SPELLING ERROR CORRECTION IN INDONESIAN USING DAMERAU-LEVENSHTEIN DISTANCE DAN N-GRAM

Diah Anggreni Ratna Sari K[1], Budi Irmawati[1]*, Ramaditia Dwiyansaputra[1]
[1]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Mataram , Indonesia
Email: reny283112@gmail.com, [budi-i, rama]@unram.ac.id

Abstract

Writing errors or spelling is a thing that needs to be considered because it can affect the calculations performed by some of the topics on Natural Language Processing that relies on the validity of the input data. Several studies have been conducted to correct writing errors that occur, one of which study by Fahma, A. I., et al using n-gram method and Levenshtein distance produced corrections with the best precision value of 0.97 for insertion type and best recall value by 1 for substitution types. With high accuracy, this study proposes to use the algorithm of development of Levenshtein, namely Damerau-Levenshtein, and n-gram methods. Damerau-Levenshtein has the same operations like insertion, deletion, substitution but with the addition of transposition operations between two characters. Damerau not only distinguishes four edit operations but also states that operations in the developed algorithms, can fit about 80% of all human writing errors. The types of n-grams used are bigram (n = 2) and trigram (n = 3). The testing results obtained in this study for the detection accuracy of the precision and recall ranged from 80%-100%. While correction accuracy testing uses equations proposed by Dahlmier and Ng, among the average accuracy values of precision and recall for all three scenarios, scenario C with a top 10 rating has the highest accuracy value of 96%.

Keywords: *Correction; Damerau-Levenshtein distance; Detection; n-gram; Spelling corrections*

**Corresponding Author*

1. INTRODUCTION

Spelling errors are crucial in writings as well as in language processing. The language processing such as POS tagging, name entity recognition, information retrieval, and machine translation needs correct training data. Spelling correction may be caused by writers' mistakes or inappropriate corrections given by an automatic spelling corrections application.

Rochmawati and Kusumaningrum compared four Approximate String-Matching algorithms: Hamming Distance, Levenshtein Distance, Damerau-Levenshtein Distance and Jaro Winkler Distance. From the four algorithms, Jaro Winkler Distance got highest MAP (the mean average precision) score of 0.87. This algorithm is more frequently to be used to detect duplication. The MAP scores for other algorithms are 085, 0.74, and 0.46 for Damerau-Levenshtein Distance, Levenshtein Distance, and Hamming Distance respectivelly [1]. *Levenshtein Distance* calculates a distance between an original word and possible corrected words in operations of insertion, deletion, and substitution. The correction is selected from words with a minimum distance. In addition, *Damerau-Levenshtein* distance is the expansion of the *Levenshtein* distance with an additional transposition between two characters [2]. Thang and Huy found that 80% of the candidate corrections are correct [3]. Maghfirah et. al combined Damerau-Levenshtein and Dictionary Lookup methods and obtained precision of 0.78 [4]. In addition, *n-gram* is a language model to predict the next word in a corpus based on the previous *n-1* words using probabilistic model [5]. Fahma et al combined Levenshtein Distance and n-gram. Moreover, Atawy and Elghany [6] do automatic correction in English text and obtained 93% accuracy.

Therefore, based on the previous works that used distance and n-gram to measure probable word corrections, our work combined the Damerau-Levenshtein distance and n-gram model to correct word spelling errors.

2. LITERATURE REVIEWS

Spelling correction can be corrected using two methods: absolute and relative corrections. The absolute

correction is performed by writing all possible misspelled word and its correction pair. The relative correction choses a word that is most similar to the misspelled word [6].

Mawardi et al. corrected spelling errors in Indonesian documents using Finite State Automata (FSA) and *Levenshtein* distance [7]. They represented their results in perplexity, hit rate correction, and false positive rate score. The experimental results showed that the lowest perplexity was unigram with 1.14 score. The highest correction hit rate was combination of the bigram and trigram with 71.20% score. However, bigram was better in the processing time of 21.23 minutes. Beside those results, all unigram, bigram, and trigram have false positive of 4.15%. Then they modified the FSA to improves the results and obtained better hit rate correction for bigram as 85.44%.

Maghfira et al. used Dictionary Lookup which is effective to decide whether a word is spelled correctly or not based on lexical resource. They also reported that *Damerau-Levenshtein* distance corrected misspelled word better than *Levenshtein* distance. Experimental results provided 78% precision and 100% recall [4].

Fahma et al. identified typographical error in documents written in Indonesian using *Levenshtein* Distance. It used to detect the number of candidates according to identified typographical errors. As the candidates of the *Levenshtein* distance were not sorted, bi-gram was used to sort the candidates based on cosine similarity measure. In this case, the words were split to two characters and their TF-IDF were used to calculate the cosine similarity. They got the best insertion of 97% and the best recall of 100% for substitution [2].

Simanjuntak et al. combined [8] Peter Norvig methode and n-gram that was evaluated in 9 types errors involved 160 sentences where each sentence had an error. Their experiment show that the combined method obtained 65.93% precision and 78.07% recall. It worked well to correct one error in a word but fail to correct more than one characters in a word.

In this work, we used *Damerau-Levenshtein* distance as an extension of *Levenshtein* distance. *Damerau-Levenshtein* distance extends the minimum operation as in *Levenshtein* Distance, such as insertion, deletion, substitution with additional operation of transposition between two characters [8].

3. RESEARCH METHOD

a. Datasets

The training data were taken from Indonesian dictionary and TED-Monolingual-Parallel-Corpus (TED corpus) parts extracted from *www.ted.com*. It has 236,543 Indonesian sentences. The text example can be seen in **Figure 1**.

> Pada Memorial Day (hari pahlawan), kami meluncurkan "The National Mall," album musik yang sadar-lokasi yang diluncurkan secara eksklusif sebagai aplikasi smartphone yang menggunakan fungsi GPS yang sudah ada untuk memetakan seluruh taman dengan suara di kampung halaman kami, Washington, D.C. Ratusan bagian musik ditandai secara geografis di seluruh taman sehingga saat si pendengar menjelajahi lingkungan itu, sebuah lagu terdengar di sekitar mereka.

Figure 1. Example of the TED corpus

To evaluate our model, we collected test data from five undergraduate reports and five journal papers. The test data were duplicated. One copy was kept originally as the gold standard data while some words in the other copy was spelled errors manually. After applied our model, the miss spelled data were compared to words in an Indonesian dictionary and to the original source data for evaluation. All data were pre-processed by case-folding, filtering, and tokenizing.

b. Calculate n-gram model

The *n-gram* model was used to sort the candidate correction given by the model. It contains a list of *n-gram* and their frequency [9]. It was trained from the TED corpus. The *n-gram* model is calculated using *Maximum Likelihood Estimation* (MLE) [5] with smoothing as in Equations (1) and (2).

$$P_1'(c_j^i|W^{i-1}) = \frac{count(W^{i-1}c_j^i) + k}{\sum_{r=1}^{k_i} count(W^{i-1}) + k * V} = \frac{count(W^{i-1}c_j^i)+0.01}{\sum_{r=1}^{k_i} count(W^{i-1})+0.01*V} \quad (1)$$

$$P_2'(c_j^i|W^{i-1}, W^{i+1}) = \frac{count(W^{i-1}c_j^iW^{i+1}) + k}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i) + k * V} = \frac{count(W^{i-1}cW^{i+1})+0.01}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i)+0.01*V} \quad (2)$$

v is the vocabulary number.

c. *Calculate Candidate Corrections (Confusion Set)*

The confusion set is a set contains all possible correction for each word. The confusion set are calculated using *Damerau-Lavenshtein* distance to find candidate words that are similar to the misspelled word. The algorithm is explained below.

1. Set *n* and *m* as the length of character *s* and *t*.
   If *n*=0 or *m*=0, return the *distance* as
   $$distance = max(n, m). \quad (1)$$
   Go to step 7.
2. Create a matrix *d* for *m*+1 rows and *n*+1 columns.
3. Fill the first row with 0..*n* and the first column with 0..*m*
4. Check each in *s* and *t*.
   $$cost = \begin{cases} 0 \ if \ s[i] = t \\ 1 \ if \ s[i] \neq t \end{cases} \quad (2)$$
5. Set *d*[*i*,*j*] = *min*(*x*,*y*,*z*)
   x = [*i*-1, *j*]+1 (3)
   y = [*i*, *j*-1]+1 (4)
   z = [*i*-1, *j*-1]+*cost* (5)
   if *i* > 1 and *j* > 1 and *s*[*i*]=*t*[*j*-1] and *s*[*i*-1]=*t*[*j*], which means that after the two compared words, there is a character that can be repositioned.
   Set *d*[*i*,*j*]=min(d[i,j], d[i-2],j2)+cost (6)
6. After steps 1 to 5, the edit distance is in d[n,m], the right bottom corner.
7. The candidate corrections are the words with the smallest *edit distance*.

d. *Evaluation Scenarios*

The model is evaluated based on *minimum edit distance* (*med*) score. We reported our model using *med* of 1, 2, and both 1 and 2 for 1-best, 5-best, and 10-best.

e. *Model Implementation*

Our model detected misspelled words by comparing a document with an Indonesian dictionary. It means if the misspelled words are available in the dictionary, it fails to detect the misspelled words. After finding the misspelled words, the model built a confusion set for each word. The confusion set consisted of all candidate corrections for each misspelled word.

The next step is to calculate an edit distance score for each candidate in the confusion set (Explain in **Subsection III c**). It returned some possible correction words. Then the correction words were ranked by *n-gram* model (Explained in **Subsection III b**) to choose the best prediction. **Figure 2** shows this model.

f. *Evaluation Metric*

The experimental results were evaluated based on a metric proposed by [10] below.

$$Presition = \frac{\#correct\_predictions}{\#correction\_given\_by\_the\_model}$$

$$Recall = \frac{\#correct\_predictions}{\#misspelled\_errors}$$

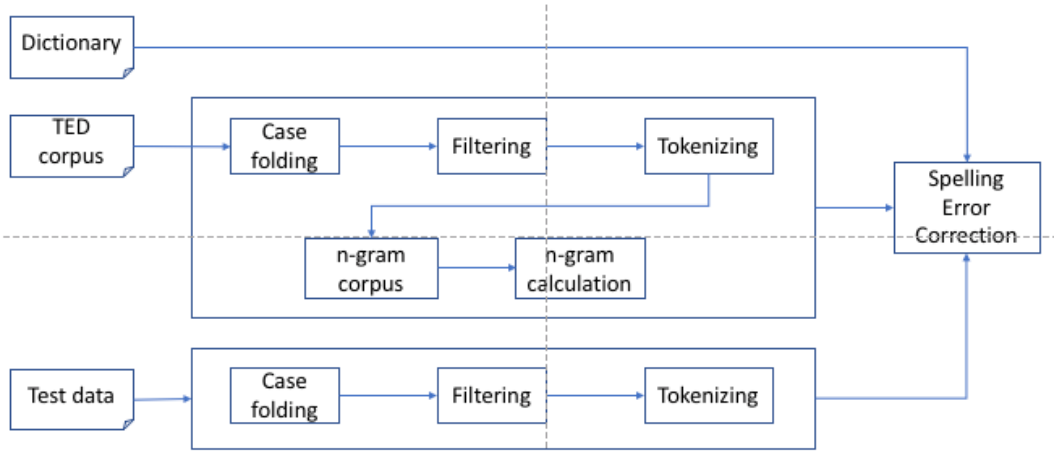

Figure 2. Model Implementation

g. Evaluation Scenarios

The experiment goal is to evaluate whether the proposed system is able to detect a misspelled word and propose a proper correction. We used 10 documents with varies number of misspelled words. In each document, we randomly choose a paragraph as the testing data.

The suggested correction is taken from the 1-best, 5-best, and 10-best based on the Median Edit Distance (MED) score below.

- Scenario A: Use candidate with MED=1.
- Scenario B: Use candidate with MED=2.
- Scenario C: Use candidate with MED=1 and MED=2.

If there are no candidate corrections in MED=1 and MED=2, in each 1-best, 5-best, and 10-best lists, we consider that the misspelled word does not have a proper correction word.

4. RESULTS AND DISCUSSION

All words with the smallest *med* were listed as candidate corrections. After they were ranked by the *n-gram* model, we took 10-best corrections and evaluate whether the correct word were there. The correct word is the prediction word which is similar to the original word.

a. Experimental Results

For evaluation, we reported whether the correct prediction was in the 1-best, 5-best, or 10-best. We displays the three results for med=1 in **Figures** 3 to 5.

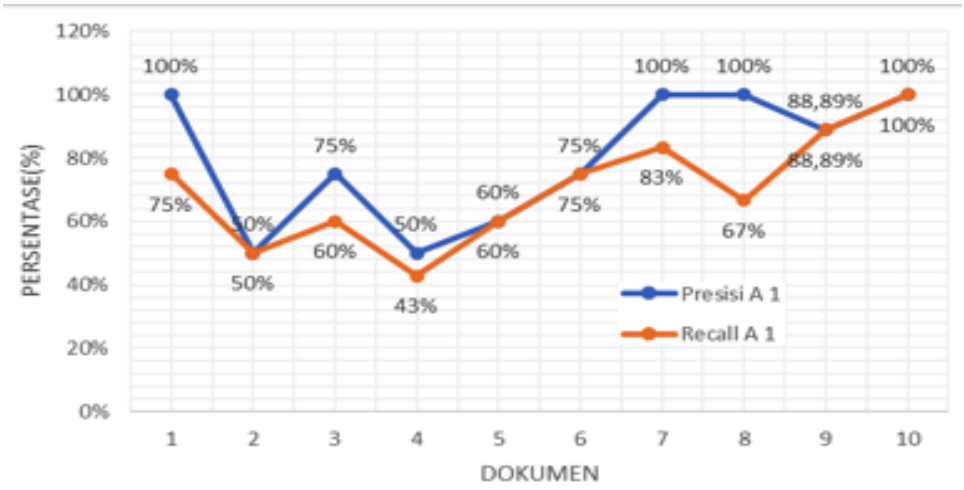

Figure 3. The precision and recall of 1-best with med=1.

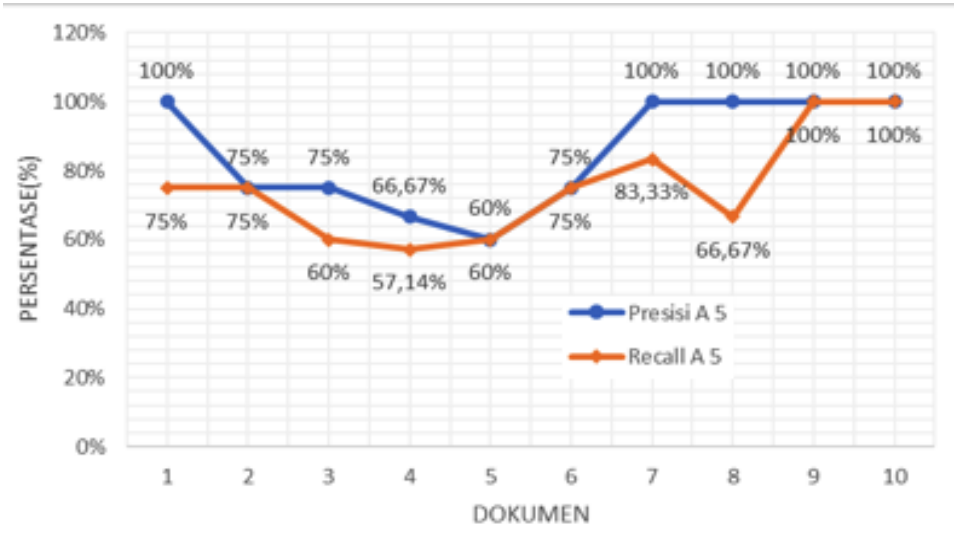

Figure 4. The precision and recall of 5-best with med=1.

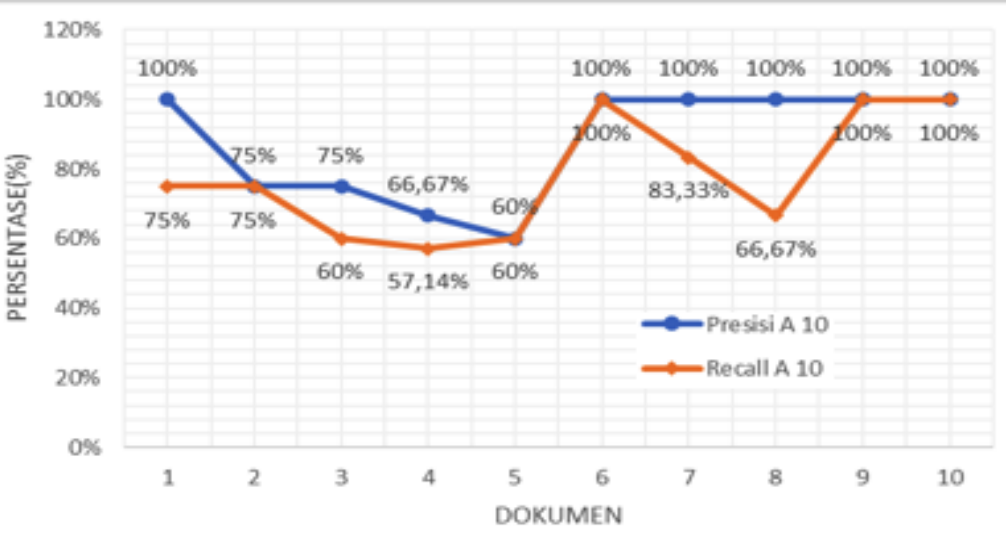

Figure 5. The precision and recall of 10-best with med=1.

TABLE 1. THE CORRECTION PRECISION RECALL WITH *MED*=1

doc	1-best		5-best		10-best	
ID	Prec	Rec	Prec	Rec	Prec	Rec
1	1.00	0.75	1.00	0.75	1.00	0.75
2	0.50	0.50	0.75	0.75	0.75	0.75
3	0.75	0.60	0.75	0.60	0.75	0.60
4	**0.50**	**0.43**	0.67	0.57	0.67	0.57
5	0.60	0.60	0.60	0.60	0.60	0.60
6	0.75	0.75	0.75	0.75	1.00	1.00
7	1.00	0.83	1.00	0.83	1.00	0.83
8	1.00	0.67	1.00	0.67	1.00	0.67
9	0.89	0.89	1.00	1.00	1.00	1.00
10	**1.00**	**1.00**	1.00	1.00	1.00	1.00
	0.80	**0.70**	**0.86**	**0.75**	**0.88**	**0.78**

For each document, the results are shown in Tables I to III with different *med* values. **Table I** shows that all the misspelled word in **documents-10** can be corrected well. The misspelled words were `membacsa' and `menuntt' which were corrected well as `membaca' and `menuntut'. The worst prediction was in **document-4** follows by document-5. We will investigate them in **Subsection IV.b** based on the misspelled words and their correction for *1-best*. These results also show that the precision score for the 5-bests are better than their recall, means that the number of proper correction is higher than the proposed candidate correction because it took proposed words with med=1, the most similar words.

TABLE II. THE CORRECTION PRECISION RECALL WITH *MED*=2

doc	1-best		5-best		10-best	
ID	Prec	Rec	Prec	Rec	Prec	Rec
1	0.75	0.75	0.75	0.75	0.75	0.75
2	0.75	0.75	0.75	0.75	0.75	0.75
3	1.00	1.00	1.00	1.00	1.00	1.00
4	0.71	0.71	0.86	0.86	0.86	0.86
5	0.60	0.60	0.80	0.80	1.00	1.00
6	0.50	0.50	0.75	0.75	0.75	0.75
7	1.00	1.00	1.00	1.00	1.00	1.00
8	0.67	0.67	1.00	1.00	1.00	1.00
9	0.78	0.78	0.78	0.78	0.78	0.78
10	**0.00**	**0.00**	**0.50**	**0.50**	**0.50**	**0.5**
	0.67	**0.67**	**0.82**	**0.82**	**0.84**	**0.84**

Table II shows that with *med*=2 the precision and recall scores are the same for all test documents. These results show that more candidate corrections, increase the probability to obtain correct candidates. However, it could not correct document-6 well and even could not correct **document-10** that successfully corrected with *med*=1. It seems that document-10 has unique correction so that candidates with med=2 do not provide proper corrections.

TABLE III. THE CORRECTION PRECISION RECALL WITH *MED*=1 AND *MED*=2

doc ID	1-best		5-best		10-best	
	Prec	Rec	Prec	Rec	Prec	Rec
1	0.75	0.75	0.75	0.75	0.75	0.75
2	0.75	0.75	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00	1.00	1.00
4	0.86	0.86	1.00	1.00	1.00	1.00
5	0.60	0.60	1.00	1.00	1.00	1.00
6	0.75	0.75	0.75	0.75	1.00	1.00
7	1.00	1.00	1.00	1.00	1.00	1.00
8	0.67	0.67	1.00	1.00	1.00	1.00
9	0.78	0.78	1.00	1.00	1.00	1.00
10	1.00	1.00	1.00	1.00	1.00	1.00
	0.82	**0.82**	**0.95**	**0.95**	**0.96**	**0.96**

Table III shows the same score of precision and recall on each test document and all *n*-best as showed by Table II. These scenario provides the best precision and recall score than scenarios A and B. These are because there are more candidate corrections provide by the proposed model. However, the corrections given for document-1 are not optimal (both precision recall scores are 0.75). It may because the document-1 does not have a correction.

TABLE IV. THE AVERAGE CORRECTION PRECISION RECALL FOR THREE *MED* VALUES

med	*1-best*		*5-best*		*10-best*	
	Prec	Rec	Prec	Rec	Prec	Rec
1	**0.80**	**0.70**	0.86	0.75	0.88	0.78
2	0.67	0.67	0.82	0.82	0.84	0.84
1 & 2	0.82	0.82	0.95	0.95	0.96	0.96

In Table IV we show the average results from Tables I to III. It shows that the **med=1** performs better than that of med=2 for the *1-best* suggestion. From the discussion on Table II, the model works well for a unique correction for a misspelled word. In general, candidates from combination of med=1 and med=2 perform the best with both precision and recall of 96%.

TABLE V. THE AVERAGE CORRECTION PRECISION RECALL FOR THE THREE SCENARIOS

med	Prec	Rec
1	0.85	0.74
2	0.77	0.77
1 & 2	0.91	0.91

Table V shows the average score of precision recall for med=1, med=2, and med=1&2. Based on the previous discussion, corrections with med=1 and med-2 provide the best result as it took corrections from the two possibilities.

b. Error Analysis

This Subsection will evaluate the model performance. Table VI presents some misspelled words in documents-4 and document-5. These documents have the lowest precision and recall for *1-best* prediction with *med*=1.

TABLE VI. THE MISSPELLED WORDS IN DOCUMENTS 4 AND 5 AND THEIR *1-BEST* CORRECTION FOR *MED*=1

Doc ID	Misspelled words	Original	Prediction	Error types
4	**maa**	masa	mana	*deletion*
4	jugsa	juga	juga	*insertion*
4	**skans**	akan	kans	*deletion+ insertion*
4	sebsgai	sebagai	sebagai	*replacement*
4	suau	suatu	suatu	*deletion*
4	**eknollogi**	teknologi	-	*replacement+ insertion*
5	**mengis**	menulis	mengisi	*replacement 2 characters*
5	mengkiuti	mengikuti	mengikuti	*transposition*
5	**langa**	yang	sanga	*replacement+ insertion*

In the **document-4**, there are three misspelled words that fail to be corrected. They are `maa', `akan', and `eknollogi'. The incorrect correction for `maa' is reasonable because *bigram* for `masa' and `mana' has a high probability. For the last two words, they have two errors in each word. In the word `skans', `s' replaced `a' at the beginning and the second `s' was inserted at the end. The `t' in `teknologi' was missing and it also has double `l'. Similar to Simanjuntak et al. [8], our work still cannot correct more than two errors in a word but with better precision and recall.

In the **document-5**, the misspelled words that fail to be corrected were `mengis' and `langa' which were corrected as `mengais' and `sangat'. The correct words are `menulis' and `yang'. In the first case, the character `g' replaced the characters `u' and `l' while in the second case, the character `l' replaced the characters `y' and it has additional character `a' at the end. Both cases have two replacements.

From four cases in **document-4** and **document-5**, we conclude that the model works well to correct one spelling character covering replacement, insertion, deletion, and transposition. However, it still fails to correct more than one errors in a word. The possible solution is by correct the character one by one and re-rank the predictions.

5. CONCLUSION

This work detected and corrected spelling errors in Indonesian documents by calculating distance between misspelled words and corrected words. The distance was calculated following a *Damerau-Levenshtein* algorithm. The prediction correction was then shorted using *n-gram* to the naturalness of the sentences. The spelling error types cover missing character, extra character, replaced character, and transposition character.

The experimental results shows that the model with smallest *med* outperforms others for one misspelled character in a word. It shows that *Damerau-Levenshtein* algorithm has high sensitivity enough of 85% even though the coverage is only 74% for this experiment. It also work well for the transposition characters.

The future work is to try other algorithms for words that contains more than one spelling errors.

REFERENCE

[1] Y. Rochmawati and R. Kusumaningrum, "Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks," *Jurnal Buana Informatika,* vol. 7, no. 2, 2016.

[2] A. I. Fahma, I. Cholissodin and R. S. Perdana, "Spelling Error Identification (Typographical Error) in Indonesian Documents using N-gram and Levenshtein Method," *Journal Pengembangan Teknologi Informatika dan Ilmu Komputer,* pp. 53 - 62, 2018.

[3] D. Q. Thang and P. T. Huy, "Determining Restricted Damerau-Levenshtein Edit-Distance of Two Languages by Extended Automata," *IEEE; http://dx.doi.org/10.1109/RIVF.2010.5632914,* 2010.

[4] T. Maghfira, I. Cholissodin and A. Widodo, "Deteksi Kesalahan Ejaan dan Penentuan Rekomendasi Koreksi Kata yang Tepat Pada Dokumen Jurnal JTIIK Menggunakan Dictionary Lookup dan Damerau-Levenshtein Distance," *Jurnal Pengembangan Teknologi Informasi dan Informasi dan Ilmu Komputer,* vol. 1, no. 6, pp. 495 - 506, 2017.

[5] D. Jurafsky and J. H. Martin., Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, US: Pearson Education, 2007.

[6] U. Sutisna et al., "Koreksi Ejaan Query Bahasa Indonesia Menggunakan Algoritme Damerau Levenshtein," *Jurnal Ilmiah Ilmu Komputer,* vol. 8, no. 2, p. 25–29, 2010.

[7] V. C. Mawardi, N. Susanto and a. D. S. Naga, "Spelling Correction for Text Documents in Bahasa Indonesia Using Finite State Automata and Levenshtein Distance Method," in *MATEC Web Conference vol. 164*, 2018.

[8] J. Jupin, J. Y. Shi and Z. Obradovic, "Understanding Cloud Data using Approximate String Matching and Edit Distance," in *SC Companion High Performance Computer Network Storage Analysis SCC*, 2012.

[9] B. B. C. r. Samanta, "A Simple Real-world Error Detection and Correction using Local Word Bigram and Trigram," in *The Twenty-Fifth Conference on Computational Linguistics and Speech Processing*, 2013.

[10] D. Dahlmeier and H. Ng, "Grammatical error correction with alternating structure optimization," in *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies Volume 1*, Stroudsburg, 2011.

[11] M. Abadi, A. Agarwal and e. al., "TensorFlow: Large- scale machine learning on heterogeneous systems.",," in *USENIX Association*, 2015.

[12] W. A. Gale and K. W. Church, "What is Wrong with Adding One?," in *Nelleke Oostdijk and Pieter de Haan," Rodopi: Corpus-Based Research into Language"*, 1994, pp. 189-198.