COMPARISON OF SMOOTHING METHODS FOR DETECTION AND CORRECTION ERRORS WORD IN INDONESIAN TEXT

Fernando Sihole[1], Kania Evita Dewi[2]

[1.2]Information Engineering - University Computer Indonesia
Jl.Dipatiukur 112-113 Bandung
E-mail : nitrousfernando@gmail.com[1], kania.evita.dewi@email.unikom.ac.id[2]

ABSTRACT

Topics in NLP (Natural Language Processing), said errors are important to note. Some research has been done to address the problem of real-word spelling errors in English. Thesis Muhammad Aburizal Siregar has overcome the problem of real-word spelling errors in Indonesian that said error detection and correction of real-word has an accuracy of 11% after the change probability calculation by using the n-gram smoothing additive method. Therefore, this study aimed to membandigkan accuracy of the method of detection and correction of spelling mistakes real-word in the Indonesian language using the smoothing Good-Turing estimate, Jelinek-Mercer, Katz smoothing, Witten-Bell and Absolute discounting for case detection and error correction estate -Word in Indonesian.

,

Keywords :spelling error detection, correction, real-word, n-grams, confusion sets.

1. INTRODUCTION

Thesis Muhammad Aburizal Siregar has overcome the problem of real-word spelling errors in Indonesian based n-gram method local error detection and correction of real-word word has an accuracy of 11% [1]. after changing the n-gram probability calculations using smoothing additive method, the value is very small compared with Samanta and Chaudhuri research by 80% [2]. Therefore, in this study will be taken some method of smoothing for case detection and error correction of real-word in the Indonesian language taken from the corpus PAN-BPPT and UI Tagged Corpus and will be added to the corpus of English Wikipedia in 2016 amounted to 20 million words which translate into Indonesian. Methods adopted are methods Bigram and local trigram proposed Samanta and Chaudhuri [2].

2. ANALYSIS AND SYSTEM DESIGN

2.1 Process Analysis

The analysis process is performed preprocessing n-gram corpus manufacture, preprocessing detection and error correction and error detection and correction process real-word. Here's the explanation.

2.2 Input Data Analysis

The input data used in this study is the test data, a list of words, and the text corpus. Here is an explanation of each data input.

2.2.1 Glossary

Glossary needed to create confusion sets. Confusion set is the set of words commonly confused with each other from the word. Glossary retrieved from the database Indonesian dictionary .csv format. Here is an example snippet database Indonesian dictionary. Glossary created by taking words from the database column Indonesian Dictionary above. The word is just a word taken the basic course, and filtering is done by removing the dashes. The word list is then stored in a database. This word list to be called when the system will create confusion sets the stage Word Error Detection and Correction.

2.3 Preprocessing Corpus N-Gram

The corpus n-gram is used to calculate the probability of a word. N-gram corpus contains a list of n-grams and the number of occurrences of the n-gram in all the documents of the corpus and test data [2]. Steps to create a corpus of n-gram is a folding case, tokenisasi sentences, filtering punctuation, word tokenisasi [4], the manufacture of the n-gram shaft, the calculation of n-gram shaft and into the corpus n-gram,

2.4 Preprocessing Error Detection and Correction

preprocessingThis will be done to prepare the test data in order to do the detection and correction of real-word spelling error on the test data. Preprocessing stages error detection and correction is equal to three initial stages of the manufacturing preprocessing n-gram shaft, which is a folding case, tokenisasi sentence, and tokenisasi word [4].

2.5 Detection and Correction of Spelling Errors Real-Word

The process of spelling error detection and correction of real-word begins with the creation of confusion set for each of the candidates. After the

recount confusion rank each element in the set. Based on the ranking results, error detection and correction word suggestions given. This is done verbatim [2].

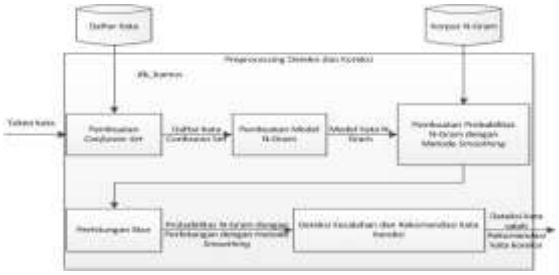


Figure 1 Diagram Error Detection and Correction Process Words

In this analysis we will use the test data has been preprocessing in Section 2.4 in order to facilitate the analysis, letnotation W^i[2] for each token as follows.

Table 1 List of Test Data Token Words

W^i	**Token**
W^0	_
W^1	sketch
W^2	circumstances
W^3	literature
W^4	europe
W^5	trash
W^6	end
W^7	century
W^8	14th
W^9	mid
W^{10}	_

Tokens of words above, will be sought Which words are real-word errors. If an error is detected, will be given the correct word suggestion to correct the wrong word.

2.5.1 Making Confusion Sets

First generate confusion set for each token word, ie the set of words that is a variant spelling of the word token. This is obtained by calculating Levenshtein distance of each token in the word list. Levenshtein distance is the minimum numberedit operations needed to transform one word into another word. Edit operation is the addition (insertion), deletion (deletion) or replacement (substitution) one character in a word.The result of the calculation can be seen in the last column of the last row matrix [5]. From the results of a calculation, found that the value of Levenshtein distance from the comparison of the word "state" and "keagaan" Levenshtein distance is 1. Counting is done for all the words in a dictionary word list. Here are the results of calculation of Levenshtein distance for the word "state".

Words that made members of confusion set are words in the word list that has a maximum value of Levenshtein distance 1. Confusion sets can be represented as

$$C(W^i) = \{c_1^i, c_2^i, \dots, c_j^i, \dots, c_{k_i}^i\} \quad (1)$$

where is the ith word in the sentence and is the number of elements in$W^i k_i$ $C(W^i)$, Here is the confusion set for each word in the test data.

Table 2 Results Members Making Confusion Set for Test Data

W_i	**word**	c_j^i	**Member**	**value Levenshtein**
W^1	sketch	c_1^1	sketch	1
W^2	circumstances	c_1^2	circumstances	0
		c_2^2	refinement	1
		c_3^2	keagaan	1
...	...	...	...	...

2.5.2 Modeling N-Gram

After it is done making a model of n-grams for each member of the confusion sets [2]. For each member, raised Bigram left, right and trigram Bigram by taking the word on the left, right, and both (for trigrams) of (taken from table 3.20). Then Bigram and trigram formed are:$C(W^i)W^i$

Bigram left: $W^{i-1}c_j^i$ (2)

Bigram right: $c_j^i W^{i+1}$ (3)

trigram: $W^{i-1}c_j^i W^{i+1}$ (4)

Where . $1 \leq j \leq k_i$ (5)

for words at the beginning and at the end of a sentence, empty words marked with an underscore (_). Preparation of n-gram models is made to all members of confusion set for each word. After that saw the number of occurrences of each Bigram and trigrams in the corpus n-gram. Bigram and trigram corpus is used to find the number of occurrences is based on Table 3.15 and Table 3.16. The number of occurrences n-gram in the corpus is denoted by

$count(c_j^i)$ For the number of occurrences unigram (6)

$count(W^{i-1}c_j^i)$ For the number of occurrences Bigram left, (7)

$count(c_j^i W^{i+1})$ for the number of occurrences Bigram right, (8)

$count(W^{i-1}c_j^i W^{i+1})$ for the number of occurrences of the trigram.

(9)

The following tables each featuring a number of occurrences of unigram, Bigram left, right and trigram Bigram on the test data. If an n-gram is not found in the corpus, it is given a value of 0. Here is a list of the emergence of left Bigram corpus of test data in the n-gram. For example, for a token = "junk", to members of confusion set = "to", the number of occurrences Bigram left = "Europe to" the n-gram corpus is 1.$W^5 c_3^5 W^5 c_j^5$

table 3 List of Test Data Number of Occurrences Unigram

c_j^0		**count**(c_j^1)
c_1^0	_	2
c_j^1		**count**(c_j^1)
c_1^1	sketch	1
c_j^2		**count**(c_j^3)
c_1^2	circumstances	1
c_2^2	refinement	0
c_3^2	keagaan	0
...		**....**

Here is a list of occurrences in the corpus of test data unigram n-gram. For example, for a token = "junk", to members of confusion set = "to", the number of occurrences unigram = "up" in the corpus n-gram is 1.$W^5c_3^5c_3^5W^5$

Table 4 List Number of Occurrences Bigram Left Test Data

c_j^1		**Bigram Kiri** $(W^0c_j^1)$	**count** $(W^0c_j^1)$
c_1^1	sketch	_ sketch	0
c_j^2		**Bigram Kiri** $(W^1c_j^3)$	**count** $(W^1c_j^3)$
c_1^2	circumstances	sketch circumstances	1
c_2^2	refinement	sketch refinement	0
c_3^2	keagaan	sketch keagaan	0
...		...	**...**

Here is a list of the test data right Bigram occurrences in the corpus n-gram. For example, for a token = "junk", to members of confusion set = "to", the number of occurrences Bigram right = "until the end" in the corpus n-gram is 1.$W^5c_3^5c_3^5W^6$

table 5 List Number of Occurrences Bigram Right Test Data

c_j^1		**Right Bigram** $()c_j^1W^2$	**count** $()c_j^1W^2$
c_1^1	sketch	sketch circumstances	1
c_j^2		**Right Bigram** $()c_j^2W^3$	**count**$()c_j^2W^3$
c_1^2	circumstances	state of the literature	1
c_2^2	refinement	literary refinement	0
c_3^2	keagaan	keagaan literature	0
...		**...**	**...**

Here is a list of occurrences in the corpus of test data trigram n-gram. For example, for a token = "junk", to members of confusion set = "to", the number of occurrences trigramnya = "Europe until the end of" the n-gram corpus is 1.$W^5c_3^5W^4c_3^5W^6$

table 6 List Number of Occurrences Trigram Test Data

c_j^1		**trigram** $()W^0c_j^1W^2$	**count** $()W^0c_j^1W^2$
c_1^1	sketch	_ Sketch circumstances	0
c_j^2		**trigram** $()W^1c_j^2W^3$	**count()** $W^1c_j^2W^3$
c_1^2	circumstances	sketches the state of literature	1
c_2^2	refinement	sketches of literary refinement	0
c_3^2	keagaan	keagaan literary sketch	0
...		**...**	**...**

table 7 Number of Occurrences Frequency Each Unigram, Bigram and Trigram

r Bigram	nr	**r** trigram	nr
0	20	0	15
1	42	1	41
2	1	2	0
3	0	3	0

The results of the calculation of the number of occurrences in the manufacture of its corpus is calculated each frequency with the letter "r" and the number of frequencies "Nr". When adding up the number of times seen from the time of making the n-gram corpus before, when searching for the frequency number 0 can be obtained from the time of detection and correction of errors.

Then also count the total number of occurrences of the n-gram for all members of confusion set for each word. The total number of occurrences is denoted by $\sum_{r=1}^{k_i} count(c_r^i)$ for the total number of occurrences unigram. (10)

$\sum_{r=1}^{k_i} count(W^{i-1}c_r^i)$for the total number of occurrences Bigram left. (11)

$\sum_{r=1}^{k_i} count(W^{i+1}c_r^i)$for the total number of occurrences Bigram right. (12)

$\sum_{r=1}^{k_i} count(W^{i-1}c_r^iW^{i+1})$ for the total number of occurrences of the trigram. (13)

k is the number of members of the confusion sets a word and r is the order of the member of the confusion set. Here are the results of counting the total number of occurrences Bigram and trigrams for test data.

Table 8 Total Number of Occurrences Confusion Set Unigram and Bigram Left

W^i	**Token**	$\sum_{r=1}^{k_i} count(c_r^i$	$\sum_{r=1}^{k_i} count(W^{i-1}c$
W^1	sketch	$count(c_1^1) = 1$	$count(W^0c_1^1) = 0$
W^2	circumstances	$count(c_1^2) +$	$count(W^1c_1^2) +$

		$count(c_2^2)$ + $count(c_3^2)$ = 1 + 0 + 0 = 1	$count(W^1c_2^2)$ + $count(W^1c_3^2)$ = 1 + 0 + 0 = 1
...	...	...	...

Table 9 Total Number of Occurrences Confusion Bigram Set Right

W^i	Token	$\sum_{r=1}^{k_i} count(c_r^i W^{i+1})$
W^1	sketch	$count(c_1^1 W^2) = 1$
W^2	circumstances	$count(c_1^2 W^3) + count(c_2^2 W^3) + count(c_3^2 W^3) = 1 + 0 + 0 = 1$
...	...	...

Table 10 Total Number of Occurrences Confusion Set Trigram

W^i	Token	$\sum_{r=1}^{k_i} count(W^{i-1}c_r^i W^{i+1})$
W^1	sketch	$count(W^0 c_1^1 W^2) = 0$
W^2	circumstances	$count(W^1 c_1^2 W^3) + count(W^1 c_2^2 W^3) + count(W^1 c_3^2 W^3) = 1 + 0 + 0 = 1$
...	...	...

2.5.3 Calculation of Probability N-Gram

The author's purpose here to see how the results of the comparison of the calculation of the probability of n-grams in the case of correction of spelling mistakes real-word Indonesian with the method of smoothing Good-Turing estimate, Jelinek-Mercer, Katz smoothing, Witten-Bel and Absolute discounting that have been studied by Stanley F. Chen and Joshua Goodman [3].

2.5.3.1 Smoothing Methods Good-Turing

Formula of good-Turing smoothing method [6] is.$\frac{1}{N}(r+1)\frac{N_{r+1}}{N_r}$ (14)

$N = \sum_r rN_r$ (15)

r is the frequency of occurrence of the word.
Nr is the number of frequency of words.

2.5.3.2 Smoothing Methods Jelinek-Mercer

Jelinek-mercer method [7] using interpolation method, which is associated with the n-gram under tingkatanya.
Value λ = 0.1 Left Bigram

$P_1'\left(c_j^i \middle| W^{i-1}\right) = \lambda\, P_1(c_j^i|W^{i-1}) + (1\text{-}\lambda)\, P_0(c_j^i)$ (16)

Right Bigram $P_2'(c_j^i|W^{i+1}) = \lambda\, P_2(c_j^i|W^{i+1}) + (1\text{-}\lambda)\, P_0(c_j^i)$ (17)

trigram

$P_3' = (c_j^i|W^{i-1}, W^{i+1}) = \lambda\, P_3\left(c_j^i|W^{i-1}, W^{i+1}\right) + (1-\lambda)(P_1\left(c_j^i|W^{i-1}\right) + P_2\left(c_j^i|W^{i+1}\right)/2)$ (18)

$P_0(C_j^i)$ is indigo probabitas maximum likelihood for uniram.

$P_1\left(C_j^i|W^{i-1}\right)$ is indigo probabitas maximum likelihood for Bigram left.

$P_2\left(C_j^i|W^{i+1}\right)$ is indigo probabitas Bigram maximum likelihood for the right.

$P_3\left(W_j^i|W^{i-1}, W^{i+1}\right)$ is indigo probabitas maximum likelihood for trigrams.

2.5.3.3 Smoothing Methods Katz

Based on the method of Katz smoothing formula equation [1] there are two options, namely if and when, as well as methods Jelinek mercer, need to use the maximum likelihood probability value.$r > 0 r = 0$

1. If you use the formula $r > 0 P_1(c_j^i|W^{i-1}) = dr$

   $dr = \frac{r^*}{r},\ r^* = (r+1)\frac{N_{r+1}}{Nr},$ (19)

   r = frequency of$count(c_j^i|W^{i-1})$
   r is the frequency of occurrence of the word.
   Nr is the number of frequency of words.
2. If you use the formula $r = 0$
   Left Bigram $\alpha\left(c_j^i|W^{i-1}\right)P_0\left(c_j^i\right)$ (20)

   $\alpha\left(c_j^i|W^{i-1}\right) = \frac{1-\sum_{w_i:count\left(W^{i-1}c_r^i\right)>0} P_1(c_j^i|W^{i-1})}{1-\sum_{w_i:count\left(c_r^i\right)>0} P_0(c_j^i)}$ (21)

   Right Bigram $\alpha\left(c_j^i|W^{i+1}\right)P_0\left(c_j^i\right) =$ (22)

   $\frac{1-\sum_{w_i:count\left(W^{i+1}c_r^i\right)>0} P_2(c_j^i|W^{i+1})}{1-\sum_{w_i:count\left(c_r^i\right)>0} P_0(c_j^i)}$ (23)

   trigram

   (24)$\alpha\left(c_j^i|W^{i-1}, W^{i+1}\right)(P_1\left(c_j^i|W^{i-1}\right) + P_2(c_j^i|W^{i+1})/2)$

   $\alpha\left(c_j^i|W^{i-1}, W^{i+1}\right) = \frac{1-\sum_{w_i:count\left(W^{i-1},W^{i+1}c_r^i\right)>0} P_3(c_j^i|W^{i-1},W^{i+1})}{1-\sum_{w_i:count\left(W^{i-1}c_r^i\right),count\left(W^{i+1}c_r^i\right)>0}(P_1(c_j^i|W^{i-1})+P_2(c_j^i|W^{i-1}))/2}$ (25)

   $\sum_{w_i:count\left(c_j^i\right)>0} P_0\left(c_j^i|W^{i-1}\right)$ is a number that has a number of occurrences of unigram more than 0.W_i

   $\sum_{w_i:count\left(c_j^i w_{i-1}\right)>0} P_1\left(c_j^i|W^{i-1}\right)$ is a number that has a number of occurrences of the right Bigram more than 0.W_i

   $\sum_{w_i:count\left(c_j^i w_{i+1}\right)>0} P_2\left(c_j^i|W^{i+1}\right)$ is a number that has a number of occurrences of the right Bigram more than 0.W_i

$\sum_{w_i:count\left(c_j^i w_{i-1}\right)>0} P_3\left(c_j^i|W^{i-1},W^{i+1}\right)$ is a number that has a number of occurrences of the trigram more than 0.W_i

$\sum_{w_i:count(W^{i-1}c_r^i),count(W^{i+1}c_r^i)>0}(P_1(c_j^i|W^{i-1}) + P_2\,(c_j^i|W^{i-1}))/2$is a number that has a number of occurrences of Bigram left and right Bigram more than 0.W_i

2.5.3.4 Smoothing Method Witten-Bell

Here's the formula for Witten-Bell smoothing method [7]:

1. The formula for the calculation of the left Bigram

$$P_1'(c_j^i|W^{i-1}) = \frac{count\left(W^{i-1}C_j^i\right) + N_{1+}\left(c_j^i|W^{i-1}\cdot\right)P_1(c_j^i|W^{i-1})}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i)+0.01\times V_j+N_{1+}\left(c_j^i|W^{i-1}\cdot\right)} \quad (26)$$

$$N_{1+}\left(c_j^i|W^{i-1}\cdot\right) = \left|\{W^i: count(c_j^i W^{i+1}) > 0\}\right| \quad (27)$$

$N_{1+}\left(c_j^i|W^{i-1}\cdot\right)$ is the occurrence of the word of Bigram kemuncula left has a value greater than 0.

2. Bigram calculation formula right

$$P_2'(c_j^i|W^{i+1}) = \frac{count\left(W^{i+1}C_j^i\right) + N_{1+}\left(c_j^i|W^{i+1}\cdot\right)P_2(c_j^i|W^{i+1})}{\sum_{r=1}^{k_i} count(W^{i+1}c_r^i)+0.01\times V_j+N_{1+}\left(c_j^i|W^{i+1}\cdot\right)} \quad (28)$$

$$N_{1+}\left(c_j^i|W^{i+1}\cdot\right) = |\{W^i: W^i count(c_j^i W^{i+1}) > 0\}| \quad (29)$$

$N_{1+}\left(c_j^i|W^{i-1}\cdot\right)$ is the occurrence of the word of Bigram right that has the appearance value greater than 0.

3. The formula for the calculation of the trigrams.

$$(30)\ P_3'(c_j^i|W^{i-1},W^{i+1}) = \frac{count\left(W^{i-1},W^{i+1}C_j^i\right) + N_{1+}\left(c_j^i|W^{i-1},W^{i+1}\cdot\right)P_3(c_j^i|W^{i-1},W^{i+1})}{\sum_{r=1}^{k_i} count(W^{i-1},W^{i+1}c_r^i)+0.01\times V_j+N_{1+}\left(c_j^i|W^{i-1},W^{i+1}\cdot\right)}$$

$$N_{1+}\left(c_j^i|W^{i-1},W^{i+1}\cdot\right) = |\{W^i: W^i count(c_j^i W^{i-1},W^{i+1}) > 0\}| \quad (31)$$

$N_{1+}\left(c_j^i|W^{i+1}\cdot\right)$ is the occurrence of the word of trigrams which have the appearance value greater than 0.

2.5.3.5 Smoothing Methods Absolute Discounting

Here's the formula for Absolute Discounting smoothing method [7]:

1. Bigram calculation formula to the left.

$$P_1'(c_j^i|W^{i-1}) = \frac{\max\{count(c_j^i W^{i-1})+0.01-\delta,0\}}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i)+0.01\times V_j} + (1-\lambda)\,P_0(c_j^i). \quad (32)$$

$$(1-\lambda) = \frac{\delta}{\sum_{r=1}^{k_i} count(W^{i-1}c_r^i)+0.01\times V_j} N_{1+}\left(c_j^i W^{i-1}\cdot\right) \quad (33)$$

$N_{1+}\left(c_j^i|W^{i-1}\cdot\right)$ is the occurrence of the word of Bigram kemuncula left has a value greater than 0.

2. Bigram calculation formula to the right.

$$P_2'(c_j^i|W^{i+1}) = \frac{\max\{count(c_j^i W^{i+1}+0.01)-\delta,0\}}{\sum_{r=1}^{k_i} count(W^{i+1}c_r^i)+0.01\times V_j} + (1-\lambda)\,P_0(c_j^i). \quad (34)$$

$$(1-\lambda) = \frac{\delta}{\sum_{r=1}^{k_i} count(W^{i+1}c_r^i)+0.01\times V_j} N_{1+}\left(c_j^i W^{i+1}\cdot\right) \quad (35)$$

$N_{1+}\left(c_j^i|W^{i-1}\cdot\right)$ is the occurrence of the word of Bigram right that has the appearance value greater than 0.

3. The formula for the calculation of trigrams

$$P_3'(c_j^i|W^{i-1},W^{i+1}) = \frac{\max\{count(c_j^i W^{i-1},W^{i+1})+0.01-\delta,0\}}{\sum_{r=1}^{k_i} count(W^{i-1},W^{i+1}c_r^i)+0.01\times V_j} + (1-\lambda)(\,P_1(c_j^i|W^{i-1}) + P_2(c_j^i|W^{i+1})/2\,) \quad (36)$$

$$(1-\lambda) = \frac{\delta}{\sum_{r=1}^{k_i} count(W^{i-1},W^{i+1}c_r^i)+0.01\times V_j} N_{1+}\left(c_j^i W^{i-1},W^{i+1}\cdot\right) \quad (37)$$

$N_{1+}\left(c_j^i|W^{i+1}\cdot\right)$ is the occurrence of the word of trigrams which have the appearance value greater than 0.

2.6 Result And Discussion

Testing the value of accuracy in the application error detection and correction of real-word word is done for to determine the extent to which the results of error detection and correction of real-word word is generated absolutely right. The corpus of n-grams are obtained from the corpus PAN-BPPT total of 500 thousand words, UI-Corpus Tagged as many as 1 million word corpus and Wikipedia in 2016 as many as 20 million words. The text used as the test data is 30 news articles from the www.detik.com and www.kompas.co.id. Accuracy value obtained by dividing the sum by the number of detection system detection is done manually.

The performance of the method of detection and correction of real-word spelling error is measured by measuring the detection accuracy and correction accuracy. Measurement accuracy of detection performed by the equation

$$akurasi\ deteksi = \frac{ds}{dm}\times 100\% \quad (38)$$

Where

ds : The resulting amount detection system

dm : the number of manually generated detection

Sedangkaran measurement accuracy correction is performed by the equation

$$akurasi\ koreksi = \frac{kb}{kb+ks}\times 100\% \quad (38)$$

Where

kb : true correction amount generated by the system

ks : incorrect correction amount generated by the system

Table 11 the results of detection accuracy and correction

Smoothing methods	Detection accuracy	accuracy Correction
Good Turing	0%	0%
Jelinek-Mercer	36%	89%
Katz	50%	24%
Witten-Bell	79%	7%
Absolute Discounting	80%	7%

In the test data of 30 texts being tested often goes wrong when the score was the highest ranking members of confusion set is not enough to state that the word there is wrong. Limit one word declared wrong is if the score is 0 or less than 1% of the highest ranking scores. In this case the score said one of greater than 1% score the highest ratings so that the wrong word is regarded as completely as well as the results of the accuracy of the correction, there are words that should not be corrected but corrected by the system due to its score is still low compared to words that often appear in corpus.

From the results of detection accuracy and correction kesalahan using 21.5 million corpus with five smoothing method can be compared to a journal belonging to Stanley F. Chen and Joshua Goodman [3], that

1. Smothing good-turing method is not good as well as the lack of good accuracy results are also shown in Table 11 with accuracy detesi and correction of 0%.
2. Mercer Jelinek Smothing method is nice, as well as with good accuracy results are also shown in Table 11 with a 36% detection accuracy and correction accuracy by 89%.
3. Katz Smothing method is very good in a large and diverse data therefore the accuracy results shown in Table 11 with a 50% detection accuracy and correction accuracy of 24% is not good for the corpus that is used today.
4. Witten-bell Smothing method is very good on the data rather large so it is with good accuracy results are also shown in Table 11 with a 79% detection accuracy and correction accuracy by 7%.
5. Witten-bell Smothing method is very good on the data rather large so it is with good accuracy results are also shown in Table 11 with a 79% detection accuracy and correction accuracy by 7%.

3. CONCLUSION AND SUGGESTION

This section will explain the conclusions and recommendations of the study.

3.1 Conclusion

Based on the results of testing the accuracy of detection and correction of spelling errors Real-word, the accuracy of a high detection is a method of smoothing Absolute Discounting by 80%, and for the accuracy of correction higher is the method of smoothing Jelinek-Mercer by 89% and the accuracy of the smallest to the accuracy of detection and correction of 0% is Good Turing smoothing method.

3.2 Suggestion

For further research, it is advisable to increase the size of the corpus of n-grams that will be used in order to increase the accuracy of detection and error correction of real-word word. It is also advisable to use a smoothing method such as Add-one and Kneser-Ney Smoothing Smoothing.

REFERENCES

[1] M. Aburizal S. Skripsi, Deteksi Dan Koreksi Kesalahan *Real-Word* Pada Penulisan Kata Menggunakan Metode N-Gram Dalam Teks Berbahasa Indonesia, 2017.

[2] B. B. C. ratip Samanta, *"A simple real-world error detection and correction using local word bigram and trigram," Proceedings of the Twenty-Fifth Conference on Computational Linguistics and Speech Processing*. 2013.

[3] S. F. C. a. J. Goodman, "An Empirical Study Of Smoothing Techniques For Language Modeling," *Computer Speech and Language*, 1999.

[4] F. J. D. d. R. L. M. E. Mays, *"Context based spelling correction,"*. Information Processing and Management, 27(5):527-522,, 1991

[5] M. Gilleland, *"Levenshtein Distance, in Three Flavors"*. Merriam Park Software https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/.

[6] EllisWeng, A. Owens Lecture 11: The Good-Turing Estimate, 4 March 2010.

[7] B. MacCartney, *NLP Lunch Tutorial: Smoothing*. Stanford Natural Language Processing Group, 21 April 2005.