JOURNAL OF APPLIED
COMPUTER SCIENCE
Vol. 20 No. 2 (2012), pp. 7-26

Effective Similarity Measures in Electronic Testing at Programming Languages

Adio Akinwale, Adam Niewiadomski

Lodz University of Technology
Institute of Information Technology
ul. Wólczańska 215, 90-924 Łódź, Poland
adio.taofiki.akinwale@guest.p.lodz.pl
adam.niewiadomski@p.lodz.pl

Abstract. *The purpose of this study is to explore the grammatical properties and features of generalized n-gram matching technique in electronic test at programming languages. N-gram matching technique has been successfully employed in information handling and decision support system dealing with texts but its side effect is size n which tends to be rather large. Two new methods of odd gram and sumsquare gram have been proposed for the improvement of generalized n-gram matching together with the modification of existing methods. While generalized n-grams matching is easy to generate and manage, they do require quadratic time and space complexity and are therefore ill-suited to the proposed and modified methods which work in quadratic in nature. Experiments have been conducted with the two new methods and modified ones using real life programming code assignments as pattern and text matches and the derived results were compared with the existing methods which are among the best in practice. The results obtained experimentally are very positive and suggested that the proposed methods can be successfully applied in electronic test at programming languages.*

Keywords: *similarity and distance measures, fuzzy relations, n-gram, programming codes.*

1. Introduction

Programming language is a problem-solving oriented and very practical process in nature whereas its assessment requires a big challenge for the purpose of uniformity and consistency. Many recent virtual learning environment systems do not provide any electronic test for the verification of the student's source codes submitted for grading. The traditional assessment is not easy to be adopted to various new developments in higher levels of programming language from structural programming to object oriented programming languages. Assessment of students' source codes under examination condition is notoriously difficult and therefore needs the attention of electronic test. Due to the problem of assessment of programming codes, many students' assignments in programming languages are always plagiarized which in one way decreases learning outcomes.
Recently many higher institutions of learning have started off to switch from traditional paper and pencil examinations into electronic test due to intranet and internet technologies. Electronic test is supposed to reduce cost and improve quality as well as fast processing of examinations. There are number of tools used by institutions to facilitate automatic grading of students' programming language assignments but these tools fail to take into account the way in which a problem has been solved [1]. This at times leads to inequitable grading results because a program producing a right output may not meet the programming specifications. The tools only work with fill in the gap type programming assessments. Electronic test at programming languages has been of interest to computer science tutors in grading students' assignments and examinations. Till now, there is no established consensus on the best way to evaluate students' assignments objectively using electronic test. An existing similarity functions have achieved some levels of practical success in solving problem associating with natural language processing, but they have generally not been assessed in term of their ability to model human judgment of programming languages. As a result, a new similarity measure of N-grams is proposed at programming language to rank students' codes into step-wise grade categories based on their programming capabilities. The methods will give reasonable marks to partly correct matching programming codes in the semantic analysis.

2. Literature reviews

Essentially, the N-grams model is a probabilistic model originally devised by the Russian mathematician, Andrey Markov in the early 20th century and later ex-

tensively experimented by Shannon and Chomsky for predicting the next item in a sequence of items [2]. N-grams have been successfully used for a long time in a wide variety of problems and domains such as text compression [3], spelling error detection and correction [4], optical character recognition, information retrieval [5], automatic text categorization [6], music representation, speech and handwriting recognition [7]. Other useful domains include computational immunology, analysis of whole-genome protein sequences [8], language identification, authorship attribution, phylogenetic tree reconstruction., data integration , filtering and cleaning, prediction of English language [9], phonetic matching algorithms, and text retrieval [10].

A typical example, acceleration of general string searching has been accomplished using n-gram signatures by Harrison in 1971 [11]. In 1995, Damarshek pursued n-grams as a method of comparing documents because the n-gram spectrum offers a uniform framework for fast compression and is also easy to compute. He used n-gram of length 5 and 6 for clustering of text by language and topic. He applied n = 5 for English Language and n = 6 for Japanese Language. He also used n-gram sliding windows approach for categorizing text in a completely unrestricted multilingual environment. He proposed a simple but novel vector-space technique that makes sorting, clustering and retrieval feasible in a large multilingual collection of documents [12]. Pearce and Nicholas followed Damashek work by using 5-gram to support a dynamic hypertext system in 1995 [13]. In the same year, a more sophisticated n-gram weighting scheme based on the G^2 statistics has been used by Cohen. The method highlighted words by statistical characteristic of the n-grams of which they are composed [14]. Yannakoudakis et. al. used n-gram that cross word boundaries that is, start with one word and end in another word and include the space character that separates consecutive words [15]. Church and Gale proposed smoothing techniques to solve the problematic of zero-frequency of n-gram that never occurred in a corpus [16]. Kuhn and De Moris suggested the weighted n-gram model which precisely approximates the n-grams length based on their position in the context [17]. Niesel and Woodland used the variable length n-gram model which changes the n size of n-grams depending on the text being manipulated so that better overall system accuracy is achieved[18]. Assale et. al. addressed an n-gram based signature method to detect computer viruses [19]. Niewiadomski used generalized n-gram matching for automatic evaluating text examination and used the method also to evaluate electronic language test using German language as a case study [20].

3. Fuzzy binary relations and similarity as a relation

Given a set U an ordinary subset A of U can be defined in terms of its characteristic function $X_A(x)$ (that return 1 if $x \in X$ or 0 otherwise). On the other hand, a fuzzy subset A of U is a function $A : U \longrightarrow [0, 1]$. The function is called the membership function and the value *A(x)* represents the degree of membership of x in the fuzzy subset A, being a generalization of the notion of characteristic function. Similarly, an ordinary binary relation on U is a subset of $U \times U$ and it can be identified by its characteristic function $U \times U \longrightarrow (0, 1)$. Therefore, the easy extension of this concept to the fuzzy case is to agree that, a fuzzy binary relation R is a fuzzy subset on $U \times U$ (that is a mapping $R : U \times U \longrightarrow [0, 1]$). So given two elements u_i and u_j in R, $R(u_i, u_j) = \alpha_i j$ represents the degree to which the pair $\langle u_i, u_j \rangle$ is compatible with the relation R.

Definition: A similarity relation on a set U is a fuzzy binary relation R : $U \times U \longrightarrow [0, 1]$ holding the following properties:

$$Reflexive \qquad R(x, x) = 1 \quad for\, any\, x \in U \tag{1}$$

$$Symmetric \qquad R(x, y) = R(y, x) \quad for\, any\, x, y \in U \tag{2}$$

$$Transitive \qquad R(x, z) \geq R(x, y) \triangle R(y, z) \quad for\, any\, x, y, z \in U \tag{3}$$

where the operator $\triangle$ is an arbitrary t-norm. A t-norm $\triangle : [0, 1] \times [0, 1] \longrightarrow [0, 1]$ is a binary operator which is commutative, associative, monotone in both arguments and $1 \triangle x = x$ hence it subsumes the classical two-valued conjunction operator. Sometimes, transitivity is qualified by an specific t-norm $\triangle$ and it is called $\triangle - transitivity$. When the operator $\triangle = \wedge$ (this is the minimum of the two elements), a similarity relation is called a fuzzy equivalent relation. Certainly in this case, there exits a close relation between similarity relations and equivalence relations. The so called $\vee - cut$ of a fuzzy equivalence relation R is an equivalence relation. If R is a fuzzy binary relation on U, the binary relation $R_\vee = (x, y) \mid R(x, y) \geq \vee$ is call the $\vee - cut$ of R. Since R can be considered as a generalization of the identity relation, intuitively, a fuzzy equivalence on a set specifies when two elements may be considered equal with regard to a property that is not sharply defined. In the sequel, we restrict ourselves to similarity relation that are fuzzy equivalence relations. Moreso, we are interested in fuzzy equivalence relations at a syntactic level [21].

We consider a relation of similarity x_1 and x_2 which is written as $x_1 \sim x_2$. These similarity relations are subject to reflexive and symmetry and may not be necessarily be transitive. In this case, relation R on X is called the relation of neighbourhood if R is reflexive on X and R is symmetry on X. Neighbourhood relationship is also referred as follows: non-sup-min transitive similarity relation, tolerance relation, proximity relation, partial preorder relation, resemblance relation, approximate equality relation, etc.

3.1. Set similarity

Definition: Let A, B be arbitrary sets on X. Function $\mu : X \longrightarrow R^+ \cup 0$ is a measure of sets if and only if

$$\mu(\phi) = 0 \tag{4}$$

$$\mu(A \smile B) \leq \mu(A) + \mu(B) \tag{5}$$

Equations 4 and 5 can narrow down to $\mu(A \smile B) = \mu(A) + \mu(B) - \mu(A \frown B)$. Other properties of similarity measure sets are:

$$A = B \longrightarrow \mu(A) = \mu(B) \tag{6}$$

$$A \subseteq B \longrightarrow \mu(A) \leq \mu(B) \tag{7}$$

The implication of the reverse is not necessarily to be true [20].

4. Test similarity measures

There are two ways to measure resemblance between a pair of test, namely:

- Similarity or dissimilarity measure
- Distance or metric measure

4.1. Distance or metric measure

Distance measures dissimilarity. The measurement between feature i and j is denoted by d_{ij}. Distance is a quantitative variable which in general will satisfy the following at least the first three conditions: 1. $d_{ij} = 0$ (distance is zero if and only

if it measured to itself), 2, $d_{ij} \geq 0$ (distance is always positive or zero), 3, $d_{ij} = d_{ji}$ (distance is symmetry), 4, $d_{ij} \leq d_{ik} + d_{jk}$ (distance satisfies triangular inequality). Distance is also metric if it satisfies all above four conditions. Due to the triangular inequality, it is not all the distance that is metric but all metrics are distances. Test distance measure tends to quantify the minimum cost of transforming one test to another. In general, cost weights are assigned for each substitution, insertion and deletion operations. Examples of familiar distance measures are as follows:

$$Minkowski\,distance \;=\; \sqrt[\lambda]{\sum_{i=0}^{n} \mid x_i - y_i \mid^{\lambda}} \tag{8}$$

if $\lambda = 1$ gives city block distance; $\lambda = 2$ gives Euclidean distance; $\lambda = \infty$ gives Chebychev distance

$$Chessboard\,distance(x, y) \;=\; (max_i \mid x_i - y_i \mid) \tag{9}$$

$$Canberra\,distance(u, v) \;=\; \sum \frac{\mid u_i - v_i \mid}{\mid u_i \mid + \mid v_i \mid} \tag{10}$$

$$Hamming\,distance(i, j) \;=\; \sum_{k=0}^{n-1} [y_{ik} \neq y_{jk}] \tag{11}$$

$$S\,pearman\,distance(i, j) \;=\; \sum_{i=1}^{n} (x_{ik} - x_{jk})^2 \tag{12}$$

4.2. Similarity or dissimilarity measure

Similarity measure is quantity that reflects the strength of relationship between two objects or two features. The measurement between feature i and j is denoted by s_{ij}. The relationship between dissimilarity and similarity is given by $s_{ij} = 1 - d_{ij}$. The relationship is usually having a range of either -1 to 1 or normalized into 0 to 1. When the similarity is 1 that it is exactly similar, the dissimilarity is 0. There are a variety of functions to measure the similarity between texts. Among the familiar similarity measures are as follows:

$$Jaccard\,similarity(x, y) \;=\; \frac{\mid x \frown y \mid}{\mid x \smile y \mid} \tag{13}$$

$$Cosine\ similarity(x, y) = \frac{\sum_{i}^{d} x_i y_i}{(\sqrt{\sum_{i}^{d} x^2}) \times (\sqrt{\sum_{i}^{d} y^2})} \tag{14}$$

$$Overlap\ similarity(x, y) = \mid x \frown y \mid \tag{15}$$

The above similarities and distance functions are inter-related and recent researches have been combined them to improve the performances of string processing for different applications[22]. For example, Fuzzy Jaccard Similarity measure, an extension of Jaccard Similarity measure combined token-based similarity and character-based similarity as follows :

$$FJaccard(s_1, s_2) = \frac{\mid T_1 \wedge_\zeta T_2 \mid}{\mid T_1 \mid + \mid T_2 \mid - \mid T_1 \wedge_\zeta T_2 \mid} \tag{16}$$

where T_1 and T_2 are token sets and ζ is an edit similarity threshold.
Character-based similarity in the above method used edit similarity to quantify the similarity of two strings where edit similarity between the two strings s1 and s2 is defined as

$$NED(s1, s2) = 1 - \frac{ED(s1, s2)}{max(\mid s1 \mid, \mid s2 \mid)} \tag{17}$$

In this case, edit distance has been modified.

5. Methods

There are many methods of measuring similarity between two strings as indicated in section 4. Our approach involves the use of n-gram where the similarity between two strings is assessed by comparing their respective n-grams. The greater the number of n-gram they have in common, the greater is their similarity. Another approach is by counting the different between corresponding numbers of n-gram occurrence. For example, Alberga in 1967 measured similarity between strings based on the proportion of removing the longest common substring from the pairs [23].

5.1. N-gram method

Definition: Let $A = (a_1, a_2, a_3, \ldots, a_n)$ be a sequence, where $a_i \in \sum(i = 1, 2, 3, \ldots, k)$ then $(a_{j+1}, a_{j+2}, \ldots, a_{j+n}) \in \sum^n$ is called a n-gram of the sequence

, where $0 \leq j \leq k - n$: the set of all the n-grams of sequence is called the n-gram set of sequence , that is $G(A, n) = (a_{j+1}, a_{j+2}, \ldots, a_{j+n}) \mid 0 \leq j \leq k - n$ is the n-gram set of sequence where $n \in Z^+$ is the length of the n-gram. It is noted that $G(A, n) = \phi$ if $k < n$.

5.2. Dice method

In evaluating one term against another term, Dice similarity is chosen because it is popular and widely used in analogous text of retrieval systems. This measure takes into account the length of terms. The coefficient values varies between zero and one. If two terms have no characters in common then the coefficient value is zero. On the other hand, if they are identical, the coefficient value will be one [24]. For two string x and y, the Dice coefficient is measured as

$$d(X, Y) = \frac{2(n - gram(X \frown Y))}{n - gram(X) + (n - gram(Y))} \tag{18}$$

5.3. Generalized n-gram matching

Generalized n-gram matching was introduced by Niewiadomski. The algorithm matches an answer string to a template string. The matched strings are denoted as s_1, s_2 and $N(s_1) = N(s_2) = N$ is the length of the string. Hence,

$$sim(s_1, s_2) = f(n_1, n_2) \sum_{i=n_1}^{n_2} \sum_{j=1}^{N-n+1} h(i, j) \tag{19}$$

where $f(n_1, n_2) = \frac{2}{(N-n_1+1)(N-n_2+2)-(N-n_2+1)(N-n_1)}$ denotes the number of possible substrings not shorter than n_1 and not longer than n_2 in $s_1, h(i, j) = 1$ iff an i-element-long substring of the string s_1 starting from j-th position in s_1 appears (at least) once in s_2 (otherwise $h(i, j) = 0$). If all substrings from one argument of comparison are found in the other, the final similarity degree is evaluated as 1 which is interpreted as the identity of s_1 and s_2 [20]

5.4. Modification of existing methods

Bigram and Trigram were modified as Bi-n-gram and Tri-n-gram where n stands for two and three letters as against one letter for each statement line or

function in programming codes. For example, a four statement lines of A, B, C , D would become for Bi-n-gram AB, CD, EF, and GH while for Tri-n-gram are ABC, DEF, GHI, and JKL. Let assuming that instead of ABCD, student answer is ACBD, Bigram would award zero grading whereas modified methods will award grading for the two matches which are similar to human judgment. In this case the edit distance is very low with 2 operations and intuitively, the similarity should be high. This necessitates for the modification of bigram and trigram into bi-n-gram and tri-n-gram which increases letter of each statement line by line and allowing more string sharing.

5.4.1. Bigram into Bi-n-gram

Generalized n-gram matching is normally used to derive bi-n-gram where n represents two letters for each statement in programming codes. The formula is as follows:

$$sim(s_1, s_2) = \frac{1}{N-n+1} \sum_{i=0}^{N-n+1} h(i) = \frac{1}{N-2+1} \sum_{i=0}^{N-2+1} h(i) = \frac{1}{N-1} \sum_{i=0}^{N-1} h(i) \tag{20}$$

5.4.2. Trigram into Tri-n-gram

Tri-n-gram is also derived from generalized n-gram matching as follows where n represents three letters for each statement in programming codes.

$$sim(s_1, s_2) = \frac{1}{N-n+1} \sum_{i=0}^{N-n+1} h(i) = \frac{1}{N-3+1} \sum_{i=0}^{N-3+1} h(i) = \frac{1}{N-2} \sum_{i=0}^{N-2} h(i) \tag{21}$$

5.5. New methods

5.5.1. Odd gram

Odd-gram was inspired by the generalized n-gram matching which takes n(n-1)/2 substrings for processing before measuring the performance. The odd gram would take half substrings of generalized n-gram matching for processing the performance which would still reduce the running time. For the method, the matched

strings are denoted as s_1, s_2 and $max(N(s_1), N(s_2)) = N$ which is the maximum length between string s_1 and s_2. If N is odd then $N = \lceil \frac{N}{2} \rceil$

$$sim(s_1, s_2) = \frac{1}{N^2} \sum_{i=N}^{N} \sum_{j=1}^{N-i+1} h(i, j) \qquad else \qquad \frac{1}{N^2 + N} \sum_{i=N}^{N} \sum_{j=1}^{N-i+1} h(i, j) \tag{22}$$

5.5.2. sumSquare-gram

Likewise Odd-gram, sumSquare gram was inspired by the generalized n-gram matching which processing time is quadratic for every N-gram in the query string with every N-gram in a line. While similarity measures of N-gram are easy to generate and manage, they do require quadratic time and space complexity and therefore ill-suited to both odd-gram in section 5.5.1 and sumSquare gram which work in quadratic. Odd-gram and sumSquare gram methods are expected to write their results into similarity measure (s) between a pair of submissions (pattern matching and text matching). Given pattern matching and text matching i and j, s_{ij} will be near to 1 if both patterns are considered identical and near to 0 if they are very dissimilar. That is, odd-gram and sumSquare grams are normalized to fall within the interval [0, 1]. Similarly, similarity measure of odd-gram and sumSquare gram are expected to be symmetric, that is the equality $s_{ij} = s_{ji}$ is expected to hold for every i, j. For the sumSquare gram, the matched strings are denoted as s_1, s_2 and $max(N(s_1), N(s_2)) = N$ which is the maximum length between string s_1 and s_2.
$N = \lfloor \sqrt{N} \rfloor$
$M = times - to - jump = N - 1$
$P = first - jump = N^2 - (N - 1)^2$

$$sim_{sq}(s_1, s_2) = \frac{6}{N(N + 1)(2N + 1)} \sum_{i=1}^{P} \sum_{j=1}^{M} h(i, j) \tag{23}$$

For an example, Consider the word ı*ALGORITHM*ȷ for string one and two. The substrings are shown in Table 1. The common substrings between the word "ALGORITHM" are used for the calculating all the six methods as follows:

1. Generalized N-gram matching
$sim(s_1, s_2) = \frac{2}{N^2+N} \sum_{i=1}^{N} \sum_{j=1}^{N-i+1} h(i, j) = \frac{2}{9^2+9} \times \frac{9+8+7+6+5+4+3+2+1}{1} = \frac{90}{90} = 1$
2. Dice's Coefficient = $d(X, Y) = \frac{2(n-gram(X \frown Y))}{n-gram(X)+(n-gram(Y))} = \frac{2(8)}{8+8} = \frac{16}{16} = 1$

```
Class Program{
    Public static main(String [] args){
     BufferedReader bf = new BufferedReader(new FileReader("C :path"));
    a   countNumberofLine( c ) {  }
    b    bf.openFile( c );
    c    readFile(String c){    }
    d    runProgram{
    e                  createSubString() {  }
    f                   initialization() {  }
    g                    double  p  = m/t;
    h                    double  m =  findDiceCoefficient(){ }
    i                   double t   = findTime() {  }
    j                   findAverage() {
    k                            printallOutput(){   }
                         }
           }
    l          findMedian(){
    m                   sorting(){
    n                           swap(a, b){
    o                                a = b;
    p                                d = a;
    q                                b = d ;
                                }
                     }
            }
    r      bf.closeFile( c );
     }
}
```

Figure 1. Sample of programming codes

Table 1. Sample of common substrings for calculating the methods

Element	Substring	Number of common substrings
1	A, L, G, O, R, I, T, H, M	9
2	AL, LG, GO, OR, RI, IT, TH, HM	8
3	ALG, LGO, GOR, ORI, RIT, ITH, THM	7
4	ALGO, LGOR, GORI, ORIT, RITH, ITHM	6
5	ALGOR, LGORI, GORIT, ORITH, RITHM	5
6	ALGORI, LGORIT, GORITH, ORITHM	4
7	ALGORIT, LGORITH, GORITHM	3
8	ALGORITH, LGORITHM	2
9	ALGORITHM	1

3. Bigram = $sim(s_1, s_2) = \frac{1}{N-n+1} \sum_{i=0}^{N-n+1} h(i) = \frac{1}{9-1} \times \frac{8}{1} = \frac{8}{8} = 1$

4. Trigram = $sim(s_1, s_2) = \frac{1}{N-n+1} \sum_{i=0}^{N-n+1} h(i) = \frac{1}{9-2} \times 71 = \frac{7}{7} = 1$

5. Odd-gram = If is odd then $N = \lceil \frac{N}{2} \rceil = N = odd = \lceil \frac{9}{2} \rceil = 5$, $sim(s_1, s_2) = \frac{1}{N^2} \sum_{i=N}^{N} \sum_{j=1}^{N-i+1} h(i, j) = \frac{1}{5^2} \times \frac{9+7+5+3+1}{1} = \frac{25}{25} = 1$

6. sumSquare gram = $N = \lfloor \sqrt{N} \rfloor = 3$, $M = timestojump = N - 1 = 2$, $P = firstjump = N^2 - (N-1)^2 = 3^2 - 2^2 = 5$, $2^2 - 1^2 = 3$, $sim_{sq}(s_1, s_2) = \frac{6}{N(N+1)(2N+1)} \sum_{i=1}^{P} \sum_{j=1}^{M} h(i, j) = \frac{6}{3(4)(7)} \times \frac{9+4+1}{1} = \frac{14}{14} = 1$

Since the pattern matching is the same as text matching, all the six methods have the same result of one.

6. Experiment

All the methods in number 18, 19, 20, 21, 22 and 23 were implemented using JAVA Programming Language. The experiment was conducted on HP Laptop with an Intel Pentium 2.10 GHz dual core CPU and 1.00 GB memory, running a 32-bit

Windows Vista operating system. A statement line or functions of programming languages are donated by unique letter which serve as input data to the system. The combination of these unique letters form string codes. Figure 1 illustrates one of the samples of code lines and their unique letters. To test the knowledge of computer science students, they are requested to study the program step by step and arrange them in sequence way the computer system will execute the codes. The sequence unique letters formed by the students represent pattern matching while the correct unique letters formed by the tutor represent text matching. In the case of the programming codes in Figure 1, the correct answer is (bcradfehigjlmnpoqk) while the first five unique letters generated by five students are as follows: 1: bcadefhigjlmnoqpkr, 2: bcadefhigrljmnopqk, 3: bacrdfehigjlmnopqk, 4: bcardefhigjlmnoqpk, 5: bcradefhigjmnolpq. These data were read by the system and generated the values of similarities, running times and performance to price for Niewiadomski generalized n-gram matching, Dice similarities, odd gram, sum-square gram, bi-n-gram and tri-n-gram of the five students' assignments as illustrated in Figure 2, 3 and 4. The performance to price (Ptp) is measured as (similarity values/running time values). The value of running time has been converted to milliseconds. For the effective measurement of each method in respect to the degree of similarity, running time and performance to price, we were able to get 302 pattern matches from 100 programming codes through students' assignments which were not the same with the correct answers. Due to the large number of the individual result of 302 pattern matches, the average and standard derivation were used. Figures 5, 6 and 7 show the total average of similarity values, running time and performance to price of the six methods.

7. Results

Looking at the Figure 2, 3 and 4, the degree of similarity values between pattern matches answered by the individual student and text matches generated by the tutor, the values of Dice and sumSquare gram method have the range of the same results (0.53, 0.53), (0.65, 0.60) as examples while the values of bi-n-gram and tri-n-gram also have the range of the same results (0.77, 0.77), (0.74, 0.73). For instance, the times of processing each individual method of Dice, odd gram and sumSquare gram are the same. Despite long string processing, tri-n-gram is the fastest in term of running time of individual method. Figure 5, 6, and 7 illustrate the total average similarity, running time and performance to price of the six

Ans. by Students		Niewiadomski	Dice	OddGram	sumSquare
bcadefhigjlmnoqpkr	Sim	0.25	0.53	0.3	0.51
Correct Answer	Time	5.67	0.65	0.93	0.66
bcradfehigjlmnpoqk	Ptp	0.04	0.82	0.32	0.78
Ans. by Students		Niewiadomski	Dice	OddGram	sumSquare
bcadefhigrljmnopqk	Sim	0.15	0.35	0.21	0.51
Correct Answer	Time	1.69	0.29	0.3	0.3
bcradfehigjlmnpoqk	Ptp	0.09	1.21	0.71	1.73
Ans. by Students		Niewiadomski	Dice	OddGram	sumSquare
bacrdfehigjlmnopqk	Sim	0.38	0.65	0.42	0.6
Correct Answer	Time	0.48	0.26	0.26	0.26
bcradfehigjlmnpoqk	Ptp	0.78	2.53	1.62	2.3
Ans. by Students		Niewiadomski	Dice	OddGram	sumSquare
bcardefhigjlmnoqpk	Sim	0.24	0.47	0.3	0.51
Correct Answer	Time	1.26	1.07	1.08	1.07
bcradfehigjlmnpoqk	Ptp	0.19	0.44	0.28	0.48
Ans. by Students		Niewiadomski	Dice	OddGram	sumSquare
bcradefhigjmnolpqk	Sim	0.21	0.53	0.27	0.51
Correct Answer	Time	1.92	1.73	1.74	1.73
bcradfehigjlmnpoqk	Ptp	0.11	0.31	0.15	0.3

Figure 2. Individual result for generalized n-gram, Dice, Odd and sumSquare methods

Ans. by Students		bi-n-gram
cdefabghijklopqrmnstwxyzABCDGHEFuvIJ	Sim	0.77
Correct Answer	Time	0.14
cdefIJabghklijopqrmnstwxyzABEFCDGHuv	Ptp	5.66
Ans. by Students		bi-n-gram
cdefabghijklopqrmnIJwxstyzABCDEFGHuv	Sim	0.71
Correct Answer	Time	0.09
cdefIJabghklijopqrmnstwxyzABEFCDGHuv	Ptp	8.01
Ans. by Students		bi-n-gram
cdabefIJghkjijopqrmnstwxyzABCDEFGHuv	Sim	0.83
Correct Answer	Time	0.13
cdefIJabghklijopqrmnstwxyzABEFCDGHuv	Ptp	6.5
Ans. by Students		bi-n-gram
cdefabIJghijklopqrmnstwxyzABCDGHEFuv	Sim	0.74
Correct Answer	Time	0.11
cdefIJabghklijopqrmnstwxyzABEFCDGHuv	Ptp	6.48
Ans. by Students		bi-n-gram
cdefIJabghijklopqrmnstyzABCDwxEFGHuv	Sim	0.77
Correct Answer	Time	0.11
cdefIJabghklijopqrmnstwxyzABEFCDGHuv	Ptp	6.93

Figure 3. Individual result for bi-n-gram

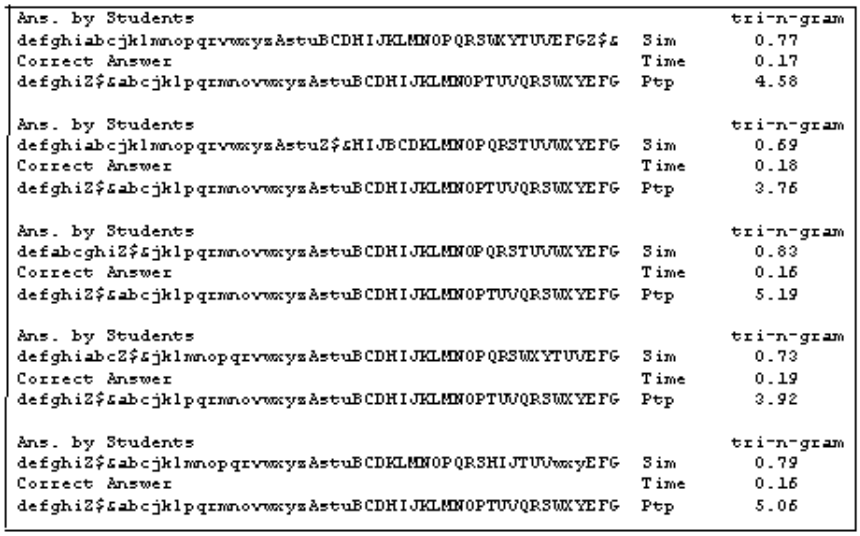

Figure 4. Individual result for tri-n-gram

methods using 302 pattern and text matches. As depicted in the Figure 5, the total average similarity of bi-n-gram is the best, followed by sumSquare, tri-n-gram, and latter by odd gram, Dice and generalized n-gram methods. The average times of processing the 302 text matches, bi-n-gram performed the best while sumSquare, Dice and odd gram have more or less the same processing times as shown in Figure 6. Among the six methods, the total average performance to price of bi-n-gram was demonstrated to be the most highly efficiency, followed by sumSquare and tri-n-gram, latter by Dice, odd gram and generalized n-gram methods as depicted in Figure 7. The experimental results indicate that bi-n-gram and sumSquare gram achieve high performance with qualitative values and outperform than Dice and generalized n-gram methods. The results achieved by the odd gram method was not exceptional better than Dice similarity values but the running times with Dice method are highly encouraging and better than generalized n-gram matching.

The results generated by the methods must be evaluated to determine their usefulness in real life of electronic test at programming languages. The evaluation used step-wise grade to determine to what percentage degree the numbers of the grade scores obtained from the Dice, bi-n-gram, and sumsquare methods are very closed to the tutors' grades. Three experts manually graded the students' assignments and

Table 2. Analysis of the results using sumsquare gram, Dice methods, bi-n-gram and Tutors' grading scores

Methods	70-100 (A)	60-69 (B)	50-59 (C)	45-49 (D)	40-44 (E)	0-39 (F)	Total
Dice	50	48	81	14	20	89	302
sunSquare	61	133	94	4	2	8	302
bi-n-gram	111	160	29	2			302
Ave (Tutor)	58	90	88	39	18	9	302

the average of the scores were converted into step-wise grade as shown in Table 2 along side the Dice method which is the best in practice. As shown in the Table 2, there are many students who passed very well using bi-n-gram method than either Dice or sumSquare gram methods whereas the numbers of grades produced by sumSquare gram method are very close to the numbers of grades by experts. Intuitively, sumSquare gram method permits to achieve a very high relatively and relevant grade results in electronic test at programming languages.

8. Conclusion

In this paper, two methods of odd gram and sumsquare gram as well as modified of bigram and trigram have been proposed to improve the performance of generalized n-gram matching of similarity measure in electronic test at programming languages. The new methods used essential features of common substrings to get better discrimination for the necessary computational performance. The computational process of the methods have drastically reduced a large amount of storage by using half nested loop to compare n-gram in the pattern match with every n-gram in the text matches. The experimental results indicate that bi-n-gram is the best among the six methods using two-digit-letter for each code line which has improved the effectiveness of n-gram analysis. The results achieved by the odd gram method was not exceptional better than Dice similarity values but the running times with Dice method are highly encouraging and better than generalized n-gram matching. The results obtained from sumSquare gram method in similarity measure are very close to the results of experts which indicate that it can be successfully used in electronic test at programming language.

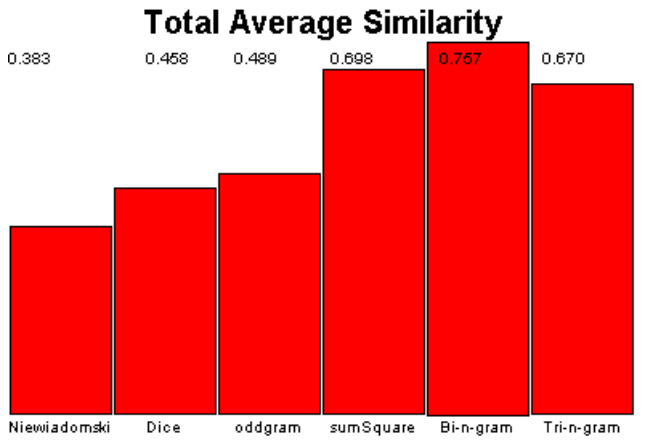

Figure 5. Total average of similarity measures of 302 pattern matches

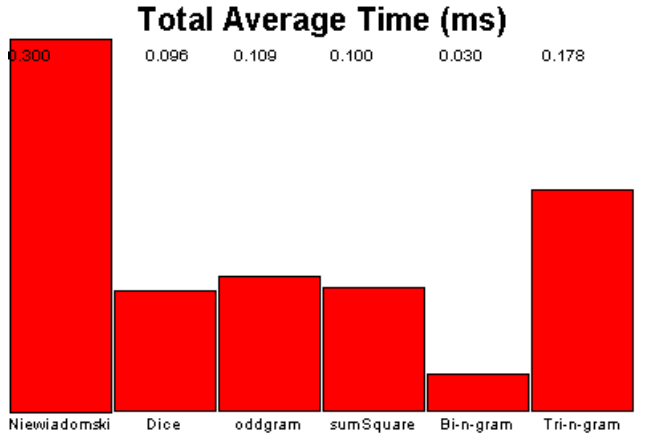

Figure 6. Total average of running times of 302 pattern matches

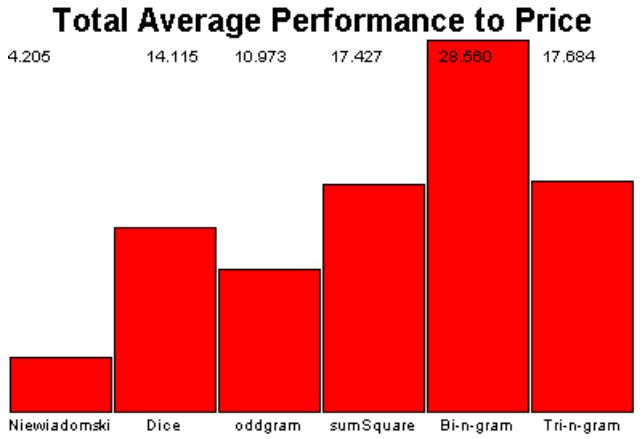

Figure 7. Total average of performance to price of 302 pattern matches

References

[1] Spinels, D., Zaharias, P., and Vrechopoulos, A., *Coping with Plagiarism and Grading Load: Randomized Programming Assignments and Reflective Grading*, Computer applications in engineering education, Vol. 5, No. 2, 2007, pp. 113–123.

[2] Markoff, A. A., *Essai d'une recherche statistique sur le text du roman*, Engene oneguine, bull. Acad imper sci. st Petersburg, Vol. 7, No. 3, 1913, pp. 153–162.

[3] Shannon, C. E., *Prediction and entropy of printed English*, The Bell System Technical Journal, Vol. 30, 1951, pp. 50–64.

[4] Zamora, E. M., Pollock, J. J., and Zamora, A., *The use of trigram for spelling error detection*, Information Processing and Management, Vol. 17, 1981, pp. 305–316.

[5] Burnett, J., Cooper, D., Lynch, M., Willett, P., and Wycherley, M., *Document retrieval experiments using indexing vocabularies of varying size*, Journal of Documentation, Vol. 35, No. 3, 1979, pp. 197–206.

[6] Trenkle, J. and Cavnar, W. B., *N-gram based text categorization*, In: Proceedings of SDAIR-94, the 3rd Annual Symposium on Document Analysis and Information Retrieval, 1994, pp. 161–175, University of Nevada, Las Vegas.

[7] Zhao, J., *Network and n-gram decoding in speech recognition*, Master's thesis, Department of Electrical and Computer Science, Mississippi State University, 2000.

[8] Cheng, B. Y., Carbonell, J. G., and Klein-Seetharaman, J., *Protein classification based on text document classification techniques*, Journal of protein, Vol. 58, No. 4, 2005, pp. 955–970.

[9] Nakamura, M. and Shikano, M., *A study of English word category prediction based on neural networks*, International conference on acoustics, speech and signal processing, Vol. 2, 1989, pp. 731–734.

[10] Tan, C. L., Sung, S. Y., Yu, Z., and Xu, Y., *Text Retrieval from Document Images based on N-Gram Algorithm*, In: Text and Web Mining Workshop, 6th Pacific Rim International Conference on Artificial Intelligence, Publisher, 2000, pp. 257–270.

[11] Harrison, M., *Implementation of the substring test by hashing*, Communication of the ACM, Vol. 14, No. 12, 1971, pp. 777–779.

[12] Damashek, M., *Gauging similarity with n-grams: Language-independent categorization of text*, Science, Vol. 267, No. 5199, 1995, pp. 843–849.

[13] Pearce, C. and Nicholas, C., *Experiments in a dynamic hypertext environment for degraded and multilingual data*, Journal of the American society for information science, Vol. 47, No. 4, 1996, pp. 263–275.

[14] Cohen, J. A., *Highlight: Language and domain independence automatic indexing terms for abstracting*, Journal of the American society for information science, Vol. 46, No. 3, 1995, pp. 162–174.

[15] Yannakoudakis, E., Goyal, P., and Huggil, J., *The generation and use of text fragments for data compression*, Information processing and management, Vol. 18, No. 1, 1982, pp. 15–21.

[16] Church, K. W. and Gale, W. A., *A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of English bigrams*, Computer speech language, Vol. 5, No. 1, 1991, pp. 19–54.

[17] Kuhn, R. and De Mori, R., *A cache-based natural language model for speech recognition*, IEEE transactions on pattern analysis and machine intelligence, Vol. 12, No. 6, 1990, pp. 570–583.

[18] Niesler, T. R. and Woodland, P. C., *A variable-length category-based n-gram language model*, In: IN PROCEEDINGS, IEEE ICASSP, 1996, pp. 164–167.

[19] Abou-Assaleh, T., Cercone, N., Keselj, V., and Sweidan, R., *N-gram based detection of new malicious code*, In: COMPSAC '04 Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts - Volume 02, 2004, pp. 41–42.

[20] Niewiadomski, A., *Methods for the linguistic summarization of data: Application of fuzzy sets and their extensions*, EXIT Piblishing House, Warsaw, 2008.

[21] Pascual, J.-I., *A procedure for the construction of a similarity relation*, In: Proceedings of IPMU'08, Malaga, edited by L. Magdalena, M. Ojeda-Aciego, and J. Verdegay, 2008, pp. 489–496.

[22] Wang, J., Li, G., and Fe, J., *Fast-Join: An efficient method for fuzzy token matching based string similarity join*, In: IEEE 27th International Conference on Data Engineering (ICDE), 2011, pp. 458–469.

[23] Arsmah, I. and Zainab, A. B., *Automated grading of linear algebraic equation using n-gram method*, Tech. rep., Institute of Research, Development and Commercialization, Universiti Teknologi MARA, 2005.

[24] Buckles, B. P. and Petry, F., *Information theoretic characterization of fuzzy relational databases*, IEEE transaction systems man cybernet, Vol. 13, No. 1, 1983, pp. 74–77.