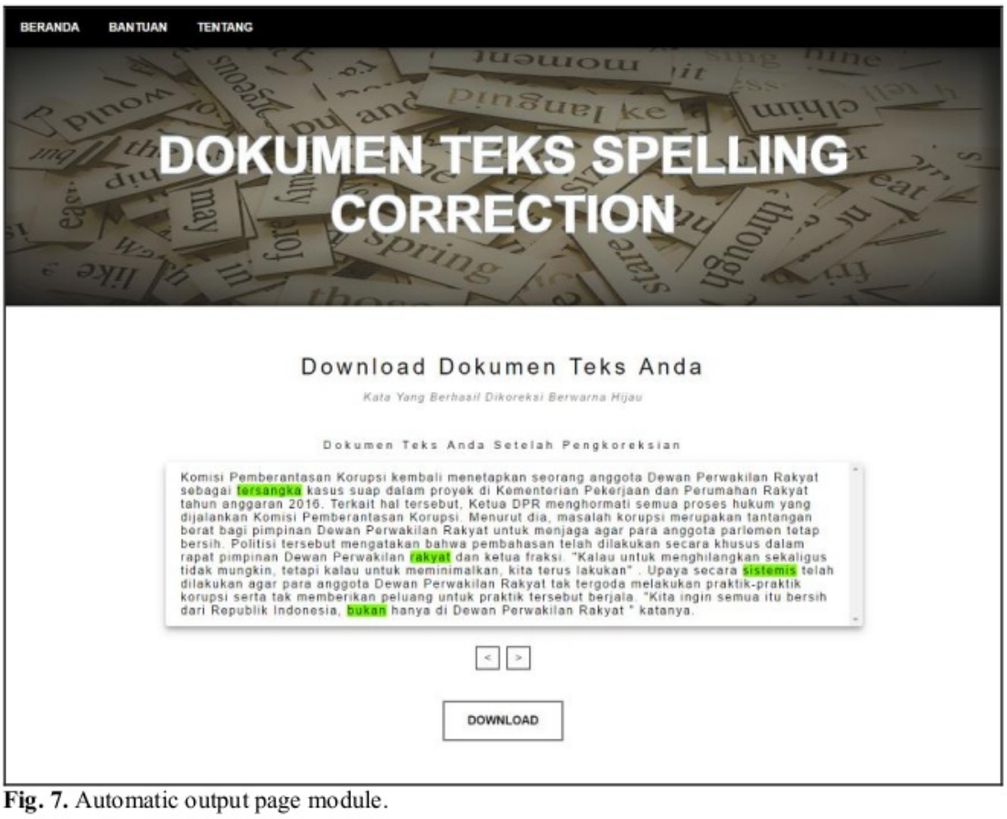

Fig. 7. Automatic output page module.

All functions in the modules have been working properly.

4.2 Perplexity testing

The testing result of N-grams method generated from training documents can be seen on Table 2. From the table, we can conclude that the smallest perplexity value is unigram (1.14 823 655), bigram (1.165 909), and trigram (1.167 692).

Table 2. Perplexity evaluation results.

N-Gram	Perplexity
Unigram	1.14 823 655
Bigram	1.165 909
Trigram	1.167 692

4.3 Correction hit rate and false position rate testing

At this stage the system will be tested against the results of the spelling correction by unigram, bigram and trigrams. The document used for testing is one of Indonesian text document from training document, which has been manipulated in such a way as to have a non-real word error.

The test was conducted on 15 documents, with different size. The highest correction hit rate on bigram is 75.36 %, trigram is 74.46 % and unigram is 64.59 %. While the highest total hit correction hit by bigram and trigram is 71.20 %. The false positive rate on unigram, bigram, and trigram has the same percentage of 4.43 %. The fastest processing time for spelling correction is unigram with 01:20.73 min, after that is bigram with 01:21.23 min, then trigram with 01:32.80 min.

There is also a test on correction hit rate and false positive rate for adjoined or sequenced word consist of two or three error words. The spelling correction results then compared between unigram, bigram and trigram. The highest corrected hit rate is bigram with 75 %, followed by unigram with 73.68 % and trigram with 69.73 %. From here, we can conclude that bigram is better than unigram and trigram for either adjoined or non-adjoined error words.

After that, there was a test on the modificated results in order to improve the correction hit rate on the designed spelling correction system. From the previous tests showed that bigram is the better N-gram method than unigram or trigram, therefore the tests and modifications are made to bigram. The comparison of results between correction hit rate and false positive rate correction spelling Bigram with FSA, without FSA, and modificated results can be seen in Table 3.

Table 3. Comparison of bigram test results with FSA, without FSA, and modifications.

Testing Category	With FSA	Without FSA	Modifications
Total documents	15	15	15
Total size document	255 kb	255 kb	255 kb
Total words	34 919	34 919	34 919
Total word error	316	316	316
Total words corrected correctly	225	276	270
Total words corrected wrongly	91	40	46
Total false Positive	1 436	1 436	1 436
Total correction Hit rate	71.20 %	87.34 %	85.44 %
Average correction hit rate	75.63 %	83.75 %	85.80 %
Total false positive rate	4.15 %	4.15 %	4.15 %
Average false positive rate	4.43 %	4.43 %	4.43 %
Total time spelling correction process	20:18.48 min	07:31:46.87 h	27:42.00 min
Average time of spelling correction process	01:21.23 min	30:07.12 min	01:50.80 min

5 Conclusions

Based on the test results, it can be concluded as follows :

- Text document spelling correction can resolve non-word error, using FSA and Levenstein distance methods.

- Bigram has greater correction hit rate than unigram and trigram, with the adjoining word error that is 75 %. While the non adjoining word error, bigram and trigram have the same correction hit rate that is 71.20 %.
- FSA method can be used to shorten the spelling correction processing time.
- The modification results show an increase in correction hit rate from 71.20 % to 85.80 %, but the average processing time becomes longer by 29.57 s.

References

1. L. Michelbacher. *Multi-word tokenization for natural language processing*. [PhD dissertation]. University of Stugartt, Stugartt, Germany (2013). pp. 27–28. https://d-nb.info/1046313010/34
2. Global Komputer. *Finite automata* [Online] from http://www.globalkomputer.com/Bahasan/Teori-Bahasa-dan-Otomata/Topik/Finite-Automata.html (2006). [Accessed on 10 January 2017]. [in Bahasa Indonesia]
3. J. Daciuk, D. Weiss. *Theoretical Computer Science,* **450**,7:10–21 (2012). https://www.sciencedirect.com/science/article/pii/S0304397512003787
4. N.M.M. Adriyani. *JELIKU*, **1**,1 (2012). https://ojs.unud.ac.id/index.php/JLK/article/view/2800
5. W. Zhang. *Comparing the effect of smoothing and n-gram order: Finding the best way to combine the smoothing and order of n-gram*. [PhD dissertation]. Florida Institute of Technology, Melbourne, Florida, pp. 1–6 (2015). https://pdfs.semanticscholar.org/ac6b/aabe681737c3baecf7315b629c7ab4c23577.pdf
6. D., Jurafsky, J.H. Martin. *Speech and language processing volume 3* [Online] from http://www.cs.colorado.edu/~martin/SLP/Updates/1.pdf (2014) [Accessed on 18 December 2016]
7. T.M. Allam, H. Abdelkader, E. Sallam. *IAJeT*, **4**, 2:94–102 (2015). http://www.iajet.org/iajet_files/vol.4/no.2/5-58594_formatted.pdf
8. D. Fossati, B. Di Eugenio. A mixed trigrams approach for context sensitive spell checking. In: *Computational linguistics and intelligent text processing*. A. Gelbukh. (Eds). CICLing 2007. Lecture Notes in Computer Science. Volume 4394. Heidelberg: Springer. pp. 623–633. https://link.springer.com/chapter/10.1007/978-3-540-70939-8_55
9. F. Kurniawan. *Implementasi algoritme fonetik untuk koreksi ejaan bahasa indonesia* [Implementation of phonetic algorithms for Indonesian spelling corrections] [Online] from http://studylibid.com/doc/94649/implementasi-algoritme-fonetik-untuk-koreksi-ejaan (2010). [Accessed on 14 January 2017]. [in Bahasa Indonesia]
10. B. Miller, D. Ranum. *Problem Solving with Algorithms and Data Structures*. [Online] from https://www.cs.auckland.ac.nz/courses/compsci105s1c/resources/ProblemSolvingwithAlgorithmsandDataStructures.pdf (2013). [Accessed on 25 January 2017].

ICESTI2017Spelling

ORIGINALITY REPORT

18%	14%	11%	13%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	docplayer.net Internet Source	4%
2	Submitted to University of Greenwich Student Paper	2%
3	Submitted to Pasundan University Student Paper	2%
4	Submitted to Universitas Diponegoro Student Paper	1%
5	en.m.wikipedia.org Internet Source	1%
6	Submitted to Universiti Teknologi Malaysia Student Paper	1%
7	"Intelligent Computing", Springer Science and Business Media LLC, 2019 Publication	1%
8	Submitted to University of Edinburgh Student Paper	1%
9	maltesas.my Internet Source	1%
10	Submitted to Universiti Kebangsaan Malaysia Student Paper	1%
11	Firda Maryana, Ana Kurniawati, Dina Agusten. "Term Frequency Method For Automated Text Summarization Application Of Indonesian News Article", 2018 Third International Conference on Informatics and Computing (ICIC), 2018	1%

Publication

12	Said Gounane, Mohammad Fakir, Belaid Bouikhalen. "Handwritten Tifinagh Text Recognition Using Fuzzy K-NN and Bi-gram Language Model", International Journal of Advanced Computer Science and Applications, 2013 Publication	<1%
13	M Firdaus, M H Mud'is, I Nurjaman, I Fauzia, B Busro. "Fuzzy string implementation matching on android-based encyclopedia and anatomy quiz", Journal of Physics: Conference Series, 2019 Publication	<1%
14	Moh Noor Al-Azam, Mochamad Mizanul Achlaq, Aryo Nugroho, Adri Gabriel Sooai, Aris Winaya, Maftuchah. "Broadcasting the Status of Plant Growth Chamber using Bluetooth Low Energy", MATEC Web of Conferences, 2018 Publication	<1%
15	Submitted to University of Wales Swansea Student Paper	<1%
16	Submitted to King Saud University Student Paper	<1%
17	Submitted to University of Hong Kong Student Paper	<1%
18	s3.amazonaws.com Internet Source	<1%
19	Submitted to University of Bristol Student Paper	<1%
20	"Natural Language Processing and Information Systems", Springer Science and Business Media LLC, 2010 Publication	<1%

21	journals.agh.edu.pl Internet Source	<1%
22	research.ijcaonline.org Internet Source	<1%
23	bestlaptopsworld.com Internet Source	<1%
24	repository.uhamka.ac.id Internet Source	<1%
25	M A Muchtar, I Jaya, Manguji Nababan, U Andayani, Lisa Noprianti Siregar, Erna B Nababan, Opim S Sitompul. "Separation of Basic Words in Angkola Batak Text Documents using Enhanced Confix Stripping Stemmer Case: Mandailing Ethnic", IOP Conference Series: Materials Science and Engineering, 2019 Publication	<1%
26	Deborah A. Stoffer, L. Gwenn Volkert. "Exploring chaos automata for protein sequences", 2008 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, 2008 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		