2014

Determining an inter-rater agreement metric for researchers evaluating student pathways in problem solving

Austin David Sullivan
Iowa State University

Determining an inter-rater agreement metric for researchers evaluating student pathways in problem solving

by

Austin D. Sullivan

A thesis submitted to the graduate facility

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major
Major: Industrial Engineering

Program of Study Committee:
John Jackman, Major Professor
Stephen Gilbert
Connie Hargrave

Iowa State University

Ames, Iowa

2014

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Jackman, for all his encouragement, support, and patience in the development of this thesis. In addition, I thank you for the guidance you have provided in my academic and professional careers. I do not believe I would be where I am today without your support. Thank you to my committee members, Dr. Gilbert and Dr. Hargrave for great insights and support during this process.

I would like to thank my friends and colleagues who have been there over the years. I have formed an incredible network of close personal relationships that I look forward to retaining in the years to come. You have all influenced me both academically and culturally and I am humbled by the generosity I have been shown. Furthermore, I would like to thank the department faculty and staff at Iowa State University for making my adventure an unforgettable experience.

Finally, thanks to my family for if it weren't for their loving support this would not have been possible.

ABSTRACT

A new inter-rater agreement metric (IRRA) was developed for measuring agreement between multiple research coders when they code activities as they observe student problem-solving sessions. The complex nature of the student data includes activities that 1) are dependent on prior activities, 2) are ordinal data types, 3) can occur at any point in time, and 4) can reoccur. The assumptions used in traditional inter-rater agreement metrics are violated in this context and may lead to erroneous conclusions in particular datasets. In this study, coded activities are considered to be a sequence codes that can be analyzed using a string matching algorithm. We evaluated the metric's performance by simulating the variability of coders in a controlled fashion. The results show that the algorithm performed well as an inter-rater agreement metric over a wide range of conditions.

CHAPTER 1: INTRODUCTION

1.1 Introduction

Student pathways during problem solving have been studied to understand the rationale a novice student uses while solving a problem and how the novice strategy differs from an expert pathway (Antonenko, et al., 2011). Expert pathway strategies differ from novice pathways because of the amount of knowledge gained from prior domain experience (Ericsson, 2006). Protocol analysis is widely used to characterize novice and expert behavior by articulating cognitive activities (Ericsson & Simon, 1993).

In protocol analysis, research coders use a coding scheme to assign codes to activities they observe as students solve problems. A coding scheme is a finite set of codes and descriptions of the possible activities. Each coded activity has a code, start time, and end time. Ideally, two research coders analyzing the same data for a student should produce the same set of coded activities. The degree of agreement between sets varies because even with training, coders interpret observed behavior differently. In studies involving multiple coders, a high degree of agreement is necessary to make valid conclusions. Inter-rater agreement metrics measure the similarity of results from multiple coders (Gwet, 2001).

The goal of this research is to develop and evaluate a new method for comparing coded activity sets produced by two or more research coders.

1.2 Statement of the problem

Measuring inter-rater agreement in student pathway studies is difficult because of the complex nature of the coded data. Inter-rater measurements typically focus on data types such as discrete, nominal, or ordinal datasets (Tinsley & Weiss, 1975). For

continuous data, statistical methods such as analysis of variance have been used (Neter, 1996). Comparing coded activity sets is challenging because: 1) coded values depend on prior activities, 2) activity types are nominal or categorical, 3) coded activity times are continuous variables, and 4) coded activity types appear multiple times throughout the analysis. Numerous metrics have been used to measure agreement between research coders, but there is no definitive metric.

1.3 Purpose of the study

The goal of this study is to develop a new comparison method based on a string-editing algorithm and evaluate its suitability as an agreement metric for coded activity sets. Protocol analysis is a new problem domain for string-editing algorithms; however, similar algorithms have been used and proven effective in a variety of applications including string comparisons, computational biology, signal processing, text retrieval, and social sciences (Abbot & Forrest, 1986; Navarro, 2001). These algorithms measure the cost associated with transforming one sequence of elements into another. Without this study a sufficient measurement quantifying coder bias in all protocol analysis datasets does not exist which can lead to incorrectly evaluated datasets from research coders.

1.4 Significance of the study

We propose a new method for measuring agreement between multiple research coders in contexts where reoccurring activities can be coded as a sequence of alphabetical characters (i.e., a sequence of activities). When this method indicates a high degree of agreement, research coders may perform protocol analysis studies independently as coders are capable of accurately applying the coding scheme. A low degree of agreement

indicates that activity descriptions may be too ambiguous, the coding scheme needs to be modified, or research coders need more training.

Training increases the level of inter-rater agreement among coders (Pulakos, 1984). In some instances reoccurring training may be beneficial because individual raters may change judgments over time (Myford & Wolfe, 2009). An inter-rater metric, such as the one proposed in this study, can be used as an indicator for the degree of proficiency of coders.

1.5 Definition of terms

Activity –Used synonymously with code and corresponds to a cognitive activity in a coding scheme having *n* codes.

Coding XML (Extensible markup Language) Files – defined by a schema, this file is exported by the ANVIL annotation software program (Kipp, 2001) which contains a research coder's set of coded activities.

Inter-rater agreement – Multiple coders agree on the interpretation of student pathway activities.

Edit distance – Used synonymously with *Levenshtein distance* to measure the minimum number of insert, delete, and substitution operations to change one sequence into another.

Protocol Analysis – The process that a research coder uses to articulate the cognitive activities of a student recorded as audio and video data (See Appendix A).

Research coder –Individual who performs the coding of activities, also referred to as a rater.

Student pathway **–** A sequence of cognitive activities performed by a student while solving a problem.

1.6 Thesis organization

The remainder of this thesis is organized as follows. *Chapter 2* provides a literature review on three relevant topics:

1) Traditional views of inter-rater agreement within content analysis
2) Algorithms used in sequence analysis
3) Shortcomings of a sequence analysis approach

Chapter 3 presents some of the issues with existing inter-rater agreement metrics. In Chapter 4 we describe the new method and the evaluation methodology. The results and conclusions are presented in Chapter 5. Chapter 6 is a discussion of opportunities for future research.

CHAPTER 2: LITERATURE REVIEW

2.1 Inter-rater agreement metrics

Content analysis as defined by Kolbe and Burnett is "an observational research method that is used to systematically evaluate the symbolic content of all forms of recorded communication" (Kolbe & Burnett, 1991 pg. 243). The analysis of these communications depends on the goal of the study and the level of detail required (e.g., sentences, words, or letters). Inter-rater agreement is "the extent to which the different judges tend to assign exactly the same rating to each object" (Tagg, 2007 pg. 98). There are numerous inter-rater metrics that have been used to measure the degree of agreement. Typically, these metrics have values between 0.0 (no agreement) and 1.00 (perfect agreement).

One of the first measures, percent agreement, or Holsti's agreement (Holsti, 1969), is easy to calculate and understand, but does not take into consideration the probability of agreeing by chance. Not being able to consider the probability of chance agreement is problematic as a value close to 1.0 would not always represent true agreement between raters. The problem becomes more severe when there are few classification categories in the coding scheme and the probability of being correct by chance becomes larger.

Scott's Pi (Scott, 1955) does take into consideration chance agreement, yet can only be used for nominal or categorized data for two raters. Fleiss later generalized Scott's Pi to any number of raters given a nominal dataset (Fleiss, 1971). Both Scott's Pi and Fleiss' Kappa take into consideration chance-agreement, yet assume coders have

distributed their codes uniformly over the entire set of codes. If this assumption is violated, the statistic is biased towards reduced agreement (Craig, 1981).

Cohen's Kappa (Cohen, 1960) was based on Scott's approach to nominal data, but includes a different measurement of chance agreement. This was accomplished using the concept of a *multiplicative marginal*. Craig reported that Cohen's Kappa was the most frequently used and well understood inter-rater statistic in the 1980's (Craig, 1981) and is still widely used today. Some weaknesses of Cohen's Kappa include the inability to handle missing data, unequal sample sizes, and different informational types (such as ratio, interval, and polar data). It is also limited to pairwise comparisons between raters.

Krippendorff's Alpha (Krippendorff, 1980) addressed the ability to handle any number of raters and additional data types. Krippendorff's Alpha has been shown to be a better metric for some applications (Hayes & Krippendorff, 2007). However, it is much more complex in usage than Cohen's Kappa and is implemented in statistical software (Hayes & Krippendorff, 2007).

All these traditional inter-rater measurements, while applicable in many cases, do not consider activity timing in measuring agreement. In some contexts, the time of an occurrence can be very important in determining agreement. Rater 1 may state that an activity starts at time *t = 0 units* while Rater 2 coded the same activity as occurring at time *t = 100 units*. The raters are in agreement on the activity, but in disagreement on the timing of its occurrence. Additionally, all traditional methods assume independent observations which are not the case in protocol analysis research as future activities are based on prior activities.

2.2 Algorithms used in sequence analysis

A *sequence*, or categorical time series, is a listing of elements (MacIndoe & Abbott, 2004). These elements can be defined as:

- A state at a point in time (e.g., employed vs. non-employed)
- A physical object (e.g., base pair of DNA)
- An event (e.g., dance step)

These elements are associated with fixed portions in time (e.g. every month) or are the result of a specific action (e.g. accepted a job offer). Sequence analysis is the study of comparing these ordered arrangements. If we describe student behavior while solving engineering problems as a sequence of activities, then sequence analysis is applicable. Sequence analysis has its roots in the classical approximate string matching problem in the field of computer science and can be solved by numerous dynamic programming techniques (Levenshtein, 1966; Sellers, 1974; Masek & Paterson, 1980).

Additional sequence analysis algorithms have been studied, such as the longest subsequence problem in which the goal is to evaluate the longest matching subsequence within two different strings (Illiopoulos & Rahman, 2009). Only a subset of relevant approximate sequence matching techniques are reviewed here; for a comprehensive review of the string matching problems and applications refer to Navarro (2001).

Given any two sequences of equal length the Hamming distance identifies differences by counting the number of positions where the characters in the two strings are different (Hamming, 1950). While effective, this algorithm requires two strings of equal length and cannot evaluate differences in positioning. To address this limitation, the Levenshtein distance, commonly referred to as the edit distance, is a metric uses the

transformational operations of insertion, deletion, and substitution of strings (Levenshtein, 1966). This algorithm not only lifts the restriction that strings be of equal length, but adds the ability to check character positions. The Hamming distance between the two strings “transform” and “ransform” is 9, but the edit distance is only 1 because a “t” can be inserted (i.e., 1 operation) at the beginning. Levenshtein distance is bounded between 0 (strings are the exactly the same) and the maximum length of the two strings (the two strings are completely different).

Needleman and Wunsch (1970) described an algorithm that uses the same concept of Levenshtein distance but uses two additional arrays. The first array, a *conversion matrix*, is used as a cost reference when converting a character to another character. The second parameter is a *penalty factor*, the cost of an insertion operation per character. These two parameters allow a cost adjustment for a specific operation. Because of the different cost structure, the metric is no longer bounded between 0 and the maximum length of the two sequences.

The Needleman and Wunsch algorithm has been referred to as the *evolutionary distance* when measuring the difference between two nucleotide sequences (Sellers, 1974) and *optimal matching* when measuring the distances of social event sequences over time (Abbott & Forrest, 1986). There is a substantial amount of recent research on *optimal matching* which is “the most frequently used technique for comparing sequences” (Brzinsky-Fay & Kohler, 2010 pp. 360). For additional applications of optimal matching see Kruskal (1983), Brzinsky-Fay & Kohler (2010), and Blanchard et al. (2012).

In the Needleman and Wunsch approach (1970), conversion matrices largely influence the final result and therefore, a proper conversion matrix must be carefully chosen. If the conversion matrix is invalid, the distance between sequences is inaccurate due to linked relationship between the arrays and the results. When the Needleman and Wunsch algorithm is used with a unit-cost conversion matrix and penalty gap, then it is equivalent to the Levenshtein distance.

These algorithms have been used to search large databases for similarity between two sequences, such as amino acid sequences. Heuristic algorithms have been introduced to deal with the large number of comparisons that must be performed (Alschul et al. 1990; Mabroukeh & Ezeife, 2010). These methods do not provide the same level of accuracy as the previously mentioned algorithms, but the computational time is greatly reduced, providing added value when comparing large datasets.

2.3 Limitations of the sequence approach

Although sequence algorithms are extremely useful, some limitations exist that should be recognized when evaluating their use in measuring similarity (Dijkstra & Taris, 1995). A number of issues related to using optimal matching algorithms have been discussed (Abbot, 1995; Abbot, 1997; Abbott & Forrest, 1986; Lesnard, 2006).

One shortfall is the ability to validate the conversion matrix, which determines the cost when certain elements are displaced, removed, or substituted. This is central to the algorithm’s decision making process. String editing algorithms make minimization choices based on the cost of converting one sequence to another (See Appendix B). Therefore, the final distance is highly sensitive to the conversion matrix associated with

the algorithm (Wu, 2000). If an accurate determination of the costs between elements cannot be performed, then the distance is inaccurate (Wu, 2000).

Sequence algorithm structures can only classify discrete data. If continuous data is used then a discrete approximation is necessary. Next, sequence algorithms do not consider the significance of activities or whether a series of activities causes changes in downstream activities (Dijkstra & Taris, 1995; Wu 2000; Elzinga, 2003). For example, a protocol analysis starting with an activity *Calculating* is not possible before the student completes the activity *Reads Problem Description*. Yet, these algorithms will calculate a distance as if it were possible with no indication of any error.

Another limitation is that sequence algorithms are indifferent to the direction of time (Wu, 2000). For example, the difference between sequences {A, B, C, D} and {B, C, D} is 1 using a unit cost edit distance algorithm. The same is true for sequences {A, B, C, D} and {A, B, C, B}. When the timing of an activity is important in measuring the similarity of two sequences, such as in some applications of optimal matching, these algorithms may not be appropriate.

In response to these shortcomings, Lesnard (2006) recommends that sequence algorithms be treated as sequence arithmetic. Lesnard describes sequence algorithms as "an abstract operation, in this respect not very different from calculating the arithmetic mean of a series of numbers … substitution operations are just some of the building blocks of the abstract process of assessing the degree of similarity between sequences" (pg. 10).

CHAPTER 3: USING INTER-RATER AGREEMENT METRICS FOR STUDENT PATHWAY DATA

3.1 Explanation of inter-rater agreement vs. reliability

In many applications the terms inter-rater agreement and reliability are used interchangeably. We follow the convention of Tinsley & Weiss (1975) on the difference between the two measures of similarity. Inter-rater *agreement* statistics measure the differences in values between two coders (raters) while evaluating a subject under similar circumstances. "When judgments are made on a numerical scale, interrater agreement means that the judges assigned exactly the same values when rating the same person" (Tinsley & Weiss, 1975 pp. 359). They recommend that differences between coders when rating an individual subject should be evaluated with an inter-rater agreement metric.

Inter-rater *reliability* differs in the way it quantifies the relationship between two coders over the evaluation of multiple subjects. Inter-rater reliability is usually reported in terms of linear correlation or ANOVA analysis (Tinsley & Weiss, 1975).

We illustrate the difference between agreement and reliability using the two hypothetical datasets shown in Tables 1 & 2. Inter-rater agreement focuses on single subject comparisons. When there is little agreement for individual subjects, reliability can still be high as shown in Table 1. Rater 2 in Table 1 always evaluates a subject three units greater than Rater 1 (i.e., no agreement). The relationship between raters is highly correlated (reliable) so that given a single rater's rating the second rater's evaluation may be predicted with high precision.

Data can also have high inter-rater agreement, but little reliability as shown in Table 2. Raters have disagreement on only one of five subjects. Yet, it is more difficult to understand the relationship between the raters so the reliability (correlation) is lower than in Table 1. The quantitative relationship between agreement and reliability is analogous to the relationship between accuracy and precision in a numerical dataset. Each metric provides a different viewpoint of the same story.

Table 1: Low Agreement & High Reliability dataset

Subject Number	Rater 1	Rater 2
Subject 1	1	4
Subject 2	3	6
Subject 3	4	7
Subject 4	2	5
Subject 5	3	6
Mean	2.8	5.8
Correlation:	1.00	

Table 2: High Agreement & Low Reliability dataset

Subject Number	Rater 1	Rater 2
Subject 1	4	4
Subject 2	3	3
Subject 3	4	4
Subject 4	3	3
Subject 5	3	5
Mean:	3.8	4.0
Correlation:	0.22	

In summary, having a high degree of inter-rater reliability does not ensure high inter-rater agreement or vice versa. With the Tinsley & Weiss (1975) definitions in mind, the goal of this research is to provide a robust inter-rater agreement metric for *n* coders evaluating a single student pathway. Coder reliability over multiple student pathways was not considered.

3.2 Cohen's Kappa

Cohen's Kappa is one of the most popular statistics for inter-rater agreement (Craig, 1981). However, Cohen's Kappa has two issues for coded sets. 1) Kappa can only be calculated between two coders and 2) cannot take into account a timing error on the

activity scheme as a whole. The second issue occurs as coders are evaluating categories over a period of time. This limitation is illustrated using a simple coded activity set shown in Tables 3 and 4. The data is further divided into one-second intervals because each activity can have a variable amount of time.

Table 3: Cohen's Kappa calculation - Coder 1 results

Activity	Start Time	End Time	Code
Finding information	0	5	A
Monitoring their status	5	10	B
Finding information	10	20	A
Monitoring their status	20	25	B
Calculating	25	35	C

Table 4: Cohen's Kappa calculation - Coder 2 results

Activity	Start Time	End Time	Code
Finding information	0	2	A
Monitoring their status	2	7	B
Finding information	7	17	A
Monitoring their status	17	22	B
Calculating	22	32	C

By partitioning the observations into one second time intervals we obtain the following sequences.

Rater 1: `AAAAABBBBBAAAAAAAAAABBBBBCCCCCCCCCC`

Rater 2: `AABBBBBAAAAAAAAAABBBBBCCCCCCCCCC---`

The sequences are of unequal length (35 codes versus 32 codes) and each code has the same duration (one second). Unequal lengths are common in student protocol analysis studies so sequences are made to be of equal length by inserting blanks in the remaining time slots. These coded sequences are not the same, but highly similar. Coder

2 coded activities lag three seconds behind Coder 1. Coders are in strong agreement, but a disagreement on timing yields unfavorable results.

Cohen's Kappa statistic treats all 35 one-second intervals as independent observations. A two dimensional confusion matrix (see Table 5) shows the relationship between coders at each second. Each column is a code assigned by Coder 1 and the row elements shows the number of codes assigned by Coder 2. In the same way, each row is a code assigned by Coder 2 and the column elements are the codes assigned by Coder 1. The matrix values are the summation of all activities by each rater.

Table 5: Confusion matrix for coders 1 and 2, highlighted indicates agreement

		Coder 1				
		A	B	C	Blank	Row totals
Coder 2	A	9	3	0	0	12
	B	6	4	0	0	10
	C	0	3	7	0	10
	Blank	0	0	3	0	3
	Column totals	15	10	10	0	35

The total number of samples is given by

$$N = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \quad (1)$$

where a_{ij} represents a single matrix entry value in row i and column j.

To compute Cohen's Kappa the first step is to calculate observed agreement of events the two research coders agreed upon (*OA*). This can be found by summing the values across the diagonal and dividing by the number of observations or

$$OA = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}}{N} \; \forall i = j \;\; (2).$$

The OA equation

$$OA = \frac{9 + 4 + 7}{35}$$

equates 0.57. Observed agreement (OA) is the percent agreement, or Holsti's method (Holsti, 1969). The original dataset (Tables 3 and 4) provided an example of strong agreement offset by a constant 3 seconds and Holsti's method does a poor job quantifying that relationship declaring that raters agree 57% of the time.

Unlike Holsti's method, Cohen's Kappa accounts for chance agreement. The expected frequency (*ef*) is calculated by multiplying the row total by the column total and dividing by the total number of ratings or

$$ef_{ij} = \frac{\sum_{i=1}^{n} a_i \sum_{j=1}^{n} a_j}{N} \;\; (3).$$

Expected frequency (*ef*) can be calculated across the entire matrix (Table 6) however, the main concern is only the entries of agreement. These entries can be found across the diagonal in parenthesis.

Table 6: Expected frequency (*ef*) values highlighted summing to expected agreement

		Coder 1				
		A	B	C	blank	Row totals
Coder 2	A	9 (5.14)	3	0	0	12
	B	6	4 (2.86)	0	0	10
	C	0	3	7 (2.86)	0	10
	blank	0	0	3	0 (0.0)	3
	Column totals	15	10	10	0	35

Expected agreement (*EA*), defined as

$$EA = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} ef_{ij}}{N} \; \forall i = j \quad (4)$$

uses the expected frequency (ef_{ij}) by summing the values across the diagonal of the expected frequency matrix (see Table 6).

Consequently, the expected agreement (*EA*) or entries expected to be correct by coder guessing, is 10.86 of the 35 total samples in this dataset. Finally, Cohen's kappa (*K*) is defined as

$$K = \frac{\mathrm{NA}-\mathrm{EA}}{1-EA} \quad (5).$$

For this example, we obtain

$$K = \frac{0.57-0.31}{1-0.31} = 0.38.$$

According to the scale suggested by Landis & Koch (1997) Tables 3 and Table 4 indicate only fair agreement.

Another more extreme example is given by the following two sequences:

Rater 1: `ABCDEFGH`

Rater 2: `BCDEFGH`

where here the behavioral activities are not parsed by one second intervals. The set of events are off by a position of one activity (A). Cohen's Kappa produces value of 0.00 in this example.

No traditional inter-rater measurement can account for this kind of lagging behavior over time periods (Cohen, 1960; Fleiss, 1971; Holsti, 1969; Krippendorf, 1980; Scott, 1955).

CHAPTER 4: INTER-RATER METRIC FORMULATION FOR STUDENT PATHWAY SEQUENCES

4.1 Definition of a sequence

A sequence, also known as a categorical time series, is a listing of elements (Abbot, 1995) and can be defined as an ordered set of *n* elements, S = { a_1, a_2, …a_n }. Letters (i.e., codes) are often used to define a sequence element rather than the full activity description. Codes make it easier to write and understand the sequence of activities. Table 7 is an example of a very simple sequence in a coded activity set.

Table 7: Sequence of recurrent activities

Activity Description	Code	Cumulative sequence of activities
Describing a problem description	A	A
Finding Information	B	AB
Drawing a diagram	C	ABC
Picking information	D	ABCD
Drawing a diagram	C	ABCDC

Sequences have certain characteristics that can distinguish them from each other. First, sequence elements can either be unique or repeatable. If an event in a sequence is unique and cannot be repeated, it is a *nonrecurrent* sequence. If a sequence represents sampling elements with replacement, it is a *recurrent sequence.* In Table 7 the sequence is recurrent because the element *Drawing a diagram* (C) appears twice. Sequences can have dependencies between their elements. This property is similar to a stochastic Markov process where the *n+1* state depends on the *nth* state. An example of this

scenario would be that it is more likely the activity *Calculating* to immediately follow *Drawing a diagram/figure*.

4.2 Basis for the metric

String matching algorithms can be viewed as a type of spell checker measuring the distance from one word to another in a dictionary. Given that comparisons are always performed pairwise, the inter-rater agreement metric must address comparisons of three or more coders. The student video and audio data have a defined length and therefore both protocol analysis outputs are similar in length, but not typically equal due to the coders applying codes within continuous time within the coding software (Kipp, 2001). Therefore, Hamming distance (1950) is automatically excluded as a potential candidate because it assumes equal lengths.

For each insertion, deletion, or substitution, unique costs can be associated with each operation. When comparing student pathway data, an assumption was made that all costs are equal to 1. This assumption ensures that the maximum dissimilarity is the maximum length of the two sequences (i.e. sequences are completely different), and assumes all coding disagreements are equally important. Typically, costs of transforming one code to another are inversely proportional to the frequency of each code being substituted for another (Lesnard, 2006). Common transformational errors in text analysis would be less costly and rare transformation errors would be associated with higher costs.

According to Chapter Wu (2000) introducing a conversion matrix without proper analysis could yield incorrect distance measurements if not carefully evaluated and the data to validate the matrix was not available. Also, the Needleman and Wunsch (1970) algorithm is bounded by the conversion matrix and therefore, more difficult to normalize.

Levenshtein distance, or Needleman and Wunsch with a unit cost conversation matrix, constrains the distance between 0 (sequences are exactly the same) and the maximum length of the two sequences (sequences are completely different).

4.3 Notation

m: sequence 1 length

n: sequence 2 length

v: matrix defined of dimensions ***n*** x ***m***

i: matrix ***v*** row iterator (0 to ***m***)

j: matrix ***v*** column iterator (0 to ***n***)

k: single pairwise combination of two coder sequences

NR: number of coders (raters) evaluating single student pathway

NC: total number of coder combinations

$\boldsymbol{ED_k}$: Edit distance between coder combination ***k***

$\boldsymbol{NED_k}$: Normalized edit distance (dissimilarity) between coder combination ***k***

$\boldsymbol{S_k}$: Similarity for coder combination ***k***

IRRA: Average similarity among all coders (***NR)***

4.4 Metric algorithm

Based on the coded activity set properties and the assumption that all transformation costs are 1, the algorithm chosen for implementation of the edit distance was a dynamic programming algorithm (Wagner & Fischer, 1974). This is due to its ease of implementation and computational efficiency in calculating a unit cost edit distance. The dynamic programming algorithm for calculating the edit distance for one pair of coder sets is shown in Figure 1.

Matrix, *v*[*m*][*n*], is created where *m* is the length of the first code set and *n* is the length of the second code set. The rest of the algorithm minimizes the costs associated to transformational operations to this matrix. A detailed example walkthrough of the algorithm can be found in Appendix B.

```
int MinEditDistance(string seq1, string seq2)
{
int m = seq1.length();
int n = seq2.length();

//Define first row of cost matrix
for (int i = 0; i <= m; i++) {
    v [i][0] = i;
}

//Define first column of cost matrix
for (int j = 0; j <= n; j++) {
     v [0][j] = j;
}

//Find the minimum distance for each cell in the matrix by evaluating minimum of //1 +
cell left, above, and left above
for (int i = 1; i <= m; i++) {
  for (int j = 1; j <= n; j++) {
       if(seq1[i] == seq2[ j ]) then
             v[i][j] = v[ i–1][ j–1]
          else
              v[i][j] = 1 + min (min (v[i][j–1], v[i–1][j], v[i–1][j–1]) )
    }
}

//Return the final value in the cost matrix
return v[m][n]
}
```

Figure 1: Minimum Edit distance pseudocode, returns single edit distance (ED_k)

The algorithm returns one value, v[m][n], the scalar minimized edit distance for a single pairwise combination of the research coders (ED_k). Similar to Scott's Pi or Cohen's Kappa, sequence editing algorithms perform a single pairwise comparison of

sequences. To provide an average rating among all research coders, all combinations of coders must be considered. Given a list of three research coders, each performing a protocol analysis on a student solving a problem, the unique pair-wise combinations between raters {1, 2}, {1, 3}, and {2, 3} are shown in Table 8.

Table 8: Unique pairwise combinations for three coders (k_1, k_2, k_3)

$k = 1$	$k = 2$	$k = 3$
{1, 2}	{1, 3}	{2, 3}

One weakness of the edit distance is that it strictly provides a magnitude value and lacks the ability to take into consideration the length of the sequences being compared. The edit distance can be normalized by dividing by the maximum possible edit distance (maximum length of two sequences). This can only be done with when all costs are 1. Otherwise, the distance is bounded by values determined by the conversion matrix.

The normalized edit distance has a value between 0 and 1 and is found by dividing the scalar value edit distance (ED_k) by the maximum length of the two sequences (strings) being compared. The normalized edit distance for combination k of two code sequences, is given by

$$NED_k = \frac{ED_k}{Max\big(Length(sequence1, sequence2)\big)} \quad (6).$$

Equation (6) measures the amount of dissimilarity between 0 and 1, with 1 corresponding to the maximum dissimilarity. Since similarity (S_k) is the complement of dissimilarity (NED_k), it follows that

$$S_k = 1 - NED_k \quad (7).$$

Equation (7) is limited in that it provides only similarity between two pairwise sequences. The total number of unique combinations (NC) is given by

$$NC = \binom{NR}{2} = \frac{NR!}{2!\,(NR-2)!} \quad (8)$$

where *NR* is the number of research coders.

Average similarity is the sum of all S_k divided by the total number of unique coder combinations (NC). This yields a singular normalized distance for the agreement of *N* research coders for a single student pathway dataset. This inter-rater agreement (IRRA) metric is the average similarity and is given by

$$IRRA = \frac{\sum_{k=1}^{NC} S_k}{NC} \quad (9).$$

IRRA has a value between 0 (no agreement) and 1 (complete agreement) for any number of coders. Typical studies use three coders or less, therefore, estimates of the variance of S_k would be inaccurate.

4.5 Evaluation methodology

The following steps were performed to evaluate the suitability of IRRA as an inter-rater agreement metric.

Step 1) Generate coded activity sets for analysis

To evaluate IRRA in a controlled manner, a *protocol analysis simulator* was created so that parameters affecting similarity could be varied. In addition, the simulator allowed the creation of any number of coded activity sets (representing coders) which are otherwise limited by the actual number of coders in a study. The simulator uses a set of probability distributions to produce random perturbations of the original coded activity

set which represents the "true" set of codes for a which is randomly generated. The parameters that are used to control the perturbations are as follows.

Length of code (sec) – The duration of an activity has a uniform distribution $\sim U(a, b)$.

Agreement – This is the probability of success, *p*, for a Bernoulli random variable indicating whether a code in the original coded activity set will remain unchanged (i.e., success is no change and a failure results in a perturbation). Therefore, the total number of perturbations has a binomial distribution. Using Monte Carlo simulation, we generate a value (success or failure) from the Bernoulli distribution for each code.

If it is a success, then there are three equally likely perturbations – delete the code, modify the code, or insert a new code. If a code is deleted, then all subsequent codes are shifted on the time scale by the duration of the deleted code. If a code is modified, then 70% of the time a new code value is randomly generated and 30% of the time the duration of the code is modified by randomly generating a new *Length of code*.

A value of 0.0 for *Agreement* would produce complete random datasets, while a value of 1.00 corresponds to the same sequence.

Repetitions –This parameter specifies the number of different research coders or protocol analyses being performed on a single recorded student session.

File Length (min/max) – The length of the original coded activity set has a uniform distribution $\sim U(a, b)$.

Time between codes (sec) – Mean empty (blank) time between activities where time between activities has an exponential distribution.

The parameters are specified as shown in the example in Figure 2.

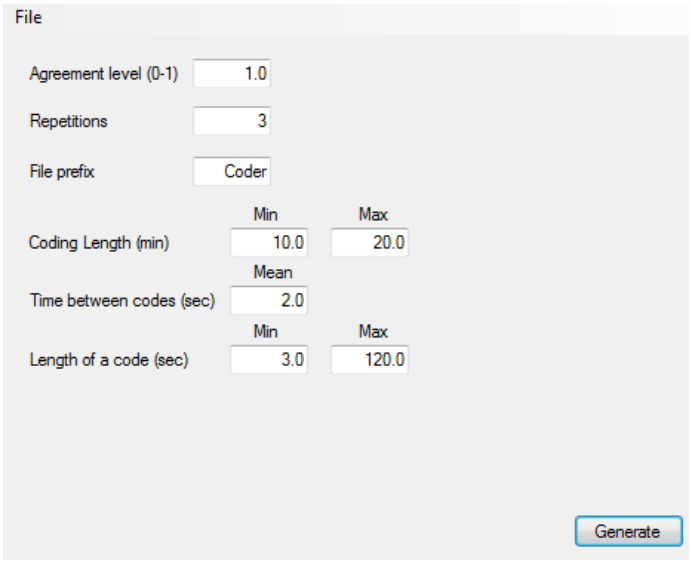
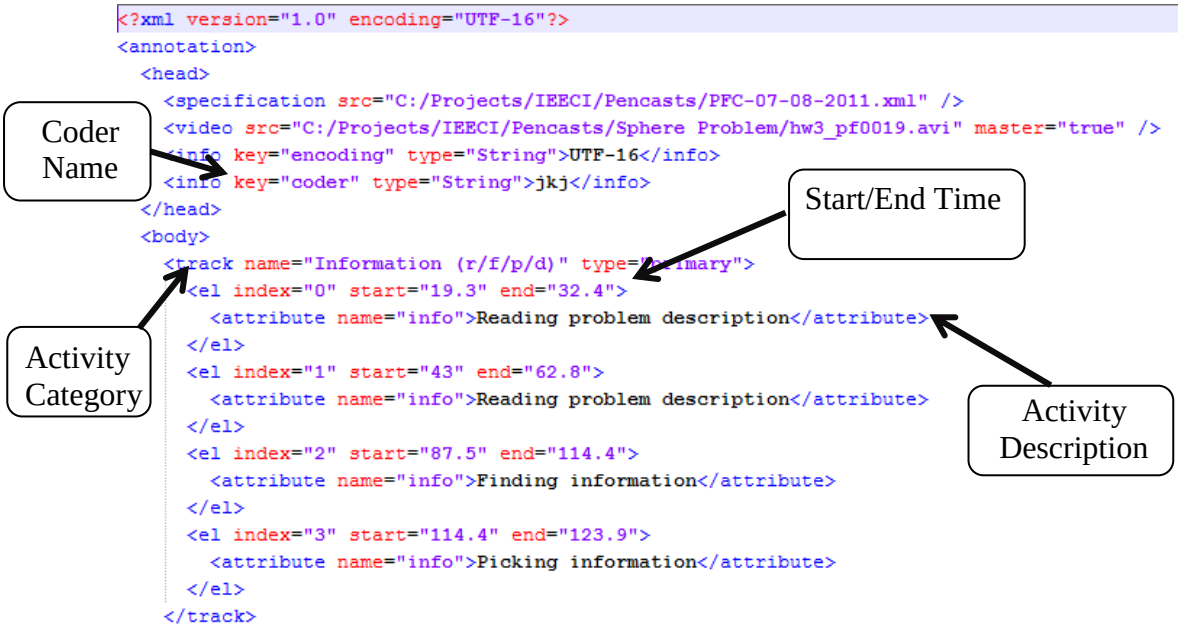


Figure 2: Protocol Analysis Simulator Parameters

After specifying the parameters, the simulator generates the codings in ANVIL (Kipp, 2001) compatible XML files. These files have the same structure as those recorded manually by coders. An example of the file structure is shown in Figure 3.




```
<?xml version="1.0" encoding="UTF-16"?>
<annotation>
  <head>
    <specification src="C:/Projects/IEECI/Pencasts/PFC-07-08-2011.xml" />
    <video src="C:/Projects/IEECI/Pencasts/Sphere Problem/hw3_pf0019.avi" master="true" />
    <info key="encoding" type="String">UTF-16</info>
    <info key="coder" type="String">jkj</info>
  </head>
  <body>
    <track name="Information (r/f/p/d)" type="primary">
      <el index="0" start="19.3" end="32.4">
        <attribute name="info">Reading problem description</attribute>
      </el>
      <el index="1" start="43" end="62.8">
        <attribute name="info">Reading problem description</attribute>
      </el>
      <el index="2" start="87.5" end="114.4">
        <attribute name="info">Finding information</attribute>
      </el>
      <el index="3" start="114.4" end="123.9">
        <attribute name="info">Picking information</attribute>
      </el>
    </track>
```

Figure 3: Protocol Analysis Simulator Output

Three sample datasets were generated using the simulator in order to evaluate parameter effects on IRRA. For each variable tested 300 independent simulations were generated each representing a new student pathway to be analyzed by the coders. Each time a parameter was changed for a dataset 300 additional simulations were run.

4.6 Effect of variation in activity agreement

This dataset was designed to evaluate the effect of varying levels of agreement on IRRA. Simulator values were *Agreement* were varied between at 0.00 and 0.95.

Dataset 1 Simulation Parameters:

Agreement: 0.00 | 0.20 | 0.40 | 0.60 | 0.80 | 0.95

Repetitions: 3

Coding Length: 10.0 – 20.0 min

Time between codes (sec): 5

Length of a code (sec): 3 – 120

4.7 Effect of number of coders

The number of coders should not affect the overall average, yet should decrease the standard deviation among independent observations. The *Repetitions* parameter was varied in the simulator to study the effects of increasing the number of coders. The activity time variation was set at a medium level as compared with the first dataset. The numbers of research coders tested was: 3, 5, 10, and 20. For the case of 20 coders a total of 6000 coded XML files were generated (20 coders x 300 student pathways) for evaluation. Agreement of 0.80 was chosen as it provided the most uncertainty, or variance, from Dataset 1 results to study the effects of increasing coders.

Dataset 2 Parameters:

Agreement: 0.80

Repetitions: 3 | 5 | 10 | 20

Coding Length: 15 - 20 min

Time between codes (sec): 2

Length of a code (sec): 30 – 60

4.8 Effect of number of activities

The final dataset examined the effect of number of activities on IRRA. The *Coding Length* parameter was varied while the *Length of a code* parameter was set to a constant. *Time between codes* is the inverse of frequency of codes and was fixed at 2 seconds. An *Agreement* value of 0.80 was chosen as it resulted in the most variation in Dataset 1.

Dataset 3 Simulation Parameters:

Agreement: 0.80

Repetitions: 3

Coding Length: 10 | 20 | 40 | 80 min

Time between codes (sec): 2

Length of a code (sec): 30

Step 2) Transform XML files into coded sequences

The XML files from Step 1 were transformed into coded sequences based on the mapping in Table 9. The activities were segmented into one second intervals before calculating IRRA. Without first parsing into one second intervals, the codes would be of varying lengths. Furthermore, not parsing by one second allows two coded sets to have

identical code sequences, but different times associated with individual code activities which is less than ideal. Times related with no activities are associated with a *Blank* activity in order to distinguish time slots with no activity apart from those with activities before applying the measurement.

Table 9: Activity code map

Category of Activity	Activity Description	Short-hand letter representation
Information	Reading problem description	A
Information	Finding Information	B
Information	Picking Information	C
Information	Deselecting Information	D
Reasoning	Connecting Multiple phenomena	E
Reasoning	Drawing a diagram/figure	F
Reasoning	Writing an equation	G
Reasoning	Identifying relevant phenomena	H
Reasoning	Stating an assumption	I
Reasoning	Describing a relevant principal	J
Reasoning	Justifying phenomena, assumption, …	K
Metacognitive	Developing a plan	L
Metacognitive	Monitoring their status	M
Metacognitive	Evaluating their approach	N
Metacognitive	Justifying their plan or evaluation	O

Step 3) Evaluate and analyze coded sets

A separate application was developed, the *protocol analysis evaluator*, to automate Step 2 and apply the IRRA metric. The evaluation process was separated from the simulation process to remove any dependencies. Upon successful assessment the evaluator results were transferred to JMP statistical software to be analyzed. A boxplot was used to visualize differences between simulated agreement and IRRA and the correlation was calculated to quantify the relationship.

CHAPTER 5: RESULTS

5.1 Varying code agreement

Increasing the levels of the Agreement parameter increases the average IRRA (Table 10) based on three simulated coders.

Table 10: Average IRRA versus Agreement Parameter

	Agreement Parameter					
	0	0.2	0.4	0.6	0.8	0.95
Average IRRA	0.122	0.161	0.263	0.459	0.69	0.914

The correlation coefficient (R) is 0.94 for the *Agreement* parameter and IRRA, indicating a strong linear association between the two variables (Rodgers & Nicewander, 1988). This provides support that IRRA is an effective metric for inter-rater agreement. A nonlinear region was found for *Agreement* values less than 0.4 (see Figure 4). The boxplot whiskers represent the minimum and maximum observations of IRRA for designated Agreement values. Agreement of 0.4, 0.6, and 0.8 yield a higher IRRA variance than that of very high or very low agreement.

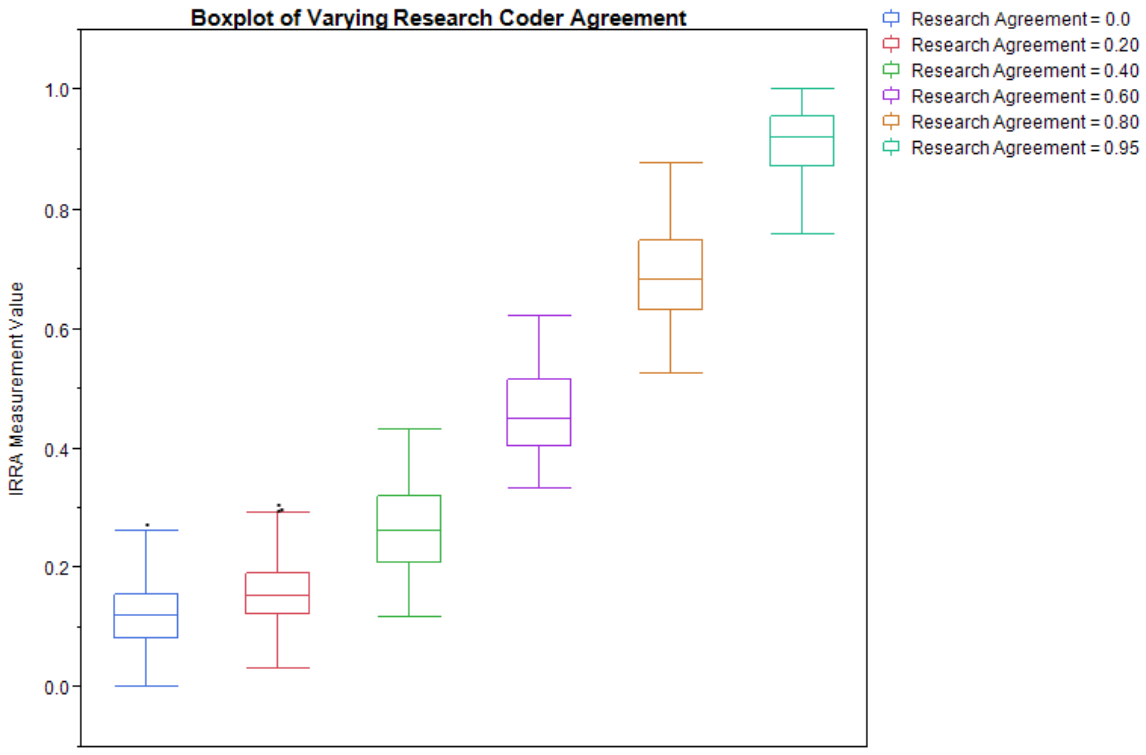


Figure 4: Boxplot of IRRA metric observations varying code agreement

The IRRA samples had a large minimum to maximum observation interval for each set of parameters which was influenced by the uniformly generated Length of Code parameter. This caused high activity sample variation which generated between 5-400 codes.

The mean was evaluated for all simulated variables represented by 300 samples each. Confidence intervals surrounding the mean do not overlap as indicated by the width of the calculated 99% confidence intervals (Table 11). The mean confidence intervals of Agreement 0.0 and 0.2 are the closest to overlapping, but still provide statistically significant results.

Table 11: Means and standard deviations of agreement levels for 300 samples each

Agreement Level	Mean	Std Dev	Lower 99%	Upper 99%
0%	0.1216	0.0521	0.1138	0.1293
20%	0.1611	0.0548	0.1529	0.1693
40%	0.2631	0.0719	0.2523	0.2738
60%	0.4587	0.0703	0.4482	0.4693
80%	0.6895	0.0788	0.6778	0.7013
95%	0.9138	0.0614	0.9046	0.923

When the Agreement parameter is 0.0, there is complete randomness amongst coders. This represents a scenario where coders are purely guessing because the data is difficult to interpret or the coders are poorly trained. The mean posted for *Agreement* = 0.0 is 0.1216 (see Table 11). However, any value up to approximately 0.30, though rare, is possible given random parameters (as seen in Figure 4). The level of agreement was used as a control for complete randomness. A similar technique was recommended in other string-editing algorithms to determine a baseline (Sellers, 1974). In this dataset, values in the range of 0.30 to 1.00 are only achieved by having some level of agreement between coders.

5.2 Varying number of coders

By varying the number of coders, we address the issue of how many coders should be used in protocol analysis. Simulations were performed with an agreement level of 0.80 and each dataset had 300 observations. As the number of coders increase, the minimum to maximum interval of IRRA measurement values decrease resulting in more reliable estimates of inter-rater agreement (see Figure 5). There are observed diminishing

returns in minimum to maximum IRRA intervals as an increase from 10 to 20 coders provides a smaller interval gain than 3 to 5 coders.

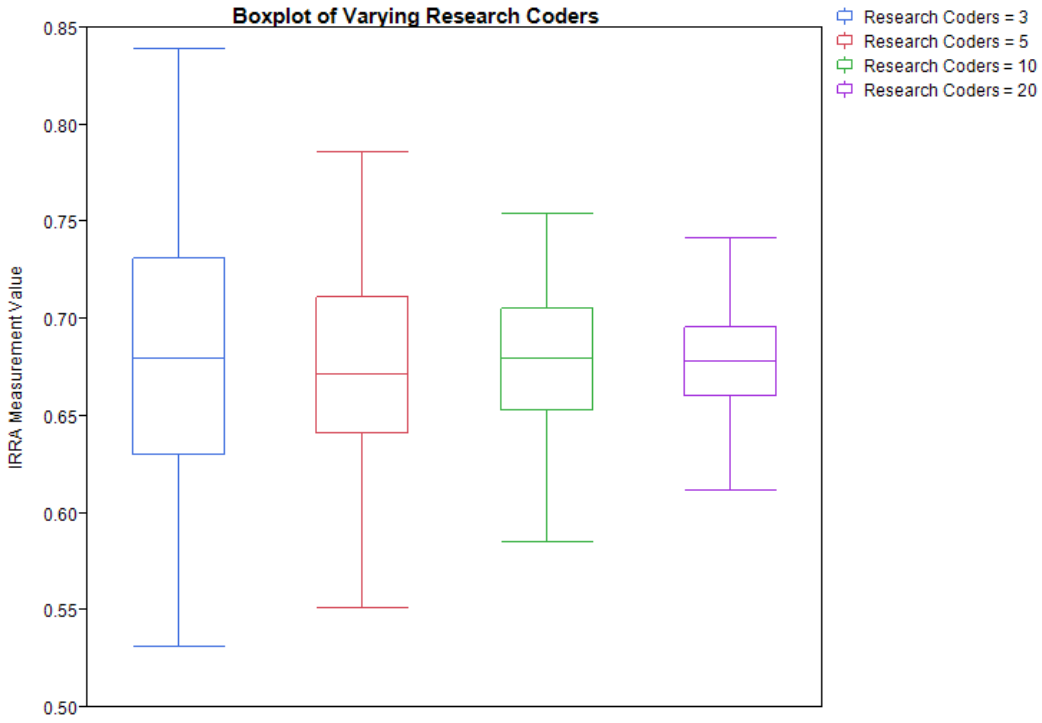


Figure 5: Boxplot of IRRA metric observations varying research coders

Additionally, there is a decrease in variation from 3 coders to 20 coders which is observed in the histogram interval (See Figures 6 & 7). It was found the IRRA metric at Agreement 80% converged around value 0.68.

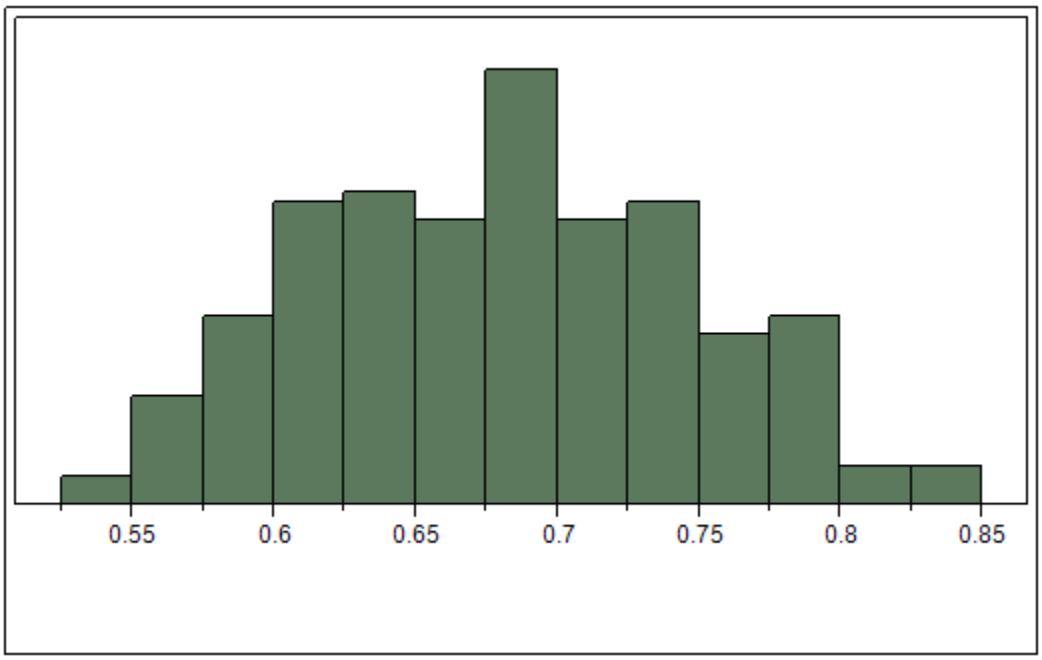


Figure 5: Histogram of 80% Agreement with 3 coder evaluations

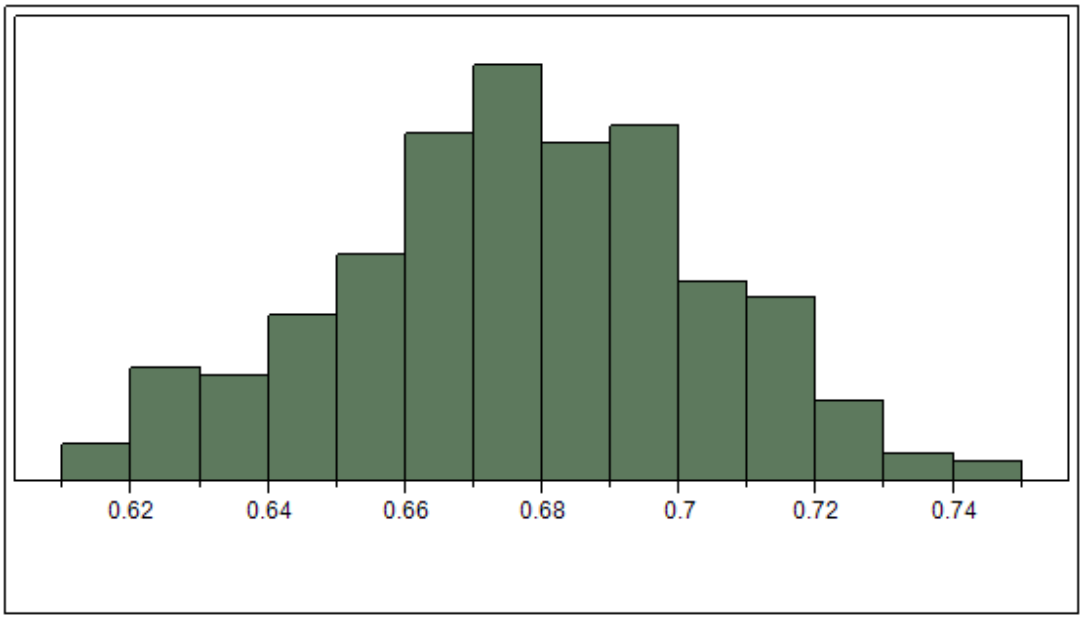


Figure 6: Histogram of 80% Agreement with 20 coder evaluations

5.3 Varying the number of activities

Dataset 3 was defined to evaluate the effects of increasing the number of activities, or codes, on IRRA. As in previous datasets, all codes are parsed by one second before applying the custom IRRA measurement in order to avoid codes of unequal code length. The results found that increasing the number of the codes has little effect on the average value (see Table 12) but decreased observed measurement variation and minimum and maximum intervals (Figure 7). The result is consistent with statistical sampling because as the number of activities increase the estimate of the standard deviation decreases. The implication in this context is that datasets with relatively few activities are unlikely to provide meaningful results unless the activities are easily interpreted. These results can provide guidance for researchers in designing their "think-aloud" scenarios so that there are sufficient activities.

Table 12: Varying Coding lengths

	Coding length (min)			
	20	40	80	160
Average IRRA Measurement	0.698	0.694	0.688	0.687

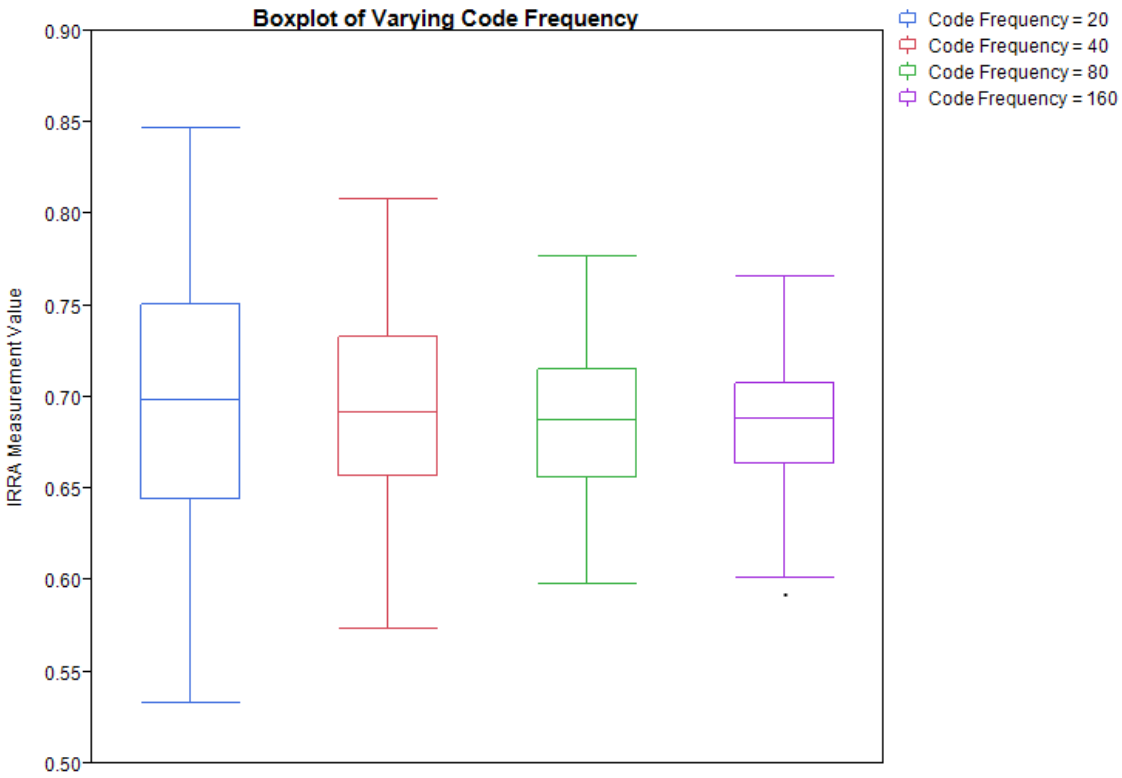


Figure 7: Boxplot of IRRA varying code frequencies

Increasing the length of a protocol analysis decreases the variation between independent observations. When protocol analysis studies increase in time the variation between coders decrease as the IRRA is able to more accurately predict a measure for agreement. Similar to number of coders there was a diminishing return on increasing the coding length. The results indicate, based on the simulator parameters, coding sets above 80 minutes long do not provide any substantial variance reduction.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

6.1 Conclusion and future work

We formulated a new inter-rater agreement metric based on previous work in string matching. The metric uses the Levenshtein Distance, an algorithm designed for determining differences in strings. In studies involving multiple coders, a high degree of agreement is necessary to make valid conclusions. The performance of the metric was evaluated using a simulator that generated coded activity sets. Codes were parsed into 1 second intervals before applying the measurement; otherwise the activities would have varying duration. We found strong correlation (R= 0.94) between simulated agreement and the IRRA metric. Agreement results in Chapter 5.1 provided suggested inter-rater intervals for future evaluations utilizing the IRRA measurement (Table 13).

Table 13: Coder agreement suggested intervals

Coder agreement	range
None	0 - 0.20
Low	0.20 - 0.40
Medium	0.40 - 0.60
High	0.60 - 0.80
Very high	0.80 - 1.00

Even though there was high correlation between agreement and IRRA, there was significant observed variance in the IRRA estimates as can be seen in the overlap of distributions (Figure 4). Increasing the number of coders and increasing the number of codes decreased the variation of IRRA.

The datasets in Chapter 3.2 that were incorrectly evaluated by Cohen's Kappa are compared to the IRRA metric (Table 14 & 15). The results indicate the custom IRRA

metric can handle all datasets available in protocol analysis as opposed to only subsection.

Table 14: Comparing Cohen's Kappa to IRRA metric example 1

Rater	Sequence	Cohen's Kappa	Custom IRRA Metric
1	AAAAABBBBBAAAAAAAAAABBBBBCCCCCCCCCC	0.38	0.91
2	AABBBBBAAAAAAAAAABBBBBCCCCCCCCCC---		

Table 15: Comparing Kohen's Kappa to IRRA metric example 2

Rater	Sequence	Cohen's Kappa	Custom IRRA Metric
1	ABCDEFG	0.00	0.86
2	BCDEFG-		

It is worth noting the custom IRRA metric, similar to other inter-rater measurements, cannot evaluate a *bad* coder. A *bad* coder is a researcher who does not know how to correctly identify the proper protocol analysis activities. The measurement evaluates the similarity, or distance, between coders evaluating activities. The possibility still exists that all coders have similar results, but all apply a similar incorrect methodology in protocol analysis surveys. One possible solution to this problem is to compare a research coder in training to that of an *expert* research coder until their results become similar enough for independent work. Another solution is to create training datasets where the behavioral actions are known and verified. Training research coders can evaluate the training sets until reaching a threshold IRRA value. Both these scenarios

are not the main purpose of the custom IRRA metric, however the measurement can provide additional uses in student to expert research.

Opportunity for future work includes refining this approach in order to determine more appropriate boundaries for coder agreement states by determining the relationship of parameters. Another opportunity of exploration is varying the parsing, or separation, timings for protocol analyses. Analyzing at lower parse times, such as one second, the algorithm favors transformational operation substitution over insertion or deletion due to quantity of data. The custom IRRA measurement may be able to provide similar results with less data than traditional methods. It is expected that increasing this interval will reduce the variance of many estimates.

A large improvement in reducing variation and increasing the ability to hierarchically cluster related sequences (Corpet, 1988) could be gained by using an algorithm with a *conversion matrix* and *penalty gap* matrices. However, additional research is required before adding these components because changing conversion matrices can have significant consequences (either good or bad) on the magnitude of the measurement (Wu, 2000). One way to calculate a conversion matrix in text matching is to have costs inversely proportional to the transition rates (Lesnard, 2006). In this methodology more common typing mistakes are less costly than uncommon blunders which allow identification of the most probably correctly spelled word to replace the incorrect word. However, this does not transfer in the same respect to inter-rater agreement. A proposed solution is to have more difficult to recognize code disagreements, such as *Monitoring their status* in place of *Evaluating their approach*

activity cost less than a more easily distinguishable action disagreement such as identifying *Drawing a figure/diagram* in place of a *Calculating* activity.

Another alternative for an inter-rater agreement metric beyond string-editing algorithms is hidden Markov chains. This approach has been used for the same sequence analysis problem domain as string-editing algorithms (Rabiner, 1989) (Hughey & Krogh, 1996). A probabilistic approach may be able to improve insights into more of the behaviors of student pathway sequences in addition to providing a reliable quantitative metric.

REFERENCES

Abbot, A. (1995). Sequence analyis: new methods for old ideas. *Annual review of socialogy*, 93-113.

Abbott, A., & Forrest, J. (1986). Optimal matching methods for historical sequences. *Journal of Interdisciplinary History*, 471-494.

Antonenko, P. D., Ogilvie, C. A., Niederhauser, D. S., Jackman, J., Kumsaikaew, P., Marathe, R. R., et al. (2011). Understanding student pathways in context-rich problems. *Education and Information Technologies*, 323-342.

Blanchard, P., Buhlmann, F., & Gauthier, J. A. (2012). Sequence Analysis in 2012. *Unpublished paper present at Lausanne Conference on Sequence Analysis*.

Brzinsky-Fay, C., & Kohler, U. (2010). New developments in sequence analysis. *Sociological Methods & Research*, 359-364.

Chi, M. T., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive science*, 121-152.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educatinoal and Psychological Measurement, 20*, 37-46.

Cohen's Kappa. (2007). Retrieved 3 23, 2013, from psych.unl.edu: http://psych.unl.edu/psycrs/handcomp/hckappa.PDF

Corpet, F. (1988). Multiple sequence alignment with hierarchical clustering. *Nucleic acids research, 16*(22), 10881-10890.

Craig, R. T. (1981). Generalization of Scott's index of intercode agreement. *Public Opinion Quarterly, 45*, 260-264.

Dijkstra, W., & Taris, T. (1995). Measuring the agreement between sequences. *Sociological methods and research, 24*(2), 214-231.

Elzinga, C. H. (2003). Sequence Similarity A Nonaligning Technique. *Sociological Methods & Research, 32*(1), 3-29.

Ericsson, K. A. (2006). The influence of experience and deliberate practice on the development of superior expert performance. *The Cambridge handbook of expertise and expert performance*, 683-703.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin, 76*, 378-382.

Gwet, K. (2001). *Handbook of inter-rater reliability*. Gaithersburg, MD: Advanced Analytics, LLC.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System technical journal, 29*(2), 147-160.

Hayes, A. F., & Krippendorff, K. (2007). Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures, 1*(1), 77-89.

Holsti, O. R. (1969). Content analysis for the social sciences and humanities. Reading, MA: Addison-Wesley.

Hughey, R., & Krogh, A. (1996). Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Computer applications in the biosciences: CABIOS, 12*(2), 95-107.

Illiopoulos, C. S., & Rahman, M. S. (2009). A new efficient algorithm for computing the logest common subsequence. *Theory of Computer Systems, 45*(2), 355-371.

Kipp, M. (2001). Anvil A Generic Annotation Tool for Multimodal Dialogue. Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech), 1367-1370.

Krippendorff, K. (1980). Content analysis: An introduction to its methodology. Beverly Hills, CA: Sage.

Kruskal, J. B. (1983). An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM review, 25*(2), 201-237.

Landis, R. J., & Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics, 33*(1), 159-174.

Lesnard, L. (2006). Optimal matching and social sciences. CREST-INSEE Work. Pap., Inst. Natl. Stat. Etudes Econ.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl., 10*, 707-710.

Mabroukeh, N. R., & Ezeife, C. I. (2010). A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR), 43*(1), Article No. 3.

MacIndoe, H., & Abbott, A. (2004). Sequence analysis and optimal matching techniques for social science data. *Handbook of data analysis*, 387-406.

Marzal, A., & Vidal, E. (1993). Computation of normalized edit distance and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 15*(9), 926-932.

Masek, W. J., & Paterson, M. S. (1980). A faster algorithm computer string edit distances. *Journal of Computer and System sciences, 20*(1), 18-31.

Myford, C. M., & Wolfe, E. W. (2009). Monitoring rater performance over time: A framework for detecting differential accuracy and differential scale category use. *Journal of Educational Measurement, 46*(4), 371-389.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys (CSUR), 46*(4), 31-88.

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology, 48*(3), 443-453.

Neter, J. (1996). *Applied linear statistical models.* Chicago: Irwin.

Pulakos, E. D. (1984). A comparison of rater training programs: Error training and accuracy training. *Journal of Applied Psychology, 69*(4), 581-588.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 257-286.

Rodgers, J. L., & Nicewander, A. (1988, Feb). Thirteen Ways to Look at the Correlation Coefficient. *The American Statistician, 42*(1), 59-66.

Scott, W. A. (1955). Reliability of content analysis: The case of nominal scale coding. *Public opinion quarterly.*

Sellers, P. H. (1974). On the theory of computation of evolutionary distances. *SIAM Journal on Applied Mathematics, 15*(1), 787-793.

Tinsley, H. E., & Weiss, D. J. (1975). Interrater reliability and agreement of subjective judgements. *Journal of Couneling Psychology, 22*, 358-376.

Wagner, R. A., & Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM (JACM), 21*(1), 168-173.

Wu, L. L. (2000). Sequence analysis and optimal matching methods in sociology: Review and prospect. *Sociological methods and research, 29*(1), 41-64.

APPENDIX A

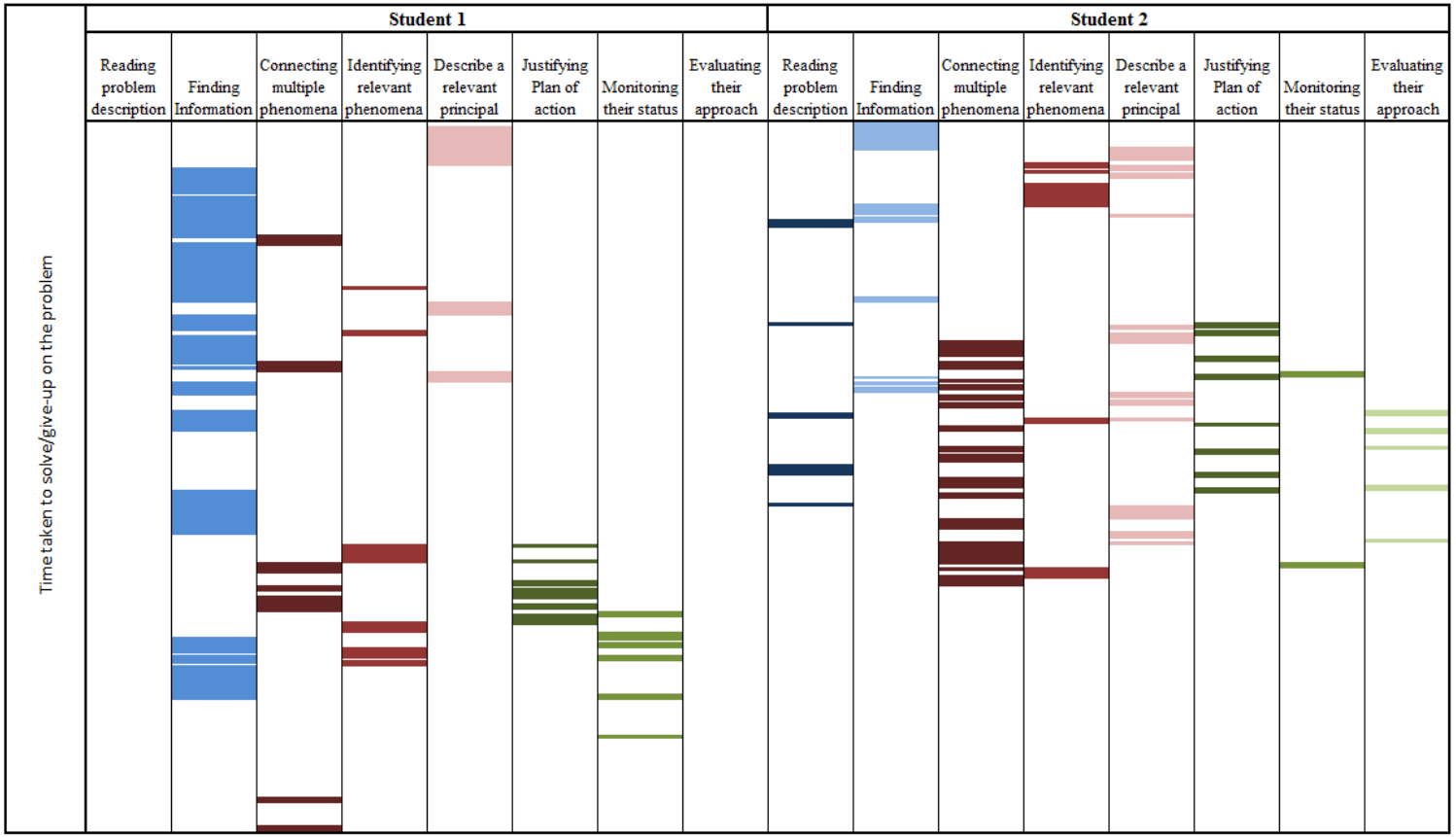


Figure A: Visualization of two independent student pathways after a protocol analysis is performed by a coder

APPENDIX B

Suppose we are given the following two strings:

String 1- **thesis**
String 2 - **oasis**

1) The first step is to start forming a table of the costs or all transformations. Here we assume a unit cost of 1 for all transformations.

2) Complete Table 1– Convert String 1: “” to String 2: “”, “o”, “oa”, “oas”, “oasi”, “oasis”

Table 1: Converting “” to “oasis”

		o	a	s	i	s
	0	1	2	3	4	5
t	1					

Table 1 explains the required steps to convert String 1: “” to String 2: “” is 0 steps as none of the characters are equal.

To convert String 1: “” to String 2: “o” is 1 step.

To convert String 1: “” to String 2: “oa” is 2 steps, etc.

3) Next, we add a row converting the cost of String 1: “t” to characters of String 2 (Table 2)

Table 2: Converting "t" to "oasis" position letters “t” and “o”

		o	a	s	i	s
	0	1	2	3	4	5
t	1	?				

Cost of converting String 1: “t” to String 2: “” in Table is obviously equal to 1. Determining the cost of converting String 1: “t” to String 2: “o” is a bit more complex as it is a minimum cost value associated with the following transformational operations.

1) Add
   - Calculate 1 plus the cost of converting “t” to “” (the left position)
   - Costs = 1 + 1 = 2

2) Remove
   - Calculate 1 plus the cost of converting “” to “o” (the above position)
   - Cost = 1 + 1 = 2

3) Change
   - Calculate 1 plus the cost of converting “” to “” (up and to the left position)
   - Cost = 0 + 1 = 1

Therefore, the minimum cost to the missing value in Table 2 is 1 which corresponds to the change operation. This cost will be added to the matrix in the missing location.

4) Continue with the next entry in the matrix

Table 3: Converting "t" to "oasis" position letters “t” and "oa”

		o	a	s	i	s
	0	1	2	3	4	5
t	1	1	?			

Convert String 1: “t” and String 2: “oa” using the minimum of the three transformational operations.

1) Add
   - Calculate 1 plus converting “t” to “o” (the left one position)
   - Cost = 1 + 1 = 2

2) Subtract
   - Calculate 1 plus converting “” to “oa” (the up one position)
   - Cost = 2 + 1 = 3

3) Change

- Calculate 1 plus converting “” to “o” (up and to the left position)
- Cost = 1 + 1 = 2

The minimum cost for the entry in Table 3 is 2 which can be added to the matrix in the missing location.

5) Continue with the next entry in the matrix

Table 4: Comparing "t" to "oasis" position letters “t” and "oas"

		o	a	s	i	s
	0	1	2	3	4	5
t	1	1	2	?		

6) Continue filling in the matrix using the same methodology

Table 5: Completed cost matrix between "thesis" and "oasis"

		o	a	s	i	s
	0	1	2	3	4	5
t	1	1	2	3	4	5
h	2	2	2	3	4	5
e	3	3	3	3	4	5
s	4	4	4	3	4	4
i	5	5	5	4	3	4
s	6	6	6	5	4	3

Each matrix includes a minimum cost path, or pathway chosen determined by minimizing the costs associated with each operation (this example all costs are 1). The minimum cost path in the cost matrix is highlighted in the completed Table 5. There may be more than one path of equal values in order to reach the minimum cost. In this example the three paths that provide the minimum costs correspond to the following word changes:

t	h	e	s	i	s
-	o	a	s	i	s

t	h	e	s	i	s
o	-	a	s	i	s

t	h	e	s	i	s
o	a	-	s	i	s

The last column in the last row of the completed cost matrix indicates the minimum distance between the two strings (aka edit distance). In this example, the edit distance is 3. The distance by this method is bound between 0 (words are the same) and 6 (completely different).