COMPARISON LEVENSHTEIN, SMITH-WATERMAN AND NEEDLEMAN-WUNSCH ON TYPO CHECKING

Ali Nurcahya Astawijaya [1], Ken Kinanti Purnamasari[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jalan Dipatiukur 114-116 Bandung 40132
E-mail : alinurcahya04@gmail.com[1], ken.kinanti@email.unikom.ac.id [2]

ABSTRACT

Typing errors can lead to misunderstanding of meaning because it can change the default word to non-standard word. Research that discusses checking for typing errors has been done a lot. In this research, checking the error of typing text in Indonesian document based on phonetic string matching using levenshtein, smith waterman & needleman wunsch method.

The typing error-checking system can be applied to the user inputting the document on the system, then the system will provide the selected recommendations and words of the word considered incorrect. The stage itself is divided into four stages, namely preprocessing, word error detection, word conversion into levenshtein, smith waterman & needleman wunsch code, recommendation process and word selection. Preprocessing is the stage of the process to get the final result of the list of each word. Word will be used in the process of detecting word errors to determine which words are considered wrong. Next will be the process of changing the word into the code according to the rules of levenshtein, smith waterman & needleman wunsch method. The code to determine the final result is recommendation and correction of the word by finding the smallest edit distance value using approximate string matching technique with levenshtein method of recommendation resulting from the word considered wrong.

Of the three types of testing performed, first is the accuracy test of word error detection. Second is the word recommendation accuracy. And the third is the accuracy of word correction.

Keywords : *Typo checking, Spelling checker, Levenshtein, Smith waterman, Needleman wunsch*

1. INTRODUCTION

In the present era of writing many documents using computers, and on every computer there is not a program that can detect writing errors [1]. Errors in writing can be caused by the fact that the word can not be converted to a word that has a real-world [2]. If typing errors in writing typing, then can change the meaning and change the reader's understanding.

There have been studies that have performed typing errors including using morphologically analyzer, cosine similarity, approximate string matching and phonetic string matching. Phonetic string matching itself has many methods such as soundex, *Levenshtein*, caverphone [3] and one of them is a *Smith-Waterman* which is the development of *Needleman-Wunsch* [4] [5] which matches strings based on the resemblance of speech. A string is said to be similar if it has the same phonetic code. The *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method can overcome the different pronunciation and word writing in English by transforming a word into code based on the sound so that it can be used in spell checking.

In the research of checking typing mistake in languages by Naushad UzZaman and Mumit Khan by using method of *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* from the research resulted accurate with 86.64% with test of 1607 wrong word example, 1473 word successfully detected and get correct repair [4] . While in the case of scientific names using *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* methods 84% [5]. While research on the Indonesian *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method applied to the search case to match the string of 51.78% [6]. So it can be concluded that *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* has a fairly high accuracy of 91.37% in Bangla, 84% in Latin and 51.78% on matching strings of names of Indonesian spelled people.

Therefore, in this study using phonetic string matching with *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method because the previous research gets a fairly high accuracy. Based on the description above, then in this research will be done by using a *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method.

2. CONTENT

2.1 Typo Checking

Typo checking is a process checking of words to detect missplled words and giving candidates the right words. Typo checking has a feature called spelling checker. Design the process of spelling checker there are :

1. Perform pre processing process from the text.

2. Then check every words, have misspelled words or not

3. The next step is to improve the word to get the right word

[2][9].

Typo Checking has 2 ways to detect an errors:

1. Detection of non-word errors is the detection of misspelled words that have no meaning. Example: nasi(rice) becomes nsi. The system only detects word errors that have no meaning.
2. Detection of real-word error is the detection of word errors that have other meanings / meanings such as nasi(rice) become basi(stale).

In this research only focus on detection of misspeled words that have no meaning such as **nasi(rice)** become **nsi**

2.2 String Matching

string matching) can be divided into two there are *exact string matching* and *in exact string matching* [3][10].

2.1. Exact String Matching

matching the string with the arrangement of characters in the matched string has the number or sequence of characters in the same string. Example: the word "clothes" will show matches only with the word "clothes".

2.2. In Exact String Matching

Is also called *fuzzy string matching*, meaning matching strings in which matching strings have a similarity in which both have different character sets (possibly number or sequence) but they have a similarity to the approximate string matching or similarity of speech (Phonetic string matching). InExact string matching can still be subdivided into two:

1. *string matching* based on similarity of writing *(approximate string matching)* is *string matching* with similaity of writing (amount of character, Arrangement of characters in the document). The degree of similarity is determined by whether or not the different writing of the two strings is compared and the value of this level of similarity is determined by the programmer. Example: run with lara, has the same number of characters but there are two different characters. If the difference between these two characters can be tolerated as a writing error then the two strings are said to match.
2. *phonetic correction* is A spelling error that appears due to a user error in entering a query that has a sound like the target term [3][10]. Phonetic string matching is a string matching technique that compares a string with another string based on each phonetic code. Strings that have the same phonetic code can be said to be based on speech.

2.3 Phonetics

It is the science that investigates the sound of language without seeing the function of sound as a differentiator of meaning in a language (7). The phonetic adjective is phonetic (phonetic). The English phonetic section closely related to phonetic string matching is a consonant classification, since consonant classification plays an important role in phonetic string matching. Consonants are based on speech tools that produce them can be divided into seven groups: [3] [8]

1. Labial or lips, which can be distinguished again into two groups, namely:

1.1. Bilabial, the sound is articulated by two lips, the example of sound p, b, m, w.

1.2. Labio-dental, the sound articulated by the lower lip and upper teeth, f, v.

2. Dental, the sound is articulated by the tip of the tongue with the upper teeth, eg th sound (in thin words)

3. Alveolar, the sound is articulated by the tip of the tongue with the teeth-ridge, for example the sounds d, t, n, l, r, s, z.

4. Palato-alveolar, the sound that has alveolar articulation followed by the rise of the tongue to the palate simultaneously, for example the sound of c, j, sh (in show word).

5. Palatal, the sound is articulated by the front of the tongue with hard palate, eg the sound of y.

6. Velar, the sound is articulated by the back of the tongue with soft palate, eg sound k, g.

7. Glottal, the sound is articulated by the glottis, the example of sound h.

Based on the articulation (inhibited) way, the consonants are divided into eight groups: [3] [8]

1. The consonant of the explosive (plosive), is a consonant that occurs with the full resistance of the air currents then the resistance is released suddenly. Example: b, d, g, k, p, t.
2. A nasal or nasal cone, is a consonant formed by blocking the airway from the lungs through the oral cavity, along with the soft palate and the lower cavity being lowered so that air passes through the nasal cavity. Example: m, n.
3. Consonant alloys (affricate), a special type of inhibitory resonant where the process occurs by inhibiting the full flow of air from the lungs, then the barrier is released slowly. Place of articulation on palato-alveolar. Example: c, j.
4. Side consonant (lateral), is a consonant formed by closing the air currents in the middle of the oral cavity so that air out through either side or a side only. Place of articulation on alveolar. Example: l.

5. Fricative consonant, is a consonant formed by narrowing the flow of air exhaled from the lungs, so that the way the air is obstructed and out by shifting. Example: f, v, r, s, z, th (in thin words), sh (in show word), h.
6. Vibrating consonant (rolled), is a consonant formed by inhibiting the flow of air exhaled from the lungs repetitively and quickly and there are many touch (tap) that occurs between the tip of the tongue with the ceiling or rear gum. Examples of rolled r (very rare).
7. Consonant of flapped, is a consonant with a process similar to that of a rolled console but there is only one touch (tap) between the tip of the tongue with a sky or rear gum. Examples of flapped r (very rare). 8. Semi vowels (semi-vowels), sounds that practically include consonants but because at the time of articulation have not formed pure consonants, they are called semi-vowels. Example: w, y.

2.4 Problem Analysis

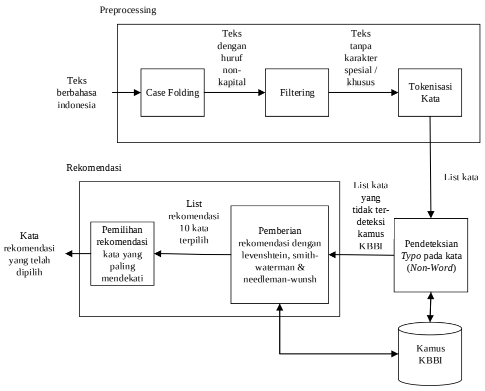
Based on the problem identification in this research there is no problem of accuracy in case of typo checking in Indonesian text using *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method to produce word improvement recommendation. From these problems, it is necessary to apply the algorithm chosen to obtain accuracy results, so it can find out whether or not the algorithm is in tpyo checking Indonesian case. So as a solution to that, the *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method will be applied to the system. *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* algorithm to assist in to recommend the wrong word and levenshtein method to assist in selection of word improvement recommendations. So that will get the result of how much accuracy and performance of this algorithm in handling problem which have been explained.

In another study relating to the *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method previously performed by "scientific name research on IPB thesis collection using *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* " gained a fairly high percentage of success of 84% in case of scientific name typing errors [5]. Therefore, researchers used the method of *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* because in previous research to get high enough accuracy in scientific language. So in this typo checking research, using *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method to recommend the wrong word type in Indonesian text.

2.5 System Analysis

The typo checking system will be built with the system picture shown in Figure 1. The typo checking system stage starts from receiving the text input of the document to preprocessing consisting of case folding, filtering, tokenisasi sentence, tokenisasi kata. The result of preprocessing in the form of a word will then be processed to the next process by looking for typos, typographical errors in the form of words that are not detected with a dictionary contained in KBBI. Dictionary words will be phonetic code in accordance with the rules of phonetic code of *Levenshtein, Smith-Waterman, dan Needleman-Wunsch*, words that have the same phonetic code will be a recommendation of word improvement and will dilakukakan selection of words that are considered wrong of the recommendations to be given.

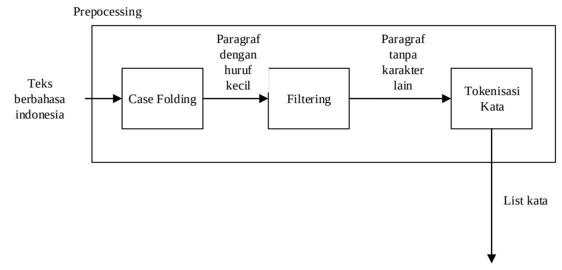
Based on the analysis of typo checking system to be built, then the picture of the system can be seen in Picture 1


Preprocessing
Teks berbahasa indonesia
Case Folding
Teks dengan huruf non-kapital
Filtering
Teks tanpa karakter spesial / khusus
Tokenisasi Kata
List kata
Rekomendasi
List kata yang tidak ter-deteksi kamus KBBI
Pendeteksian *Typo* pada kata (*Non-Word*)
Pemberian rekomendasi dengan levenshtein, smith-waterman & needleman-wunsh
List rekomendasi 10 kata terpilih
Pemilihan rekomendasi kata yang paling mendekati
Kata rekomendasi yang telah dipilih
Kamus KBBI

Picture 1 Main System Process Scheme

2.5.1 Preprocessing Analysis

Preprocessing stage is the step to prepare input data to be processed at next stage with final result to get list every word. Preprocessing in this research consists of several steps, namely: case folding, filtering, tokenisasi sentence and tokenisasi words.



Picture 2 Preprocessing Scheme

2.5.2 Determination of Typo

Determination of typos is by comparing each word with a list of words contained in the KBBI dictionary, words that are not detected KBBI dictionary is considered the wrong word.

2.5.3 *Levenshtein* Algorithm

This algorithm is known as Levenshtein Distance. This algorithm was discovered by Vladimir Losifovich Levenshtein who was a scientist from Russia in 1965. This algorithm measures the similarity between 2 strings to determine the smallest value or called the edit distance value, the two strings

to be compared would be better known as the source string (s) with Target string (t) [11], for example:
Jika (s) = “datang”, dan (t) = “datank” ==> then *Levenshtein*(s,t) = 0, because that two of string hav same value

1. if (s) =”datang”, dan (t) = “datank” ==> then *Levenstein*(s,t) =1 , because there are difference between two string is in the third letter 'b' with 'n'. For example if there are 2 words x = DATANG with y = DATANK, what is the value of edit distance difference letters of both words

1. Creation of matrix tables

Tabel 2 Creation of matrix tables

		d	a	t	a	n	k
	0	1	2	3	4	5	6
d	1						
a	2						
t	3						
a	4						
n	5						
g	6						

a. Filling matrix table
when i = 1, dan j = 1
*x[i] = B , y[i] = B, cost = 0
b. After that put the minimal value minimal from

- d[i-1,j] + 1 ===> d[1-1,1] + 1 = 1
- d[i,j-1] + 1 ===> d[1,1-1] + 1 = 2
- d[i-1,j-1] + cost ===> d[1-1,1-1] + 0 = 0

2. filling the matrix table

Tabel 3 Filling Matrix Table

		d	a	t	a	n	k
	0	1	2	3	4	5	6
d	1	0	1	2	3	4	5
a	2	1	0	1	2	3	4
t	3	2	1	0	1	2	3
a	4	3	2	1	0	1	2
n	5	4	3	2	1	0	1
g	6	5	4	3	2	1	1

3. result filling matrix table .

Tabel 4 *Edit Distance result*

		d	a	t	a	n	k
	0	1	2	3	4	5	6
d	1	**0**	1	2	3	4	5
a	2	1	**0**	1	2	3	4
t	3	2	1	**0**	1	2	3
a	4	3	2	1	**0**	1	2
n	5	4	3	2	1	**0**	1
g	6	5	4	3	2	1	**1**

From here we can know the value of edit distance of both DATANG and DATANK word is worth 1 (one). From the word that is considered wrong will be the source string that will be compared by any recommendations generated system. The word recommendation that has the smallest edit distance value will be selected as an incorrect word correction

2.6 System Testing

System testing phase aims to find errors or deficiencies in the system being tested. Testing intends to know whether the prototype made is in accordance with the purpose of research.

2.6.1 *Black Box Testing*

Tabel 5 *Black Box Testing*

No	Process Name	Testing Point	Type of Testing
1	Choose input	select data input	*Black box*
2	*Case Folding*	choosing menu “*case folding”* process changing character become small characters	*Black box*
3	*Filtering*	Choosing menu “*filtering*” process deleting character not valid	*Black box*
4	Tokenisasi Kalimat	Select menu “tokenisasi kalimat” process showing list of sentence	*Black box*
5	Tokenisasi Kata	Select menu “tokenisasi kata” process showing list of words	*Black box*
6	*Typo Checking*	Select “*typo checking*” process determining of typo	*Black box*

2.7 Accuracy Testing

2.7.1 Accuracy Testing Levenshtein

This test aims to determine the accuracy made in this study. In this research using detection test of word error, testing of improvement accuracy and recommendation.

The purpose of testing the detection of word errors is to see how accurate the detection of word errors. Measurement accuracy of word error detection commonly used in the research of typing error checking [12]:$Accuracy\ Of\ Typo\ Checking$

$$= \left(\frac{Total\ Of\ Expected\ Typo}{Total\ Of\ Typo}\right) x\ 100\%$$

where,
Total Of Expected Typo : There are total amount of expected typo.
Total Of Typo : There are total amount of typo.

Tabel 6 Accuracy Of Typo Checking

No Document	Total Of Expected Typo	Total of Typo	Accuracy
1	6	61	9.8360
2	5	45	11.1111
3	5	81	6.1728
4	8	59	13.5593
5	5	62	8.0645
6	4	57	7.0175
7	5	23	21.7391
8	4	44	9.0909
9	8	56	14.2857
10	8	49	16.3265
11	5	32	15.625
12	5	42	11.9047
13	9	61	9.8360
14	5	42	11.9047
15	7	97	7.2164
16	9	54	9.2592
17	5	110	4.5454
18	5	47	10.6382
19	5	85	5.8823
20	6	57	10.5263
21	5	123	4.0650
22	4	42	9.5238
23	5	110	4.5454
24	5	97	5.1546
25	4	107	3.7383
26	4	45	11.1111
27	5	38	13.1578
28	5	44	11.3636
29	5	89	5.6179
30	5	71	7.0422
Average	5.5333	64.3333	9.6620

From Table 6 the analysis of word error detection using KBBI 4th edition dictionary resulted in an accuracy of 9.6620%. The following is an explanation of the cause of the accuracy of typo checking recommendations generated by a small system.

1. List of incomplete words on the KBBI dictionary so that many words are not detected.
2. The name and place is considered a word error because it is not contained in the KBBI dictionary

.2.7.2 Accuracy Smith-Waterman

The purpose of this test is to see how many words the recommendations are correct. Based on the scenario made The test of accuracy of recommendation in this research use accuracy test. Measurement of repair accuracy commonly used in the research of typing error checking [12]:

$$Accuracy\ Of\ Recommendation = \left(\frac{Total\ Of\ Expected\ Recommendation}{Total\ Of\ Typo}\right) x\ 100\ \%$$

where,

Total Of Expected Recommendation : there are total amount of expected recommendation.

Total Of Typo : there are total amount of typo.

Tabel 7 Accuracy Of Recommendation

No Documen	Total Of Expected Recommendation	Total of Typo	Accuracy
1	13	61	21.314
2	6	45	13.3333
3	6	81	7.4074
4	6	59	10.1694
5	13	62	20.9677
6	4	57	7.0175
7	7	23	30.4348
8	7	44	15.9090
9	6	56	10.7142
10	13	49	20.4082
11	5	32	15.625
12	6	42	14.2857
13	10	61	16.3934
14	11	42	26.1905
15	18	97	18.5567
16	10	54	18.5185
17	4	110	3.6363
18	2	47	4.2553
19	6	85	7.0588
20	13	57	22.8070
21	13	123	10.5691
22	9	42	21.4286
23	6	110	5.4545
24	9	97	9.2784
25	8	107	7.4766
26	7	45	15.5556
27	6	38	15.7895
28	5	44	11.3636
29	17	89	19.1011
30	6	71	8.4507
Average	8.4	64.3333	14.3156

From Table 7, the analysis of word recommendation test using Smith-Waterman yielded an accuracy of 14.3156%. The following is an explanation of the cause of the accuracy of typo checking recommendations generated by a small system ie.

1. List of incomplete words on the KBBI dictionary so that many words do not have the correct recommendations.
2. The word error affects the phonetic code conversion causing the resulting phonetic code not to match the correct phonetic code so that the recommendation should not arise.

2.7.3. Accuracy Testing Needleman-Wunsch

The purpose of this test is to see how many words have the correct improvement in the measurement of the correctness accuracy commonly used in the research of typing error checking [12]:

$$Accuracy\ Of\ Fixed\ Words = \left(\frac{Total\ Of\ Excpected\ Fixed\ Words}{Total\ Of\ Typo}\right) x\ 100\%$$

where,
Total Of Expected Fixed Words : there are total amount of expected fixed words
Total Of Typo : there are total amount of typo.

Tabel 8 Accuracy Of Fixed Words

No Document	Total Of Expected Fixed Words	Total Of Typo	Accuracy
1	13	61	21.314
2	5	45	11.1111
3	5	81	6.1728
4	6	59	10.1694
5	10	62	16.129
6	2	57	3.5088
7	6	23	26.087
8	5	44	11.3636
9	4	56	7.1428
10	13	49	18.367
11	4	32	12.5
12	5	42	11.905
13	3	61	4.918
14	7	42	16.667
15	8	97	8.2474
16	4	54	7.4074
17	1	110	0.9090
18	1	47	2.1277
19	3	85	3.5294
20	9	57	15.7894
21	9	123	7.3171
22	7	42	16.6667
23	3	110	2.7273
24	6	97	6.1856
25	1	107	0.9346
26	4	45	8.8889
27	5	38	13.1579
28	4	44	9.0909
29	5	89	5.618
30	4	71	5.6338
Rata-rata	5.4	64.3333	9.7195

From Table 8 the analysis of the results of the word-compression testing using the levenshtein method yielded an average accuracy of 9.7195%. The following is an explanation of the cause of the accuracy of word typo checking improvements generated by a small system**:**

1. There is no correct recommendation
2. KBBI is incomplete so it has little recommendation
3. The correct word has a large edit distance so it is not selected
4. The result of variation of phonetic code error is different and does not produce the recommended recommendation

2.8 Analysis Of Testing Result

The system test that has been done shows that the system which is made for error detection, recommendation and correction of preprocessing process, word error detection, word conversion into *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* phonetic code and recommendation and correction of word by *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method, fulfill requirement Functional.

Tabel 9 Analysis Of Testing Result

No Document	*Levenshtein*	*Smith-Waterman*	*Needleman-Wunsch*
1	9.8360%	21.314%	21.314%
2	11.1111%	13.3333%	11.1111%
3	6.1728%	7.4074%	6.1728%
4	13.5593%	10.1694%	10.1694%
5	8.0645%	20.9677%	16.129%
6	7.0175%	7.0175%	3.5088%
7	21.7391%	30.4348%	26.087%
8	9.0909%	15.9090%	11.3636%
9	14.2857%	10.7142%	7.1428%
10	16.3265%	20.4082%	18.367%
11	15.625%	15.625%	12.5%
12	11.9047%	14.2857%	11.905%
13	9.8360%	16.3934%	4.918%
14	11.9047%	26.1905%	16.667%
15	7.2164%	18.5567%	8.2474%
16	9.2592%	18.5185%	7.4074%
17	4.5454%	3.6363%	0.9090%
18	10.6382%	4.2553%	2.1277%
19	5.8823%	7.0588%	3.5294%
20	10.5263%	22.8070%	15.7894%
21	4.0650%	10.5691%	7.3171%
22	9.5238%	21.4286%	16.6667%
23	4.5454%	5.4545%	2.7273%
24	5.1546%	9.2784%	6.1856%
25	3.7383%	7.4766%	0.9346%
26	11.1111%	15.5556%	8.8889%
27	13.1578%	15.7895%	13.1579%
28	11.3636%	11.3636%	9.0909%
29	5.6179%	19.1011%	5.618%
30	7.0422%	8.4507%	5.6338%
Rata-rata	9.6620%	14.3156%	9.7195%

2. CLOSING

3.1 Conclusion

Based on the results of research that has been done, it can be seen that the typo checking using *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method has the detection of a bad word error from the results of 30 news document test with the average accuracy of word detection of 9.6620% in each document while manual errors made The average of each document as much as 5 word errors, detected word errors are not necessarily pure typos errors because the built system can not detect the name and place because the word error is determined by a word that is not contained in the data dictionary so the data dictionary is very dependent in determining the word error and word recommendations. But from an average of 5 words per document that was deliberately made wrong. The system can properly detect, provide recommendations and provide correct fixes because the word is contained in the data dictionary. However, the location of the letters themselves can affect the conversion into the phonetic code so that sometimes the pure typewriter does not get appropriate recommendations and corrections such as consonant type errors at the beginning of a word or in the middle of a word. If typing errors at the end of the word mostly do not affect significantly if the word is long or have long syllables because the phonetic code maximum consists of only 4 letters so that if it has get the next four letters will be ignored

The average accuracy of word detection is 9.6620%, the accuracy of word recommendation is 14.3156% and the accuracy of small word improvement with an average of 9.7195%.

The result of word error detection depends on word completeness on the data dictionary used, and word recommendations depend on the same phonetic code generated from the data dictionary, via *Levenshtein, Smith-Waterman, dan Needleman-Wunsch* method and word selection depending on the smallest edit distance value of the resulting recommendation.

3.2 Facilities

In this research using KBBI dictionary as a resource in determining the wrong word, has accurate detection of error words that are not good because it can not detect the name and place. So further research is suggested to replace dictionary resources with more and more complete dictionaries in word inventory, or additions to the detection of names and places such as NER (named entity recognition) so as to detect names and places to improve word detection, recommendation accuracy and improved word correction accuracy.

BIBILIOGRAPHY

[1] T. A. Hariawan, Implementasi Spell Checker Dengan Menggunakan Algoritma Levenshtein Pada Kata Berbahasa Indonesia, Daerah Istimewa Yogyakarta: Sinta, 2005.

[2] M. Fachrurrozi and A. A. Manik, Perbaikan Ejaan Kata pada Dokumen Bahasa Indonesia dengan Metode Cosine Similarity, Palembang: Teknik Informatika Universitas Sriwijaya, 2015.

[3] M. Syaroni and R. Munir, Pencocokan String Berdasarkan Kemiripan Ucapan (Phonetic String Matching) dalam Bahasa Inggris, Bandung: Departemen Teknik Informatika ITB, 2004.

[4] N. UzZaman and M. Khan, A Encoding for Bangla and its Application in Spelling Checker, Bangladesh: Center for Research on Bangla Language Processing BRAC University, 2005.

[5] H. Cahyadi, Pencarian Nama Ilmiah pada Koleksi Tesis Perpustakaan IPB, Bogor: Departemen Ilmu Komputer IPB, 2010.

[6] S. Rahadian, R. W. Ciptasari and Adiwijaya, Perbandingan Pencocokan String Nama dalam Bahasa Indonesia Berdasarkan Kemiripan Ucapan (Phonetic String Matching) Needleman dan Smith-Waterman, Bandung: Fakultas Teknik Informatika Universitas Telkom, 2007.

[7] Anonim, "Fonetik," 11 Mei 2017. [Online]. Available: https://kbbi.web.id/fonetik. [Accessed 15 Mei 2017].

[8] D. Jones, The Pronouncation of English 4th edition, Cambridge: Great Britain at the University Press, 1972.

[9] M. Y. Soleh and A. Purwanti, A Non Word Error Spell Checker for Indonesian using Morphological Analyzer and HMM, Bandung: Department of Informatics ITB, 2011.

[10] Binstock and Rex, Practical Algorithms for Programmers, Addison Wesley, 1995.

[11] A. Pahdi, Koreksi Ejaan Istilah Komputer Berbasis Kombinasi Algoritma Damerau-Levenshtein dan Algortima Soundex, Banjarbaru: Journal Speed, 2016.

[12] M. F. Wahyudipraja, Implementasi Algoritma Cocke -Younger-Kasami (CYK) Dan Levenshtein Untuk Merekomendasikan Perbaikan Struktur Kalimat Dan Kesalahan Pengetikkan Bahasa Indonesia, Bandung: JBPTUNIKOMPP, 2015.