ISSN: 0067-2904

Active Verb Spell Checking *Mem- + P* in Indonesian Language Using the Jaro-Winkler Distance Algorithm

Dimas Aryo Anggoro, Inayah Nurfadilah
Informatics Department, Universitas Muhammadiyah Surakarta, Indonesia,

Received: 23/2/2021 Accepted: 11/6/2021 Published: 30/4/2022

Abstract
The Indonesian language is used as a means of communication, including written communication. Unfortunately, many mistakes are found in Indonesian language writing, such as the writing of active verbs with the prefix *mem*- followed by the letter P. This problem can be addressed with the spell-checking method. Spell checking is a process in computer programs to check the spelling of each word in electronic text or documents in the correct order. To better the active verb this study used the Jaro-Winkler Distance algorithm. Meanwhile, for system development, the Iterative Waterfall method was used. The system output is active verbs of *mem*- + P which is standardized according to the Great Dictionary of the Indonesian Language of the Language Center (KBBI) and the morphological rules of the Indonesian Language Spelling System. The accuracy value of the algorithm was 80% for common mistakes, 95% for specific mistakes, and 80% for correctable mistakes with word suggestions.

Keywords: Indonesian Language, spell checking, autocorrect, active verbs, Jaro-Winkler distance algorithm.

1. Introduction

The Indonesian language is the official language of Indonesia, as stated in the 1928 Youth Pledge. The Indonesian language is used as a means of verbal communication, oral and written. In written communication, The Indonesian language has a scientific context function related to the use of spelling, correct word writing, and the use of various languages. Unfortunately, many mistakes are found in The Indonesian language writing [1]

In research [2], there were writing mistakes with a percentage of 43.15% or 183 items in the use of letters, 145 items or 34.20% mistakes in words writing (including incorrect prefix/suffix), 68 items or 16.04% mistakes in punctuation marks, and 28 items or 6.6% mistakes in loanwords. These various mistakes are on the result of the use of non-standard words according to the Great Dictionary of the Indonesian Language of the Language Center (KBBI). Therefore, considering whether a word is standard or non-standard, word class, the separation, and the combination of prepositions, and verbs in The Indonesian language must be meticulously studied by using the The Indonesian Language Spelling System (EBI) to become a reference of good and correct the Indonesian language [3]. The verbs discussed are those using affixes according to morphological rules.

However, the problem frequently arising is the people's lack of understanding of EBI, including writing active verbs according to the prevailing rules. Consequently, The Indonesians and foreigners often write standard verbs wrong. Even in scientific writing, errors in writing verbs still appear according to morphological rules [4], especially for the active

*Email: dimas.a.anggoro@ums.ac.id

verb *mem- + p,* which has various morphological rules. This happens because verb additions have complex morphological rules. Consequently, conventional improvements are often complicated for people who are not Indonesian language experts. Therefore, an instrument is required to help the public editing articles so that active verbs contained become standard according to KBBI 5th edition (latest edition) and EBI. The instrument commonly used is the spell checker.

The spell checker is a process in a computer program to check the spelling of each word in electronic text or documents in the correct order [5]. Due to the problems described previously, this study aims to create a spell checker application to change active verbs with the prefix *mem-* followed by a root word the first letter of 'p', in which 'p' will experience fusion process and become letter 'm'. The fusion process occurs in words that can be used as verbs but does not occur in cluster words (two consonants).

Therefore, this program can detect writing mistakes and correct them by autocorrect or give a suggestion for correction. After detection, this program will automatically change the active verb with prefix *mem-*, which is not standard, becomes standard. The initial input is in the form of words/sentences embedded in the input column, then the program runs and produces standard verbs that refer to KBBI. After generating standard verbs, the output can be copied and stored in the clipboard.

This program can be used by various groups. Besides The Indonesian language use, this system can also be helpful by foreign speakers learning Indonesian language/BIPA (Indonesian Language for Foreign Speakers). In the BIPA program, students are required to master writing skills [6]. This system will assist BIPA/non-BIPA students in writing standard verbs according to morphological rules.

The method to change the active verb is the Jaro-Winkler Distance (JWD) algorithm. This algorithm was chosen becuase it has the highest Mean Average Precision (MAP) value compared to other string searching algorithms. In a study [7], the overall MAP JWD value obtained 0.87, with details; 0.95 for correcting letter-changing mistakes, 0.92 for correcting letter-omission mistakes, 0.90 for correcting letter-addition mistakes, and 0.70 for correcting letter replacement mistakes. Whereas, the Hamming Distance, Levenshtein Distance, and Damerau Levenshtein Distance algorithms have a MAP value of 0.46, 0.74, and 0.85, respectively.

To date, there is no similar study found regarding the development of spell checkers to replace active verbs with the non-standard prefix *mem-* into standard active verbs. Relevant previous research searching for the root words contained in the KBBI with prefix input words [8]. Meanwhile, in this research, the input and output contain words with prefixes, yet it is improved following standard words. The difference between the previous research and this research lies in the final results.

2. Method

The method used in developing spell checkers is the Iterative Waterfall. The iterative waterfall is a combination of the waterfall method in the iterative method (repetition). This method is preceded by the development of some part of the total system implementation and slowly increasing functionality through the repetition steps that shall be optimized [9]. In the same study, it was concluded that this method is suitable for low-risk projects. Iterative Waterfall was chosen because it is flexible to fulfill the needs of this analysis and can be developed in parallel [10]. Iterative Waterfall shown in Figure 1:

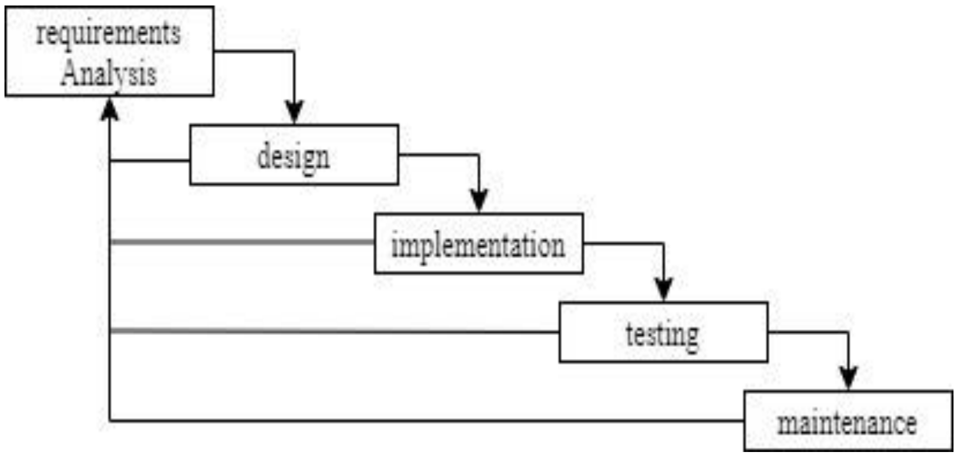


Figure 1-Iterative Waterfall method

2.1 Requirements' Analysis

Requirements' analysis is the first stage in the analysis, which aims to understand the required system and its limitations. Information on the development of this system uses the morphological theory of Indonesian Language affixes.

2.1.1. Morphological Theory

Morphology is a branch of linguistics that studies the details of word arrangement structurally on the morphemes that form it [11]. Several morphological can changes occur, one of which is affixation used as a theoretical basis in this study. Affixation refers to a series of changes that root or lexeme undergoes so that lexeme becomes a word, be it a simple word or a complex word [12]. In this case, the use of prefix (affix at the beginning of a lexeme) *mem-* is an affixation process.

Mem- is an allomorph of the morpheme *meng-* to express an active verb, either transitive or passive. *Menge-* as a morpheme undergoes changes to form *mem-* due to the morphophonemic process, which is the process of phonological changes [13]. Table 1 shows the state of phonological change *mem-* followed by a root starting with P under the following conditions:

Table 1- Active verb affixation starting with P

Condition	**Morphophonemic**	**Example**		**Root Word**
		Input	**Output**	
Vowel	There is fusion	mempaksa	memaksa	paksa
Cluster	There is no fusion	memproduksi	memproduksi	produksi
Specific words	There is no fusion	mempunyai	mempunyai	punya

Table 1 demonstrates how the phoneme *mem-* will experience several changes when followed by a word starting with P. If the letter P is followed by a vowel, a fusion process will occur so that the P at the beginning of the word disappears and phoneme *mem-* automatically followed by a vowel. When it is followed by a cluster or two consonants, there is no fusion so the P at the beginning of the word still appears. It also applies to specific words whereط there is no fusion considering the effectiveness/ ineffectiveness of the word given the presence or the absence of fusion, it is no longer related to the prevailing morphological rules.

2.2 Design

Design is a form of translation of needs analysis before the coding process. In this research, the design stage utilized use case diagram and activity diagram. Use Case Model is a representation of the interaction between users and the system. Activity Diagram model is a workflow representation of system activities.

2.2.1 Use Case Diagram

The Use case diagram is a part of the UML to specify all scenarios and describe actions that may be taken by users [14] [15]. This study only consists of one actor, namely a user who can access texts and standard verbs, as shown in Figure 2:

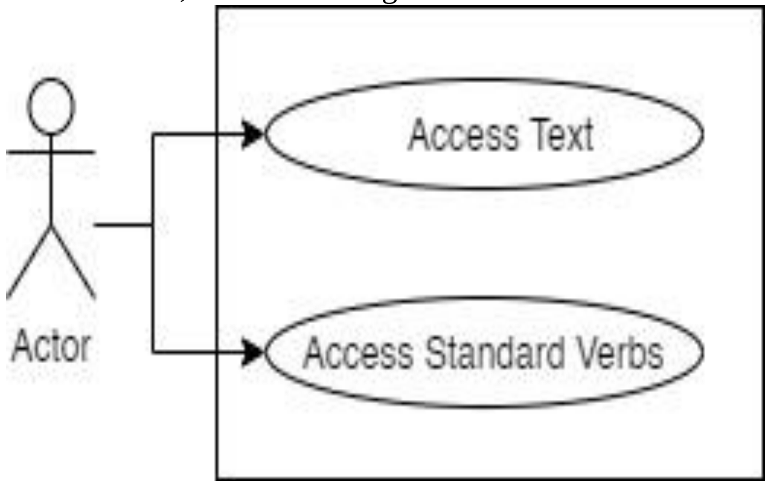


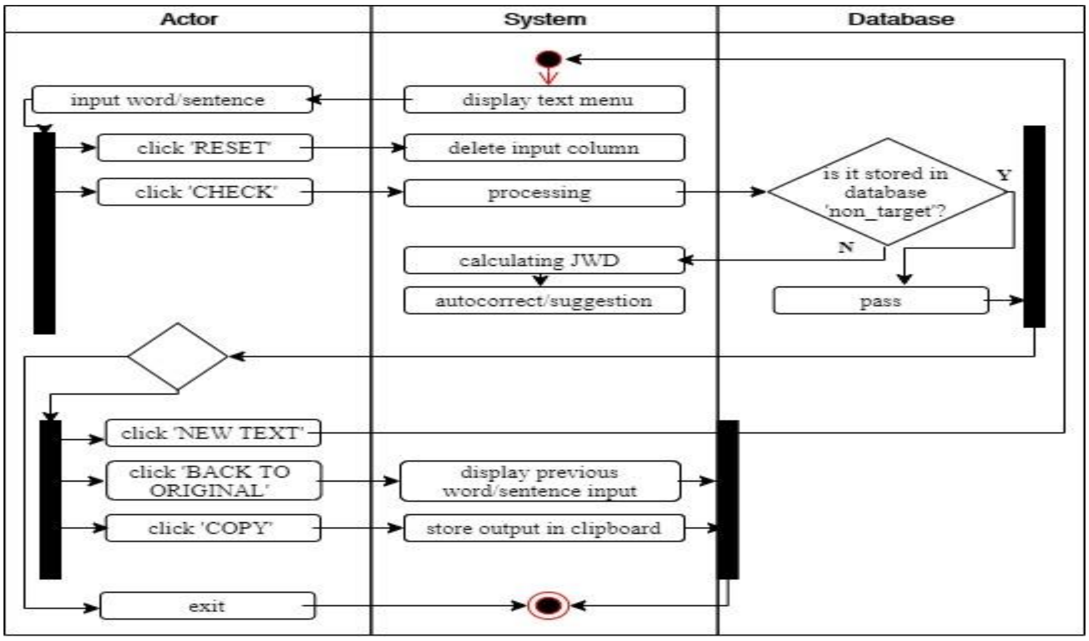
Figure 2-Use case diagram system

2.2.2 Activity Diagram

An Activity Diagram depicts the main scenario functionally including variants and failure processing [16]. In this system, there are two activity diagrams based on the displayed page, namely accessing text and accessing standard verb table.

1. Access Text

Accessing text is a page for correcting the verb *mem-* + P, as well as the home page of the system. In the input field, the user can enter a word/sentence and opt to click the "CHECK" or "RESET" button. If the "CHECK" button is clicked, the system processes user input by preprocessing it, then it will be matched with the database. If the word is stored in the database that includes the verb *mem-* + p, the system will autocorrect it or give a suggestion by picking a word from the database. Users may select the "RESET" button to clear the input field or click the "COPY" button to copy the output in column text. The Activity Diagram of access text is shown in Figure 3:



Figuree 3-Activity diagram accessing text

2. Accessing Standard Verb Table

This page is a sub-menu in the system. On the page of the standard verb table, the list on the system is exported from the database. Then the user can view or search for certain words by entering keywords in the text input "search". The keywords the user enters can be either standard verbs or root words of the verb.

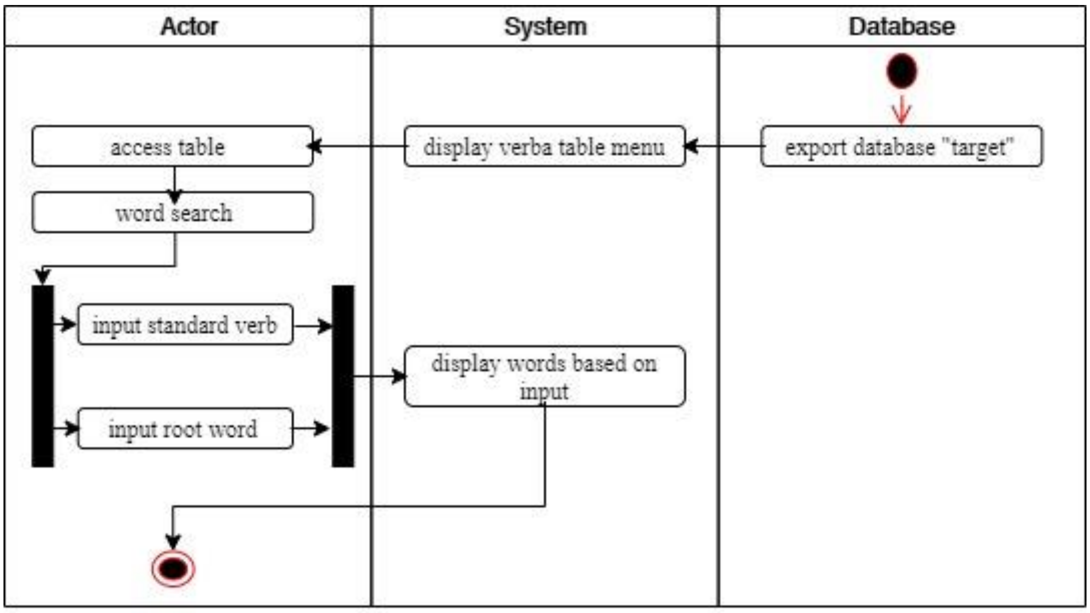


Figure 4-Activity diagram accessing files

2.3 Implementation

Implementation initiates with the writing of the program's code. The program code in this study used the PHP programming language with the Laravel framework. PHP is an open-source programming language that runs on various platforms. Meanwhile, the framework used is Laravel. Laravel is a framework designed to optimize PHP code based on the Model View Control (MVC) pattern. The MVC pattern allows modifying the program code separately [17].

The implemented program code follows a flow chart algorithm according to the data at the design stage. The following is a flow chart for the spell checker of active verb based on the data that was described above shown in the Figure 5:

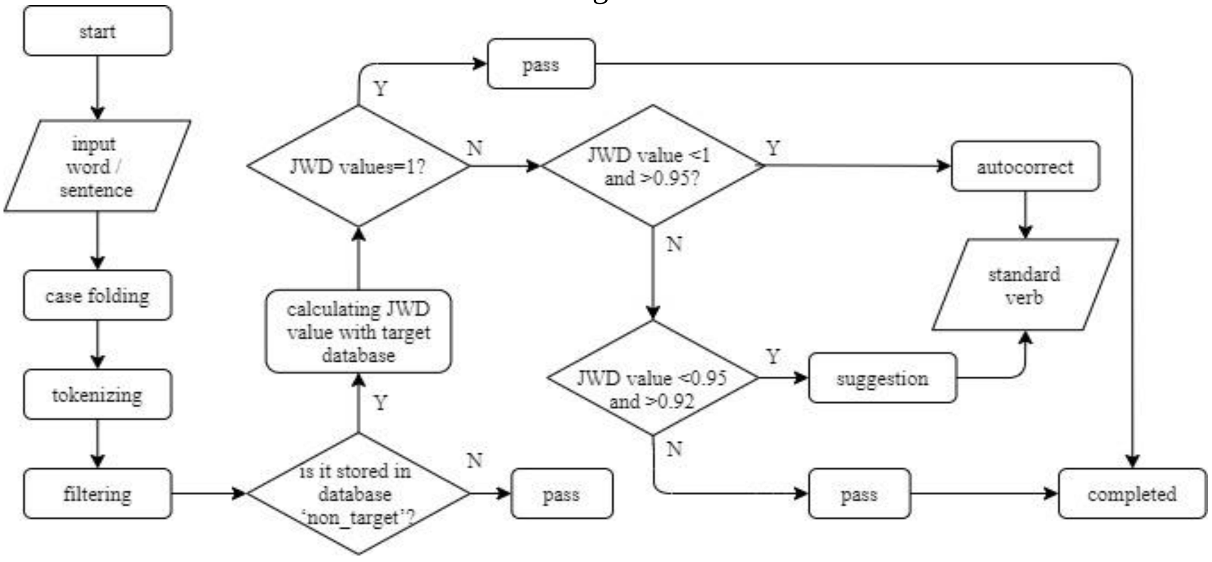


Figure 5-System flow chart

From the flow chart in the Figure 5, once the user has entered a word/sentence in the input column, the system will proceed to the preprocessing stage before calculating JWD. Preprocessing consists of three stages: tokenizing, filtering, and case folding.

2.3.1 Preprocessing

Preprocessing is the initial stage for verbs that can be fixed into standard words.

1. Tokenizing

Tokenizing is used to divide words from the text into a sequential set of morphemes [18]. This is closely related to morphological analysis [19]. This initial stage aims to cut each word from the data that has been inputted, usually using spaces. In this study, tokenizing used spaces to cut sentences into arrays. Input Example: *Saya Memang Mempengaruhi Untuk Mempaks*, tokenizing result is as shown in Figure 6:

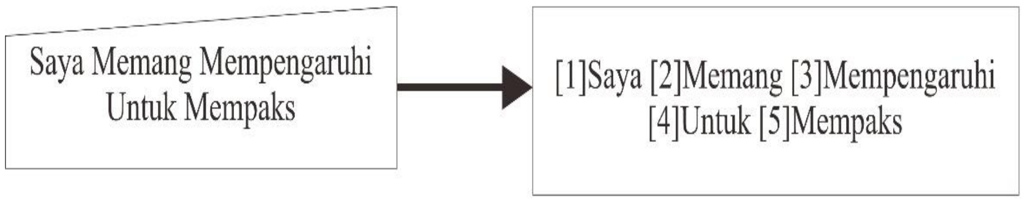

Saya Memang Mempengaruhi Untuk Mempaks
[1]Saya [2]Memang [3]Mempengaruhi [4]Untuk [5]Mempaks

Figure 6-Tokenizing process

2. Case folding

Case folding is the process of converting all letters to lowercase or uppercase (from "a" to "z"). All characters excluding letters are removed and cleaned so that all characters with accents are converted into ASCII characters. This is done because the Jaro-Winkler Distance process is case sensitive, so the letter case may affect the distance calculation results proximity. In this study, all letters were converted into lowercase [20]. The sentence/word input at this stage is the result of the tokenizing stage as shown in the Figure 7:

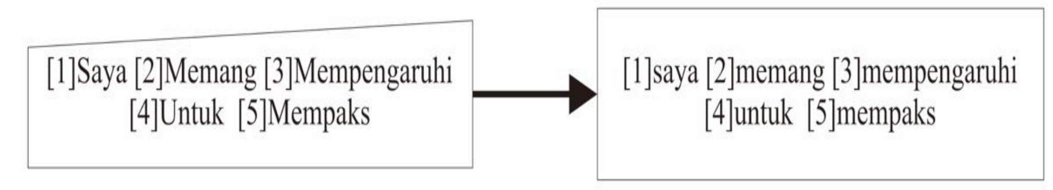

[1]Saya [2]Memang [3]Mempengaruhi [4]Untuk [5]Mempaks
[1]saya [2]memang [3]mempengaruhi [4]untuk [5]mempaks

Figure 7- Case folding process

3. Filtering

Filtering is a stage to pick important words from tokenizing results. It may use a stoplist algorithm (removing unnecessary words), or a wordlist (storing important words) [21]. In this study, filtering of a wordlist was conducted in two stages.

The first step is to capture all words starting with *mem*, this is the initial process to determine whether the word starting with *mem* is included in a verb with the first letter "p" or not. The first stage of the filtering process is shown in Figure 8:

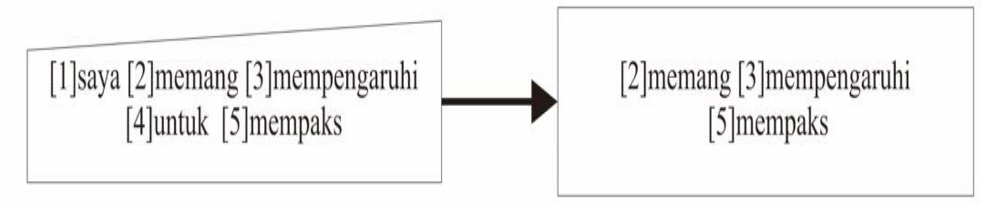

[1]saya [2]memang [3]mempengaruhi [4]untuk [5]mempaks
[2]memang [3]mempengaruhi [5]mempaks

Figure 8-Stage 1 of filtering process

The second stage of filtering is to capture words starting with *mem*- that are included in the verb class. Apart from verbs, words starting with *mem* are discarded. Besides, this second stage of filtering only captures non-standard verbs *mem* + p. Every word passing the second stage of filtering process is corrected into a standard verb. The second stage of filtering process is displayed in the Figure 9:

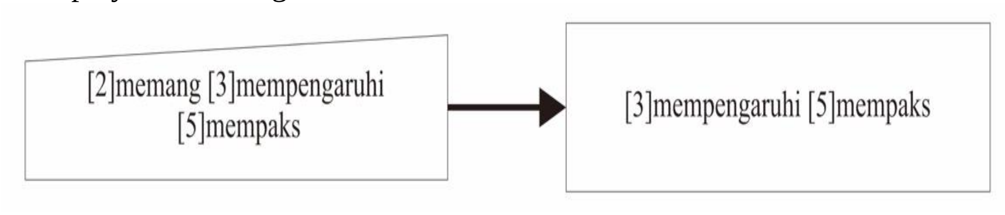

[2]memang [3]mempengaruhi [5]mempaks
[3]mempengaruhi [5]mempaks

Figure 9-Stage 2 of filtering process

2.3.2 Database

Based on the data on the needs analysis, the database required is the words starting with *mem*-. Database development used MySQL, which was linked with PHP program code. This study employed a database that has two entities. The first entity is named NonVerba, which contains words other than *mem*- + P with the attributes of ID_NonTarget and Kata_NonTarget. The data on the Kata_NonTarget attribute comes from (KBBI) 5th edition – all words starting with *mem*- and are not a verb. The second entity is Verb, containing the word *mem*- + P with attributes of ID_Target, Kata_Target, and Kata_Dasar. The data on the Kata_Target entity comes from KBBI 5th edition – all lexeme verbs *mem*- starting with P. While the Kata_Dasar entity comes from the Noun class of KBBI 5th edition.

2.3.3 JWD Calculations

After the preprocessing stage, the system performs filtering until it retrieves words that needs correcting. Correction uses JWD by calculating the distance. Distance calculation functions to match words with those in the system database. If a match is found, word correction can be performed immediately. If there is no match, the word will be skipped. JWD has 3 basic components: calculating the length of the string/word, finding an equal number of characters in two strings, and finding the number of transpositions.

The Jaro distance (d_j) between string one (s1) and string two (s2) is calculated using following formula:

$$d_j(s1,\ s2) = \frac{1}{3}\left(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m}\right) \quad (1)$$

where m = number of matched characters between s1 and s2; $|s1|$ = length of string 1; $|s2|$ = length of string 2; t = number of transpositions.

The transposition value (t) can be obtained from the following formula:

$$\left(\frac{\max(|s1|,|s2|)}{2}\right) < 1, \quad (2)$$

A high Jaro value indicates a greater similarity between two words. Jaro value of 0 denotes those words are different and a value of 1 implies they are similar [22].

JWD uses a prefix scale (p) which gives a higher level of assessment, and prefix length (l) states the length of the prefix, which is the same character length of the string being compared until the difference is detected. If strings s1 and s2 are compared, then JWD (d_w) value is:

$$d_w = d_j + \left(lp\left(1 - d_j\right)\right) \quad (3)$$

where d_j = Jaro distance for s1 and s2; $l =$ length of the common prefix at the beginning of the string, maximum value of 4 characters; $p =$ constant of scaling factor. The standard p value according to Winkler is p = 0.1

The following is an example of the JWD calculation. If the strings are s1 MARTHA and s2 MARHTA then: m = 6; $|s1|$ = 6; $|s2|$ = 6. The characters that are replaced are only T and H, then t = 1. Thus, the value of d_j is:

$d_j(s1,\ s2) = \frac{1}{3}(\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6}) = 0.944$

Pay attention to the arrangement of s1 and s2. If the value of l = 3, and the constant value of p = 0.1 so the value of d_w is:

$d_w = 0.944 + (3\ \times 0.1\ (1 - 0.944)) = 0.951$

2.4 Testing

At the testing stage, integration is administered so that the system runs properly according to planning. In this study, testing was conducted using two methods: comprising unit testing and algorithm testing. Unit testing was carried out to find out that each function in the system is running according to plan [23]. In this system, the function or classes that will be executed are preprocessing class, JWD calculation class, and output class. Meanwhile, algorithm testing is conducted to determine how well the algorithm used in the system runs.

For algorithm testing, the accuracy value used is accuracy. To determine the accuracy value, the information retrieval measurement method uses a table confusion matrix. Tabulation of this table is a classification model based on the count of test records, then the model predicts true or false. Table 2 shows the confusion matrix for binary classification problems.

Table 2- Confusion matrix for class classification problems

Predicted class	True Positive (TP)	False Negative (FN)
	False Negative (FN)	True Negative (TN)

Once confusion matrix table is created, the accuracy value can be obtained with the following formula:

$$Accuracy = \frac{TP+FN}{TP+FP+FN+TN} \tag{4}$$

where TP = The result of the prediction is positive and matches the positive target; FN = The result of the system prediction is negative but the target result is positive; FP = The result of the system prediction is positive but the target result is negative; TN = The result of the system prediction is negative and matches the negative target.

To determine the accuracy value, it is important to calculate the threshold value needed to be categorized as a word that must be corrected, then to be converted into standard verbs. The determination of the correct threshold greatly influences the performance of the identification method [24]. In this study, the threshold for autocorrect was 0.95, while for suggestion was 0.92.

2.5. Maintenance

Maintenance is the final stage in the form of the installation and system repair process if bugs/errors found are anywhere at the testing stage. Maintenance is carried out following the release of the latest version of KBBI and changes to the morphological rules of EBI.

3. Results and Discussion

3.1 Results

The method for developing this system used iterative waterfall, which was carried out sequentially. To simplify the display creation, this system also used the bootstrap CSS library.

On the home page, the system displays the input column with “CHECK” and “RESET” buttons. The “CHECK” button will display the output page in the form of a word/sentence with corrected verb *mem-* + p.

Once an input states that the word is included in a verb that should be corrected, the system will automatically correct it according to the JWD calculation. For instance, Table 3 shows the input which contains categories of verbs with a vowel after p, consonants after p (cluster), and specific verbs. Then the following conditions may occur:

Table 3- Correction of non-standard verbs into standard verbs the system

Condition	Input	Output	Standard Verb	JWD Values	Type of Mistakes	Type of Correction	Information
Vowel	Mempaksa	memaksa	memaksa	0.9708333	Letter Addition	*Autocorrect*	Letter 'p' after prefix 'mem-' disappears due to the fusion process
Cluster	memproduks	memproduksi	memproduksi	0.9818181	Letter Omission	*Autocorrect*	Letter 'p' after prefix 'mem-' disappears due to the absence of fusion process
Specific Words	mempunyai	mempanyui	mempunyai	0.9777778	Letter Replacement	*Autocorrect*	Letter 'p' after prefix 'mem-' disappears due to the absence of fusion process

3.1.1 Preprocessing

The preprocessing stage facilitates JWD calculation so that the result is more accurate as it converts into the same format.

1. Tokenizing

The tokenizing stage in this study used space as the separator between one word and another. The words are stored in an array variable which will be compared to an array in the database. Tokenizing used the "explode" function in PHP.

2. Case folding

Case folding in this study matched the format through lowercase. To generate a lowercase word, this study used PHP "strlower" function.

3. Filtering

Filtering was carried out in two stages. The first step was filtering the input words starting with *mem-* followed by any letter using "stripos" function, with the value of the boolean variable *mem-* is false. Meanwhile, the second filtering was capturing words excluded from non-target using the 'array_dif' function.

3.1.2 Database

There were 1,398 non-verb words and 667 verb words *mem* - + p. These words are taken from KBBI 5th edition.

3.2 Testing

In this study, testing utilized unit testing and algorithm testing.

3.2.1 Unit Testing

Unit testing was performed by the system developer by ensuring that the program in each system class is running well. The results of unit testing are as follows:

1. Preprocessing class

In this class, sentence input was processed from tokenizing, case folding, to filtering (filtering one and two). In the preprocessing class, the system appeared to run well. Yet, once input was in form of a text containing more than one paragraph, the system could not perform case folding.

2. Jaro-Winkler Distance class

In this class, testing was conducted by calculating d_w value for word input that had passed the preprocessing stage with the target database. The d_w calculations applied to this system were similar to those in pure d_w and manual d_w programs.

3. Output class

From the sentence input: *Saya Memang Mempengaruhi Untuk Mempaks*, word detected is "*mempengaruhi*" and "*mempksa*", thus when the user clicked "CHECK" button, the output of those two words had different signs. In the word *mempengaruhi*, the font became bold as a sign that it underwent autocorrect process by calculating the JWD value ≥ 0.95 d_w < 1. Whereas the font of *mempksa* word was underlined as a sign that the word experienced a suggestion process because the JWD value was ≥ 0.92 d_w < 0.95. The suggested words are shown in the table below the output column. Suggested words displayed were sorted in ascending alphabetical order.

3.2.2 Algorithm Testing

At the initial stage of this test, the relevance judgment was used to determine whether the system could display the standard verbs according to its threshold value or not.

Autocorrect evaluation is divided into two categories: common mistakes and specific mistakes. Common mistakes contain word inputs that usually occur in society, which is people’s unconsciousness of the fusion of verb *me*- + P, by the addition of 'p' after the prefix *mem*.

Meanwhile, specific mistakes are divided into four categories, comprising mistakes due to the letter addition (word input is added by one letter randomly), writing mistakes due to letter omission (word input is deleted by one letter randomly), writing mistakes due to letter replacement (word input is replaced by one random letter), and writing mistakes due to letter exchange (word input is added one letter randomly).

Apart from auto-correction, algorithm evaluation was also done for suggestion. Each category contained 15 words taken from the least number of characters to the largest, consisting of 6-20 characters. Each word was also taken randomly. Thus, the total of the evaluation was 15 words for common mistakes, 60 words for specific mistakes, and 15 words for suggestion. Table 4 presents the results of algorithm testing of the autocorrect, in which 85% for JWD accuracy in common mistakes, 95% for specific mistakes, and 80% for suggestions.

Table 4- Verb correction accuracy values

Autocorrect		***Suggestion***
Common Mistakes	**Specific Mistakes**	
$\frac{12}{15}\ x\ 100\% =$ 80%	$\frac{57}{60}\ x\ 100\% =$ 95%	$\frac{12}{15}\ x\ 100\% =$ 80%

3.3 Maintenance

Maintenance has not been conducted because the latest version of KBBI (above 5th edition) has not been released.

4. Conclusion

In this study, it can be concluded that this system can improve the non-standard *mem + p* active verbs to be standard *mem + p* active verbs. The autocorrect applied to words with 1-2 characters difference, while offering suggestions for words with 3-4 characters difference. In this system, the threshold JWD value was 0.95 for autocorrect, and 0.92 for suggestion. These values become determinants of the accuracy of this system that can correct word errors with several categories. However, this system is still not suitable for correcting verbs with confixes. This is because it does not apply a stemming algorithm to separate the root or derivative words, so any words that are entered in the system are considered the same category.

References

[1] E. Kuntarto, *Bahasa Indonesia Untuk Perguruan Tinggi*, vol. 3. 2013.

[2] B. D. Nurwicaksono and D. Amelia, "Analisis Kesalahan Berbahasa Indonesia Pada Teks Ilmiah Mahasiswa," *AKSIS J. Pendidik. Bhs. dan Sastra Indones.*, vol. 2, no. 2, pp. 138–153, 2018, doi: 10.21009/aksis.020201.

[3] M. E. Palupi, "Kesalahan Penulisan Kata Bahasa Indonesia Pada Kain Rentang dan Papan Iklan di Tempat Umum," *Wanastra J. Bhs. dan Sastra*, vol. 10, no. 2, pp. 13–20, 2018, doi: 10.31294/w.v10i2.3810.

[4] B. A. Syafi'i, I. K. Niha, and S. Nisaa', "Analisis Kesalahan Morfologi Dalam Penulisan Makalah Mahasiswa Hukum Ekonomi Syariah Iain Surakarta," *J. Penelit. Hum.*, vol. 22, no. 1, pp. 31–46, 2021, doi: 10.23917/humaniora.v22i1.8153.

[5] R. Kuc, *Solr Cookbook - Third Edition*, 3rd ed. Birmingham B3 2PB, UK: Packt Publishing Ltd, 2015.

[6] J. Nasution, "Analisis Kesulitan Pemelajar Bahasa Indonesia Bagi Penutur Asing (BIPA) Di Samarkand State Institute of Foreign Languages (SAMSIFL), Uzbekistan Pada 4 Keterampilan (Skills) Berbahasa," *J. Ilm. AQUINAS Terbit*, vol. 3, no. 1, pp. 27–40, 2020, doi: http://dx.doi.org/10.1234/jia.v3i1.629.

[7] Y. Rochmawati and R. Kusumaningrum, "Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks," *J. Buana Inform.*, vol. 7, no. 2, pp. 125–134, 2016, doi: 10.24002/jbi.v7i2.491.

[8] R. Muhamad, F. Abdul, and A. Yudhana, "Sistem Kelas Kata Berimbuhan Menggunakan Algoritma Porter Stemmer Sebagai Pembelajaran Bahasa Indonesia," *Telemat. J. Inform. dan Teknol. Inf.*, vol. 16, no. 1, pp. 11–17, 2019.

[9] A. Alshamrani and A. Bahattab, "A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model," *IJCSI Int. J. Comput. Sci. Issues*, vol. 12, no. 1, pp. 106–111, 2015.

[10] R. D. Amlani, "Advantages and limitations of different SDLC models," *Int. J. Comput. Appl. Inf. Technol.*, vol. 1, no. 3, pp. 6–11, 2012.

[11] A. Jelita, "Effective Techniques for Indonesian Text Retrieval," *Ph.D Thesis*, pp. 1–286, 2007.

[12] S. Hardyanti, Wagiran, and S. P. T. Utami, "Perbandingan Afiks Pembentuk Verba Bahasa Indonesia dan Bahasa Jawa," *J. Sastra Indones.*, vol. 6, no. 1, pp. 34–40, 2017.

[13] M. Ramlan, *Morfologi Suatu Tinjauan Deskriptif*, 12th ed. Yogyakarta: C.V. Karyono, 2001.

[14] D. Rosenber and M. Stephens, *Use Case Driven Object Modelling with UML*. Berkeley: Appress, 2007.

[15] S. Shofia and D. A. Anggoro, "Sistem Informasi Manajemen Administrasi Dan Keuangan Pada Tk-It Permata Hati Sumberrejo-Bojonegoro," *JITK (Jurnal Ilmu Pengetah. dan Teknol. Komputer)*, vol. 5, no. 2, pp. 221–230, 2020, doi: 10.33480/jitk.v5i2.1192.

[16] T. Weyrath, F. Schöttl, S. Berthold, and H. Schreyer, "A new UML tool-based Methodology for the Software Requirements Analysis," in *Embedded Real Time Software and Systems (ERTS2012)*, 2012, pp. 1–10.

[17] B. Bankov, "Software Evaluation of PHP MVC Web Applications," *Int. Multidiscip. Sci. GeoConference SGEM*, vol. 19, no. 2.1, pp. 603–610, 2019, doi: 10.5593/sgem2019/2.1/S07.079.

[18] A. Aliwy, "Tokenization as Preprocessing for Arabic Tagging System," *Int. J. Inf. Educ. Technol.*, vol. 2, no. 4, pp. 348–353, 2012, doi: 10.7763/ijiet.2012.v2.149.

[19] J.-P. Chanod and P. Tapanainen, "A Non-deterministic Tokeniser for Finite-State Parsing," in *12th European Conference on Artificial Intelligence*, 1996, pp. 1–3.

[20] A. Prasetyo, W. M. Baihaqi, and I. S. Had, "Algoritma Jaro-Winkler Distance: Fitur Autocorrect dan Spelling Suggestion pada Penulisan Naskah Bahasa Indonesia di BMS TV," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 4, p. 435, 2018, doi: 10.25126/jtiik.201854780.

[21] P. Novantara and O. Pasruli, "Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Skripsi," *J. Buffer Inform.*, vol. 3, 2018, doi: 10.25134/buffer.v5i2.

[22] M. Agarwal, "Text Steganographic Approaches: A Comparison," *Int. J. Netw. Secur. Its Appl.*, vol. 5, no. 1, pp. 91–106, 2013, doi: 10.5121/ijnsa.2013.5107.

[23] K. Yunho, C. Yunja, and K. Monzoo, "Precise Concolic Unit Testing of C Programs Using

Extended Units and Symbolic Alarm Filtering," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 315–326, doi: 10.1145/3180155.3180253.

[24] D. Enda and F. P. Putra, "Identifikasi Kesalahan Tata Bahasa Pada Pernyataan Kebutuhan Menggunakan Probabilistik Model Bahsa N-Gram," *CESS (Journal Comput. Eng. Syst. Sci.*, vol. 4, no. 2, pp. 130–137, 2019, doi: 10.24114/cess.v4i2.12934.