Register 6 (2) 2020 pp. 130-140
Jurnal Ilmiah Teknologi Sistem Informasi (Scientific Journal of Information System Technology)

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

Available online to www.journal.unipdu.ac.id

Unipdu

S2-Accredited – SK No. 34/E/KPT/2018
Journal page is available to www.journal.unipdu.ac.id:8080/index.php/register

Typo handling in searching of Quran verse based on phonetic similarities

Naila Iffah Purwita [a], Moch. Arif Bijaksana [b], Kemas Muslim Lhaksmana [c], Muhammad Zidny Naf'an [d]

[a,b,c] *Department of Informatics, Telkom University, Bandung, Indonesia*
[d] *Department of Informatics, Telkom Institute of Technology Purwokerto, Purwokerto, Indonesia*

email: [a] nailaip@telkomuniversity.ac.id , [b] arifbijaksana@telkomuniversity.ac.id , [c] kemasmuslim@telkomuniversity.ac.id , [d] zidny@ittelkom-pwt.ac.id

ARTICLE INFO

Article history:
Received 22 July 2020
Revised 18 August 2020
Accepted 18 August 2020
Published 27 August 2020

Keywords:
autocomplete
Damerau–Levenshtein distance
phonetic similarity
Quran
typographical error

IEEE style in citing this article:
N. I. Purwita, M. A. Bijaksana, K. M. Lhaksmana and M. Z. Naf'an, " Typo handling in searching of Quran verse based on phonetic similarities," *Register: Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 6, no. 2, pp. 130-140, 2020.

ABSTRACT

The Quran search system is a search system that was built to make it easier for Indonesians to find a verse with text by Indonesian pronunciation, this is a solution for users who have difficulty writing or typing Arabic characters. Quran search system with phonetic similarity can make it easier for Indonesian Muslims to find a particular verse. Lafzi was one of the systems that developed the search, then Lafzi was further developed under the name Lafzi+. The Lafzi+ system can handle searches with typo queries but there are still fewer variations regarding typing error types. In this research Lafzi++, an improvement from previous development to handle typographical error types was carried out by applying typo correction using the autocomplete method to correct incorrect queries and Damerau Levenshtein distance to calculate the edit distance, so that the system can provide query suggestions when a user mistypes a search, either in the form of substitution, insertion, deletion, or transposition. Users can also search easily because they use Latin characters according to pronunciation in Indonesian. Based on the evaluation results it is known that the system can be better developed, this can be seen from the accuracy value in each query that is tested can surpass the accuracy of the previous system, by getting the highest recall of 96.20% and the highest Mean Average Precision (MAP) reaching 90.69%. The Lafzi++ system can improve the previous system.

1. Introduction

Indonesia is one country that obliges its people to embrace a religion. The majority of Indonesia's population is Muslim. At present, there are more than 207 million Muslims in Indonesia [1]. The primary source of Muslim teachings in the world is the Quran, which was revealed gradually to the Prophet Muhammad for 23 years using Arabic language and writing. The Quran has 30 Juz, consisting of 6326 verses with 114 Surahs [2]. When viewed from the number of verses as many as 6326, then the number of words in the Quran will be far more than the number of verses, so if a user wants to do a word search manually, it will take a lot of time.

The development of application systems and software related to the Quran has been done for a long time to meet the needs of users, including Tanzil and IslamiCity. Tanzil is an application to search words using Arabic characters, while IslamiCity is similar to Tanzil but in its search uses international Latin letters. In Indonesia, any writing accustomed to using Latin script will be difficult for Indonesian people when searching for words with Arabic script. The solution to overcome this difficulty is to develop a verse search system based on phonetic similarity. Phonetic search will make it easier to find

ISSN 2502-3357 (Online)
N. I. Purwita et al./Register 6 (2) 2020 pp. 130-140 ISSN 2503-0477 (Print)

a string that is similar to the pronunciation of similar words [3]. One researcher from Indonesia has succeeded in developing a phonetic-based Quran search system called the Lafzi application. Lafzi is an application system that is used to search verses in Latin script which is suitable for Indonesian pronunciation [4]. The transition of Arabic into Latin script will result in differences in sound from the original. Therefore it must be transliterated into phonetic code. The Lafzi application can correct writing errors because the sound of Arabic letters is almost the same as pronunciation but does not correct typing errors. The user will find it difficult to search for verses because they have to type each verse correctly. If the user commits typos, even if only one letter is wrongly typed, the system will not display the verse in which the user looking for. The cause of typographical errors is usually wrong in the placement of finger positions on the keyboard. For example, a user wants to write a query ("*MASALULAZINA*") but actually type ("*MASALUKAZINA*"). This can happen because the position of the letters "*L*" with "*K*" on the QWERTY keyboard is close together. The appearance of a suggestion is needed to make it easy for the user to correct errors in the query so that the user can find the appropriate search results.

Previous research has been conducted on the Quranic Latin Query Correction as a Search Suggestion [5], but the research is still less variation in the type of typing errors, only able to handle the types of typo namely substitution, deletion, and insertion. Whereas in typing errors, there are four types of typos, namely typo substitution, deletion, insertion, and transposition. So far, typing errors is not only the insertion, replacing, or sweeping of characters, but there is an error in exchanging two adjacent characters, which is referred to as a transposition. The Damerau Levenshtein Distance method can overcome word errors better than the Levenshtein Distance method because it can reach a wider variety of errors and can improve the accuracy results [6]. Thus, in this final project, the various types of typing errors will be increased by developing a correction query system. The type of typo that will be developed is adding a transposition typo, and increasing the accuracy of other typo types. There are four types of typos in this system: typo substitution, typo insertion, typo deletion, and typo transposition. The development of this system uses autocomplete and Damerau Levenshtein distance. Autocomplete is used to complete the missing trigrams in each verse so that it can correct incorrect queries and the Damerau Levenshtein distance is used to calculate edit distances on two strings, the initial query and the suggested query. The calculation of the edit distance is used to determine the similarity between the input query and the suggested query. Then the value will be compared, the smallest suggestion query value from the edit distance calculation will be used as an output suggestion on the system [7].

2. Literature Review

2.1. Related work

Lafzi [4] is a search for lafaz in the Quran based on phonetic similarity. This system can make it easier to find verses. Searching for this system uses the Latin script adjusted to the pronunciation of Indonesian people. Trigrams are used as search methods that are applied to phonetic code. The results of several information retrieval tests for Arabic show that the trigram is the best gram number for indexing Arabic texts. This system has a disadvantage that is not able to overcome typing errors, only able to overcome errors based on pronunciation.

Table 1. The evaluation result of Lafzi and Lafzi+

Query Type	Lafzi		Lafzi+	
	Recall	MAP	Recall	MAP
Normal Query	16.21%	77.60%	16.21%	77.60%
Subtitution Query + Weighted	30.21%	19.67%	77.60%	76.00%
Subtitution Query + Non Weighted	30.21%	19.67%	77.63%	76.00%
Deletion and Insertion Query + Weighted	85.23%	79.83%	93.22%	86.00%
Deletion and Insertion Query + Non Weighted	85.23%	79.83%	93.40%	86.00%

The second research is the development of Lafzi. Lafzi+ [5] is the development of the Lafzi system, which is to improve the function of the system. In this case, the addition of functions and improvement of existing systems. The function added to Lafzi+ is a cross-verse search. There is a Lafzi system available with several functions, enhanced functions including searching for clipped verses, sound changes to stop signs, and incorrect query handling. The weaknesses in this study are the types of typo grouping

N. I. Purwita et al./Register 6 (2) 2020 pp. 130-140
ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

is only based on substitution, deletion, and insertion, while typing errors can occur because of exchanging with adjacent letters namely transposition.

The comparison of Recall and MAP calculation results from the Lafzi and Lafzi+ systems can be seen in Table 1. Based on Table 1, the Lafzi+ system can surpass the Lafzi system by obtaining the highest MAP of 86.00% and Recall of 93.40%, while the Lafzi system is only able to get a MAP value of 79.83% and Recall of 85.23%.

Islamicity is a search system similar to Lafzi, which is a search for the Quran based on the phonetic text [8]. This system is intended for international Muslims, for searching using the international Latin script. The output in this system is in the form of a Quranic verse with a verse translation. The advantage of this system is that it can translate into various languages but cannot translate into Indonesian. Besides that, another weakness in this search by using international Latin script makes it difficult to use for Indonesian people. Islamicity also can't handle verse lookups with typo queries. The phonetic code rules made in the system do not match the pronunciation of the Quran in Indonesia.

Research to correct spelling of words has been done a lot, such as Yeh, Chang, Liu, and Hsu [9], which researched correct spelling in Chinese characters using Knuth-Morris-Pratt (KMP). This system works by extracting data from NLP-TEA then tagging it automatically to segment the word so that the POS tag results will be obtained. Next, check the spelling of Mandarin by determining whether the words are correct or not based on the POS tag that is made. Then KMP will work to correct the wrong word by providing the correct word correction results. Based on the test results, this system has a weakness in correcting the spelling so that the result of the recall is only 34%, and the precision is 62%. In another study, Hossain, Labib, Rifat, Das, and Mukta [10] developed a system for correcting words in Bengali transliteration using Levenshtein distance. Processing in the system by taking a word from the Bangla language transliterated using a text parser and then used as input. The word is spelled incorrectly, and then the system will split the word. The input will be compared with the Bangla corpus data by calculating the distance using the Levenshtein distance. If the result is above 65%, the word will be entered as suggested. In correcting words using the unigram method. The weakness of this suggestion is that word correction may not be raised if the calculation result is less than 65%. It uses the Levenshtein distance method so that it can only calculate errors in the types of substitution, insertion, and deletion only. According to [11] on his proposal in dealing with typos for dealing with medical texts in the world of medicine, Damerau Levenshtein distance handles word errors better than other distance metrics. From the accuracy results compared to the Levenshtein distance, Damerau can outperform it with an accuracy of 76%. In comparison, the results of the Levenshtein accuracy are only 74.75%, and this happens because, in the calculation of the error, the Damerau method can calculate the type of transposition error.

2.2. Character N-Gram

The N-Gram methodology is widely used in language modeling to predict the next word based on the order of the previous word, in other words, that the probability of the next word depends on the *n-1* of the last word [12]. In addition to being able to predict words, the N-Gram method can also handle spelling errors on queries by detecting the spelling of characters [13]. N-gram modeling consists of two levels, namely N-Gram words, and N-Gram characters [14]. This study is using N-Gram characters because the use of N-Gram characters for handling typo is more suitable than N-Gram words. Examples of some N-Gram characters can be seen in Table 2 using the string "*Wahuwayahsa*".

In general, N-Gram has three types of models, namely unigram, bigram, and trigram. The character string will be divided into *n* letter characters. Table 2 shows the bigram's modeling in character chopping by looking at one previous character to be the next character chunk. Meanwhile, to chop a string in a trigram model requires two last letters to be used as the next piece. Unigram has a string of size $n = 1$, bigram with a length of string $n = 2$, and a trigram that has a length of string $n = 3$. The advantage of using N-Gram in string matching is that if an error occurs in some strings it tends not to affect other strings because of its characteristics which divide the string into small parts [15].

2.3. Typographical error

Typographical error is an error in the process of typing words, so it can change that word. Most of the text in any form can contain typos, such as text communication, e-mail, or chat [16]. In addition to text

ISSN 2502-3357 (Online)
N. I. Purwita et al./Register 6 (2) 2020 pp. 130-140 ISSN 2503-0477 (Print)

communication, typing errors can also occur when using a search engine to do a word search. Thus affecting the search result, so that the result is not found or does not match the desired search. Typing errors can occur due to several factors including the lack of knowledge of writing the desired query or due to the distance of the layout on the adjacent keyboard so that it allows the user to press characters incorrectly [17]. There are four types of typos, namely, insertion, deletion, substitution, and transposition.

Table 2. Examples of N-Gram characters

N-Grams Model	Example of N-Grams
Unigram (n=1)	*W,A,H,U,W,A,Y,A,H,S,A*
Bigram (n=2)	*WA,AH,HU,UW,WA,AY,YA,AH,HS,SA*
Trigram (n=3)	*WAH,AHU,HUW,UWA,WAYAYA,YAH,AHS,HSA*

Table 3. Example of query typos

Type of Typos	Example of Typos
Insertion	*ZALIKALFAWZUL'AZIIIM*
Deletion	*ZALIALFAWZUL'AZIM*
Substitution	*ZALILALFAEZIL'AZIM*
Transposition	*ZLAIKALFAWZUL'AZIM*

Table 3 shows an example of a typo based on the kind of typo with the string "*ZALIKAL-FAWZUL'AZIM*". The first type is the insertion, in Table 3, there is an example of the typo insertion "*ZALIKALFA-WZUL'AZIIIM*", this error happens by inserting the character "*II*" in the word. The next type is the deletion with the example "*ZALIALFAWZUL'AZIM*" the error happens by removing the "*K*" character in the string. The third type, substitution with the example "*ZALILALFAEZIL'AZIM*". The error in this string is occurred by replacing the character "*K*" with "*L*". The "*W*" character is replaced by "*E*". and the "*I*" character replacing the "*U*" character. This character replacement is based on proximity to other characters on the keyboard. The last type is transposition with an example of "*ZLAIKALFAWZUL'AZIM*". The error in the string is by swapping the "*A*" and "*L*" characters. This type of typo does swap characters with adjacent characters in a string.

2.4. Longest Common Subsequences (LCS)

Longest Common Subsequences (LCS) a classic problem in the field of computers used to find the maximum length of a sequence of two strings [18]. For example, string 1 "*BIMILLAHIMAJREHA*", string 2 "*BISMILLAHIRAHMAN*", then common subsequences found from these two strings are "*BISMILLAHI*" with a length of 10. LCS is widely used to analyze time series, file comparisons, and biometrics [19, 20]. Besides being used in computer science, it can also be used in biology to sort DNA or RNA [21, 22]. LCS algorithm can compare two strings by calculating the difference. So this method can correct spelling mistakes. LCS can be completed in $O(mn)$ using dynamic programming, where m and n are the lengths of the two input sequences.

2.5. Damerau Levenshtein distance

Problems with string correction require changing strings from one string to another by finding the order of the edit operations. Therefore we need a method for correcting strings. Levenshtein Distance is an algorithm used for string processing, discovered in 1965 by a Russian researcher named Vladimir Levenshtein. This algorithm is also known as edit-distance. Over time, scientist Frederick J. Damerau developed an edit-distance algorithm that made it possible to calculate the editing of transposition between two characters [23]. So with the Damerau Levenshtein method in comparing distances between strings, it is necessary to pay attention to four types of typing errors, namely insertion or addition of a letter, deletion of a letter, replacement of a letter with another letter, and exchange of a consecutive letter [24]. To find the similarity of distance values from two compared strings, we use the formula in Equation 1 [25, 26].

The calculation of edit distance in Equation 1 uses two strings $\boldsymbol{d_{a,b}(i,j)}$. Where a is the first string to be checked and b is the second string. For i and j, each is a character from the string a and string b. For calculations using Damerau, we only need to modify the Levenshtein distance algorithm. This algorithm has four counting operations, where $\boldsymbol{d_{a,b}(i-1,j)+1}$ is the operation for character deletion. $\boldsymbol{d_{a,b}(i-1,j)+1}$ is the operation for inserting characters, $\boldsymbol{d_{a,b}(i-1,j-1)+1}$ is used for character

N. I. Purwita et al./Register 6 (2) 2020 pp. 130-140
ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

replacement operations, and the last operation, $\boldsymbol{d_{a,b}(i-2,j-2)+1}$, is an additional operation for calculating the transposition characters.

$$d_{ab}(i,j)=\begin{cases} \boldsymbol{max(i,j)} & \boldsymbol{if\ min(i,j)=0}, \\ \boldsymbol{min}\begin{cases} \boldsymbol{d_{a,b}(i-1,j)+1} \\ \boldsymbol{d_{a,b}(i,j-1)+1} \\ \boldsymbol{d_{a,b}(i-1,j-1)+1(a_i \neq b_j)} \\ \boldsymbol{d_{a,b}(i-2,j-2)+1} \end{cases} & \boldsymbol{if\ i,j>1\ and\ a_i=b_j-1\ and\ a_i-1=b_j} \\ \boldsymbol{min}\begin{cases} \boldsymbol{d_{a,b}(i-1,j)+1} \\ \boldsymbol{d_{a,b}}(\mathrm{i,j\text{-}1})+1 \\ \boldsymbol{d_{a,b}(i-1,j-1)+1(a_i \neq b_j)} \end{cases} & \boldsymbol{otherwise}. \end{cases} \quad (1)$$

3. Research methods

3.1. System overview

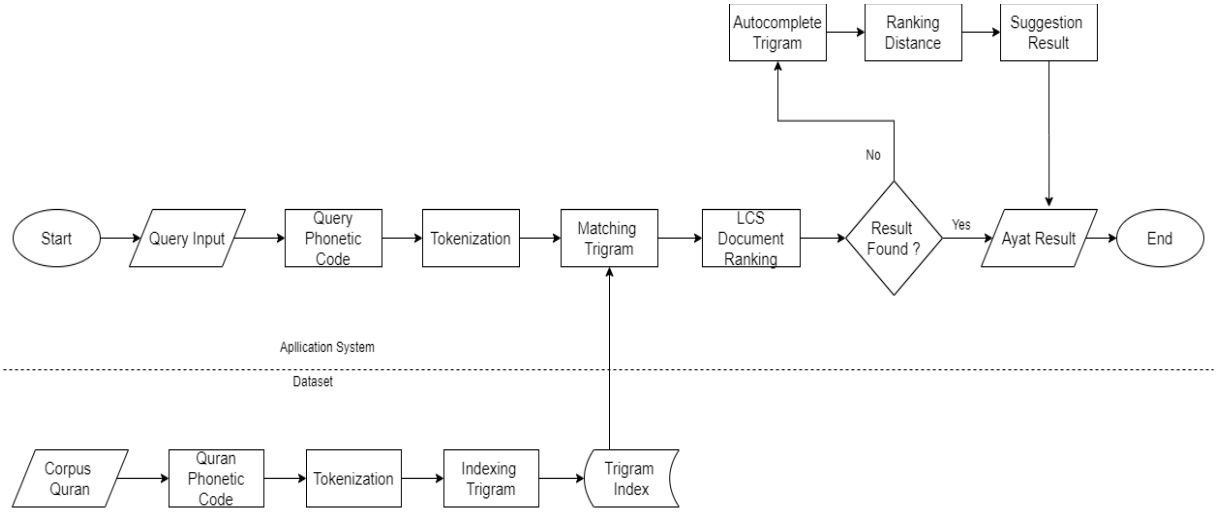

Figure 1. Flowchart system overview

The research that was conducted produced a system that can facilitate the search for verses of the Quran. This system can correct search queries to tolerate user typing errors. Parallel search system flow is illustrated in the form of a flowchart as shown in Figure 1. Processing on the system is divided into two, namely data processing and application system processing. In the dataset process, using Quran text data taken from tanzil to be used as the corpus of the Quran, the data is converted into phonetic code strings. Then tokenization is carried out for trigram retrieval, the tokenization is using trigram because, in Arabic text indexing, the trigram is the best number of grams [27]. The results are trigram indices with a trigram index is in the form of an inverted index. Then, the trigram match of the query is compared to the trigram on the index. This process is done only once to form the index. Furthermore, the application system processing is carried out every time a query is entered into the system. Overall, making a dataset with the process of creating an application system is almost similar, but there are additional processes in the application system. In the application system, a method for document ranking is added after trigram matching is carried out if the search results are found, the system will display the Quranic verse. However, suppose the results are not found, in that case, the next process will be carried out to complete the missing index trigram. After getting the candidate suggestions from the autocomplete, the similarity value is measured using the Damerau Levenshtein distance results of the smallest similarity value that will display the reference word. The difference between this system and the previous system, namely in the previous system Lafzi and Lafzi+, in the document ranking is made twice using the Longest Contiguous Subsequence, however in the Lafzi++ system that we created it is sufficient to summarize just one ranking process by using solely the Longest Common Subsequence. We also used the autocomplete method to complement the missing trigrams by adding an option to resolve the transposition typo. To calculate the edit distance for each candidate query, we use the Damerau method because it can calculate the distance for the typo substitution, insertion, deletion, and transposition.

http://doi.org/10.26594/register.v6i2.2065

ISSN 2502-3357 (Online)
N. I. Purwita et al./Register 6 (2) 2020 pp. 130-140 ISSN 2503-0477 (Print)

3.2. Dataset

The dataset used for this study was in the form of a corpus of the Koran converted into Latin letters. The records are taken from tanzil, for which each row of data contains one verse. This Latin Quran is converted into phonetic code and trigram tokenization is performed, after given tokenization, it is indexed by the formation of an inverted index.

3.3. Preprocessing

Phonetic coding is a preprocessing stage for the application that employs rules tailored based on the recitation, which is adjusted to the pronunciation in Indonesia. This procedure is applied to the Quran corpus with Arabic-Latin literacy and also query input in the form of Latin text, the result of this preprocess is in the form of a phonetic code string. Examples of applying this phonetic coding procedure are listed in Table 4 with the query "*adlin mingkum hadyan baaligha alka'bati au kaffaaratun*".

Table 4. Example phonetic coding of a query

Query: ***adlin mingkum hadyan baaligha alka'bati au kaffaaratun***		
Step	**Results**	**Information**
1	*adlin mingkum hadyan baaligha alka'bati au kaffaaratun*	vocal substitution becomes a, i, or u
2	*adlin mingkum hadyan baaligha alka'bati au kafaaratun*	merging the same consonant
3	*adlin mingkum hadyan baligha alka'bati au kafaratun*	merging the same vowels
4	*adlin mingkum hadyan baligha alka'bati aw kafaratun*	diphthong substitution, AI becomes AY and AU becomes AW
5	*'adlin mingkum hadyan baligha 'alka'bati 'aw kafaratun*	labeling hamza
6	*'adlin minkum hadyan baligha 'alka'bati 'aw kafaratun*	substitution for ikhfa reading, NG sound replaced N
7	*'adlin minkum hadyam baligha 'alka'bati 'aw kafaratun*	substitution for iqlab reading from NB to MB
8	*'adli minkum hadyam baligha 'alka'bati 'aw kafaratun*	eliminate the letter N if it meets consonants Y, N, M, W, L, R
9	*xadli minkum hadyam baliga xalkaxbati xaw kafaratun*	matching letters to phonetic code
10	*XADLIMINKUMHADYAMBALIGAXALKAXBATIX AWKAFARATUN*	space removal

The first stage in Table 4 for phonetic coding is changing the vowel "*O*" to "*A*", the vowel "*E*" becomes "*I*", and for the vowels, "*A*", "*I*", and "*U*" are remaining the same. The next step is if there are consonants that are the same and are side by side, then combined into one or in other words, one of them is deleted. The third step is if there are same and adjacent vowels, then, the vowels are combined into one. Next, by changing diphthong from "*AI*" to "*AY*" and "*AU*" to "*AW*". Then, in labelling hamza, the hamza is marked by the letters "*A, I, U*" at the beginning of words or after spaces with a single quotes. Furthermore, in the substitution for ikhfa pronunciation, removes the letter "*G*" in reading the ikhfa "*NG*". The seventh step is by substituting the iqlab pronunciation by changing the letter "*NB*" to "*MB*". The eighth step is substituting the idgham pronunciation by removing the consonant "*N*" if it meets the consonant "*Y, N, M, W, L, R*". In the ninth stage, matching letters from "*GH*" to "*G*" and apostrophes become "*X*". Then the last step is to delete spaces.

3.4. Tokenization

In this tokenization process, one type of N-Gram character is trigram, because it is capable of indexing well. In the results of the phonetic query code and the corpus Quran, tokenization was carried out to take the trigram. Queries can be substrings, so the trigram does not need a start or end marker. The tokenization process uses an overlapping window of three characters. Tokenization function in retrieving phonetic code text separated into several sequential tokens. Trigram tokenization results from the query "*MALIKINAS*" is ["*MAL*","*ALI*","*LIK*","*IKI*","*KIN*","*INA*","*NAS*"] with a total of seven trigrams.

Table 5. Inverted index

ID	**Term**	**Posting List**
1085	*MAL*	{"4":[1],"14":[5], "33":[60,109],...}
102	*ALI*	{"4":[2],"7":[55],"8":[2],...}
1025	*LIK*	{"4":[3],"9":[3],"11":[53],...}
759	*IKI*	{"4":[4],"26":[87],"30":[101],...}
941	*KIN*	{"9":[36],"11":[75],"30":[102],...}
804	*INA*	{"6":[5],"7":[10],"10":[6],...}
1183	*NAS*	{"5":[22],"6":[6],"10":[34],...}

N. I. Purwita et al./Register 6 (2) 2020 pp. 130-140
ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

3.5. Indexing trigram

After the conversion process into phonetic code and trigram tokenization has been carried out, in the corpus of the Koran an inverted index is formed. Inverted index storage file consists of document id, term, and post list. The term contains a trigram while the posting list contains a letter index and the position where the trigram appears. The indexing process is done by two methods, namely indexing with vocal and non-vocal indexing.

Table 5 is an example of an inverted index contained in the database, with the example string "*MALIKINAS*". In this string, the trigram tokenization is performed "*MAL, ALI, LIK, IKI, KIN, INA, NAS*". Then look for the inverted index of each token in the database, as in Table 5. For example, the term "*MAL*" is in the order of vowel indices to "1085" with the position appearing on "4: [1]" meaning Term "*MAL*" is found in the 4th paragraph id with the trigram position in verse being first.

3.6. Trigram matching

The trigram matching process is done by comparing the results of the trigrams on query input with trigrams that have been indexed in the Quran. Trigrams are calculated from the same document as the trigrams on query input, the calculations can be seen from the information contained in the index. The results will be used as output candidates.

3.7. Document ranking

Document ranking is calculated based on the matching number of trigrams with documents and queries. Then the position of the term in the document is calculated. The term in question is obtained from the trigram index. So that more and more trigrams are the same and ordered, the denser the position of the term will have a higher score, compared to the trigram that appears a lot of misalignment, randomized, and separate positions. To find the candidate results, use the Longest Common Subsequence (LCS) method. This method is applied for assigning a sequence score to the position index of the trigram appearance. The index which has the longest order will be used as a candidate. The following is the definition of recursive LCS.

$$\mathrm{LCS}(x_{i-1}, y_{j-1}) = \begin{cases} 0 & if\ i = 0\ or\ j = 0 \\ 1 + LCS(x_{i-1}, y_{j-1}) & if\ x_i = y_j \\ \max\left(LCS(x_i, y_{j-1}), LCS(x_{i-1}, y_j)\right) & if x_i \neq y_j \end{cases} \quad (2)$$

Equation 2 can be explained as follows. If comparing the length of the second sequence is equal to 0, the value 0 is returned. If the character at position *i* is the same as the character at position *j* then to fill $(\boldsymbol{i}, \boldsymbol{j})$ is currently added +1. If the position of the character $\boldsymbol{x_i}$ is not the same as the position of the character $\boldsymbol{y_j}$, then the maximum value of $(\boldsymbol{x_i}, \boldsymbol{y_j - 1})$ and $\boldsymbol{x_i - 1}, \boldsymbol{y_j}$ is taken.

3.8. Autocomplete trigram

To predict words for input queries with typographical errors such as insertion, deletion, substitution, and transposition, autocomplete is used to correct incorrect queries. An incorrect query results in an incomplete trigram index. The algorithm mechanism is that the user enters a query, then the system processes the trigram and ranks the document. If there is a candidate document that contains an imperfect trigram index sequence, it will be processed at this stage with the algorithm options that have been made, among others: addition to the left and right and middle of the trigram index that are not suitable, addition to the middle and left, addition to the middle and right, addition in the middle, subtraction on the left with the addition of the middle part, subtraction on the right with addition in the middle, subtraction on the right and left with an addition in the middle, subtraction on the left and addition in the middle and right, subtraction on the right as well as adding to the middle and left, reduce the middle, reduce the middle and right, reduce the middle and left, reduce the middle, right, left. The number of trigrams of input will be different from the number of trigrams in the candidate, so calculate the difference between trigrams of input minus the trigrams of candidates. Completing the index on the addition or subtraction on the right and left is done in stages. Retrieval of the trigram to complete the missing trigram uses the forward index used in the algorithm option. The trigram index result that has been completed will be converted into a fixed query and used as a corrected candidate query and then stored in a variable.

3.9. Edit distance ranking

ISSN 2502-3357 (Online)
N. I. Purwita et al./Register 6 (2) 2020 pp. 130-140 ISSN 2503-0477 (Print)

The ranking of metric distance uses the Damerau Levenshtein Distance method to calculate each suggestion candidate in handling typographical errors. This candidate is obtained from the trigram autocomplete process. Suggestion candidates that are autocomplete are placed on a list. Each candidate is compared to the original query. The candidate is calculated using edit distance in Equation 1. Then the candidate that has the smallest edit distance value will be used as a suggestion for the typo query input. The calculation of edit distance on substitution errors is different from other errors. In substitution, errors are calculated by the distance using the weighting of each character. This method can make it easier to correct spelling caused by letter-spacing on the QWERTY keyboard. It should be underlined, the weighting only uses the letters a-z characters, and the application is only on the QWERTY keyboard as shown in Figure 2, because Android users in Indonesia use a QWERTY keyboard layout. Figure 2 illustrates the layout of distance measurements for weighting. Weighting is given to the letters with the closest neighborhood to the keyboard, between two letters bound by a knot. For example, the letter "*A*" has a proximity to the letters "*Q, W, S, Z*" because the letter "A" is tied vertically with the letter. The letter "*A*" is not next to the letter "*D*" because a node does not bound it. This method is only used to tolerate substitution errors because these errors often occur because of the switch in pressing characters on the keyboard. For errors in deletion, insertion, and transposition the best way is to use unweighted with a weight of 1 because this error is not affected by placement.

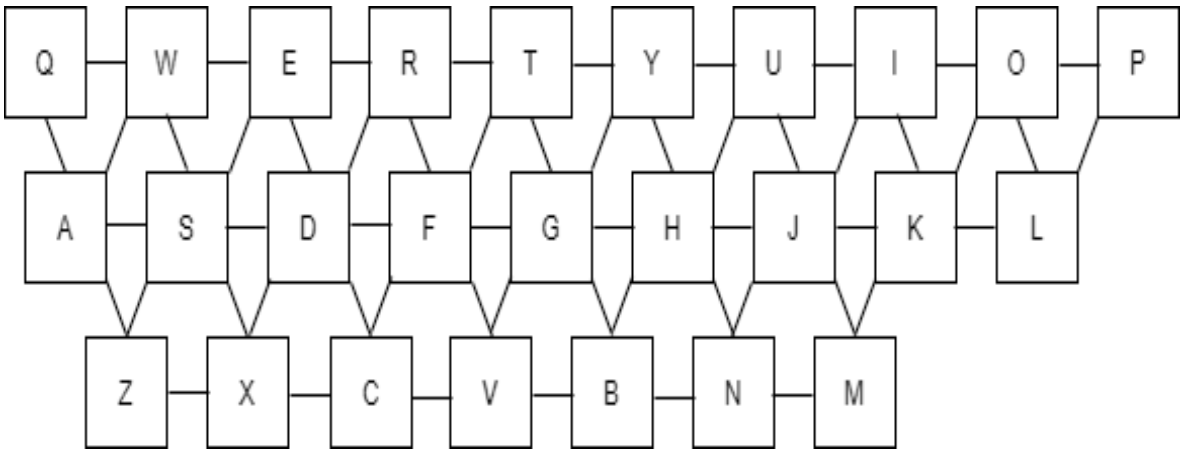

Figure 2. The adjacency map of a qwerty keyboard [28]

3.10. Calculation accuracy

The calculation of accuracy is done by comparing the verse results in the system with the gold standard paragraph. The gold standard here is the existence of a verse in the Quran. Testing is done by entering the test data query made in this study, then comparing the output results from the system with the gold standard, finally, evaluate using Recall and Mean Average Precision (MAP). To calculate the recall value, the following equation is used:

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

Equation 3 has a description where TP is the amount of output issued by the gold standard, while FN is the amount of the gold standard that is not issued by the system. Meanwhile, MAP is the average number of AP values. AP calculates every relevant document taken and takes into account the position of the relevant document in the results. Suppose the AP score is obtained as follows $\frac{\frac{1}{5}+\frac{2}{6}}{2} = \mathbf{0.266666}$. Where $\frac{1}{5}$ is the first output with the same position with the 5th order, and $\frac{2}{6}$ is the second output with the same position, and 6 is the sixth order precision. While the divider value is two because the numerator is 2. If the result is in the gold standard and system but has a different order, then it does not need to be included in the calculation.

4. Result and Discussion

System testing is done by adding 200 new data test, with details of 50 data test queries for substitution typo, 50 typo insertion queries, 50 queries for typo deletion testing, and 50 queries for transposition testing. These queries are taken from randomized Quranic fragments. For the data test testing scheme, it is divided into 4 types, the first is to test the system in dealing with error replacing characters, the

N. I. Purwita et al./Register 6 (2) 2020 pp. 130-140
ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

substitution query is used. The second is to test for errors by adding characters, then testing using the typo insertion query. Third test with some missing characters, using the typo deletion query. Finally, for typos with the characters that are exchanged, it is tested using the typo transposition query. This testing method is carried out to determine the level of accuracy of the system in handling typos in the four types of the typo, so we tested it by calculating Recall and MAP. Recall serves to determine the success of the system in achieving the gold standard, while precision is used to determine the accuracy of documents in retrieving relevant information.

Table 6. Example of queries and output suggestion

Type of Typos	Query User	Suggestion	Appearance on the system
Insertion	*ILAYAKUMROSULA*	*'ILAYKUMRASULA*	{73:15, 49:7, 3:101}
Deletion	*ILAKUMROSULA*	*'ILAYKUMRASULA*	{73:15, 49:7, 3:101}
Substitution	*ILAYKUMROSUKA*	*'ILAYKUMRASULA*	{73:15, 49:7, 3:101}
Transposition	*IALYKUMROSULA*	*'ILAYKUMRASULA*	{73:15, 49:7, 3:101}

Table 6 shows an example of a test query. Table 6 contains columns for typo name names, user input queries, suggestions, and the whereabouts of the verses on the system being made. For example, for the type of typo transposition with the input query "*IALYAKUMROSULA*" The system displays the suggestion "*'ILAYKUMRASULA*" where the piece of verse is found in Chapter 73 verse 15, Chapter 49 verse 7, and Letter 3 verse 101.

Test results on typo handling research for searching Quran verses based on phonetic similarity are shown using Recall and Mean Average Precision (MAP). The tests use a program to calculate the value of Recall and MAP on the system and also use Microsoft Excel to display the test result data. This test aims to measure the performance of the system being developed. Before using the Damerau Levenshtein method, we tried to use the Jaro Winkler method with autocomplete, but the results obtained by the Recall and MAP values in each test were far below the Lafzi and Lafzi+ systems. In the Jaro Winkler method, the highest Recall value is only 26 %, and the highest MAP is only 24 %. There are many inappropriate suggestions in using this method, and even many systems cannot issue suggestions. Because this method is problematic in this study, then we tried to use another method, namely Damerau Levenshtein distance with autocomplete because, in this method, it can calculate the distance with typo substitution, insertion, deletion, and transposition, this method can also give distance weight. The Damerau method can give better-corrected query results than Jaro–Winkler and can provide a suggested query on any incorrect input query. The Recall and MAP values in testing with the Damerau method can be seen in Table 7. System testing for normal queries is a system that is tested in the form of verse pieces that conform to Arabic characters, for the normal query test results get an accuracy of 77.60% recall and 16.21% MAP. In normal query testing, it only gets a Recall value of 77.60% because the system does not issue all the results according to the gold standard. For MAP, it only gets 16.21% because the issued results are having an unordered order. The transposition test in this study obtained a recall result of 80.03% and a MAP of 75.72%. These results are better because the previous system had not been able to handle the search for transposition optimally, so N/A was written in the table. The test results in Table 7 show that each development carried out obtained higher accuracy than the Lafzi and Lafzi+ system.

Table 7. Test results

Query Type	Lafzi		Lafzi+		Lafzi++	
	Recall	MAP	Recall	MAP	Recall	MAP
Normal Query	77.60%	16.21%	77.60%	16.21%	77.60%	16.21%
Subtitution Typo Queries	30.21%	19.67%	79.63%	74.67%	80.77%	79.34%
Insertion Typo Queries	63.70%	60.33%	90.87%	86.26%	96.20%	89.59%
Deletion Typo Queries	58.92%	63.75%	89.68%	87.76%	94.82%	90.69%
Transposition Typo Queries	N/A	N/A	N/A	N/A	80.03%	75.72%

This is evident from the development of Lafzi+ which obtains better accuracy than the previous system, the highest recall value is 90.87% and for MAP is 87.76% in testing for insertion and deletion queries. Then developed again in this study, namely Lafzi++, by obtaining a higher Recall value on each type of test query type compared with the previous system. Then it was further developed in this study, namely Lafzi++, by obtaining a higher recall value in each type of typo query test compared to the

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

previous system. This happened because in the Lafzi+ system when testing the data, several verses did not match the correction query, because in the Lafzi+ system, in finding the query correction it produces an incomplete trigram so that the resulting suggestion query predictions are not suitable. By providing improvements to the autocomplete process in finding trigrams and adding algorithm options for transposition, the system can provide correct suggestions and can increase the Recall and MAP values. Obtaining the highest accuracy, the recall reached 96.20% using the insertion query and MAP of 90.69% based on the removal testing query.

5. Conclusions

Based on the test results obtained in the research on Typo Handling for Al-Qu'ran Verse Search Based on Phonetic Similarities, this system is proven to be able to overcome typing errors better than the previous system development using autocomplete and Damerau Levenshtein distance. Combining these two methods is able to handle typing errors in the query namely typo substitution, deletion, insertion, and transposition. The autocomplete used is able to improve the missing trigram index so that it can provide good query suggestions candidates and use Damerau to calculate the distance from each suggestion candidate obtained from the autocomplete, so as to be able to generate appropriate suggestion queries. Judging from the results of accuracy, all Recall and MAP values in Lafzi++ can outperform the Lafzi system, and Lafzi+, with typo substitution obtaining Recall 80.77% and 79.34%, for insert queries it produces the highest recall among others at 96.20% and MAP 89. 59%. The highest MAP was obtained from deletion typo queries of 90.69% and recall of 94.82%, and transposition queries reached Recall 80.03% and MAP 75.72%. Suggestions for further development, the system is able to implement auto search suggestions such as the search feature on Google, to overcome verse searches by users who only remember partial verse fragments.

6. Acknowledgement

We would like to express our gratitude to Telkom University for funding this research.

7. References

[1] M. M. Hamzah, "Peran dan Pengaruh Fatwa Mui dalam Arus Transformasi Sosial Budaya di Indonesia," *Millah: Jurnal Studi Agama,* vol. 12, no. 1, pp. 127-154, 2017.

[2] I. Humaini, T. Yusnitasari, L. Wulandari, D. Ikasari and H. Dutt, "Informatian Retrieval of Indonesian Translated version of Al Quran and Hadith Bukhori Muslim," in *International Conference on Sustainable Energy, Electronics, and Computing Systems (SEEMS)*, Greater Noida, India, 2018, 2018.

[3] J. .. Pardeshi and B. Nandwalkar, "Survey on: Rule Based Phonetic Search for Slavic Surnames," *Int.J.Computer Technology & Applications,* vol. 7, no. 1, pp. 65-68, 2016.

[4] M. A. Istiadi, "Sistem Pencarian Ayat Al-Quran Berbasis Kemiripan Fonetis," Institut Pertanian Bogor, Bogor, 2012.

[5] W. Satriady, M. A. Bijaksana and K. M. Lhaksmana, "Quranic Latin Query Correction as a Search Suggestion," *Procedia Computer Science,* vol. 157, pp. 183-190, 2019.

[6] V. C. Mawardi, R. Rudy and D. S. Naga, "Fast and Accurate Spelling Correction Using Trie and Damerau-Levenshtein Distance Bigram," *TELKOMNIKA,* vol. 16, no. 2, pp. 827-833, 2018.

[7] T. N. Maghfira, I. Cholissodin and A. W. Widodo, "Deteksi Kesalahan Ejaan dan Penentuan Rekomendasi Koreksi Kata yang Tepat Pada Dokumen Jurnal JTIIK Menggunakan Dictionary Lookup dan Damerau-Levenshtein Distance," *Jurnal Pengembangan Teknlogi Informasi dan Ilmu Komputer (J-PTIIK),* vol. 1 , no. 6, pp. 498-506, 2017.

[8] G. R. Bunt, iMuslims: Rewiring the House of Islam, Chapel Hill, North Carolina, United States: University of North Carolina Press, 2009.

[9] J.-F. Yeh, L.-T. Chang, C.-Y. Liu and T.-W. Hsu, "Chinese Spelling Check based on N-gram and String Matching Algorithm," in *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications*, Taipei, Taiwan, 2017.

ISSN 2502-3357 (Online)
ISSN 2503-0477 (Print)

[10] M. M. Hossain, M. F. Labib, A. S. Rifat, A. K. Das and M. Mukta, "Auto-correction of English to Bengali Transliteration System using Levenshtein Distance," in *7th International Conference on Smart Computing & Communications (ICSCC)*, Sarawak, Malaysia, 2019.

[11] K. Balabaeva, A. Funkner and S. Kovalchuk, "Automated Spelling Correction. for Clinical Text Mining in Russian," in *Medical Informatic Europe Conference Conference*, 2020.

[12] S. J. Putra, M. N. Gunawan and A. Suryatno, "Tokenization and N-Gram for Indexing Indonesian Translation of the Quran," in *6th International Conference on Information and Communication Technology (ICoICT)*, Bandung, 2018.

[13] B. C. Gencosman, H. C. Ozmutlu and S. Ozmutlu, "Character n-gram application for automatic new topic identification," *Information Processing & Management,* vol. 50, no. 6, pp. 821-856, 2014.

[14] K. Srinivasa and B. N. S. Devi, "GPU Based N-Gram String Matching Algorithm with Score Table Approach for String Searching in Many Documents," *Journal of The Institution of Engineers (India): Series B* , vol. 98, p. 467–476, 2017.

[15] P. Náther, "N-gram based Text Categorization," Comenius University, Bratislava, Slovakia, 2005.

[16] N. Nizamkari, "Mining typos in text," in *IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, New York, 2016.

[17] S. Thaiprayoon, A. Kongthon and C. Haruechaiyasak, "ThaiQCor 2.0: Thai Query Correction via Soundex and Word Approximation," in *5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*, Krabi, 2018.

[18] M. Castelli, R. Dondi, G. Mauri and I. Zoppis, "Comparing incomplete sequences via longest common subsequence," *Theoretical Computer Science,* vol. 796, pp. 272-285, 2019.

[19] R. Khan, M. Ahmad and M. Zakarya, "Longest Common Subsequence Based Algorithm for Measuring Similarity Between Time Series: A New Approach," *World Applied Sciences Journal*, vol. 24, no. 9, pp. 1192-1198, 2013.

[20] G. Kawade, S. Sahu, S. Upadhye, N. Korde and M. Motghare, "An analysis on computation of longest common subsequence algorithm," in *International Conference on Intelligent Sustainable Systems (ICISS)*, Palladam, 2017.

[21] C. Blum and M. J. Blesa, "Hybrid techniques based on solving reduced problem instances for a longest common subsequence problem," *Applied Soft Computing,* vol. 62, pp. 15-28, 2018.

[22] M. R. Islam, C. M. K. Saifullah, Z. T. Asha and R. Ahamed, "Chemical reaction optimization for solving longest common subsequence problem for multiple string," *Soft Comput,* vol. 23, p. 5485–5509, 2019.

[23] R. Gabrys, E. Yaakobi and O. Milenkovic, "Codes in the Damerau Distance for Deletion and Adjacent Transposition Correction," *IEEE Transactions on Information Theory,* vol. 64, no. 4, pp. 2550-2570, 2018.

[24] C. Zhao and S. Sahni, "Efficient computation of the Damerau-Levenshtein distance between biological sequences," in *IEEE 7th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)*, Orlando, FL, 2017.

[25] J. Kysela, "A Comparison of Text String Similarity Algorithms for POI Name Harmonisation," in *Lecture Notes in Computer Science*, Cham, Springer, 2018.

[26] A. Anton, "Romanian Biometric Word List for Public Key Fingerprint Validation," in *IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, Timisoara, 2018.

[27] A. F. A. Nwesri, "Effective Retrieval Techniques for Arabic Text," RMIT University, Melbourne, Victoria, Australia, 2008.

[28] A. Samuelsson, "Weighting Edit Distance to Improve Spelling Correction in Music Entity Search," KTH Royal Institute of Technology, Stockholm, 2017.