Deep Analysis of Process Model Matching Techniques

MUHAMMAD ALI1,2, KHURRAM SHAHZAD1, SYED IRTAZA MUZAFFAR1,3, AND MUHAMMAD KAMRAN MALIK1

1Punjab University College of Information Technology (PUCIT), University of the Punjab, Lahore 54590, Pakistan
2Software Development and Maintenance Centre, University of Gujrat, Gujrat 50700, Pakistan
3Faculty of Information Technology, University of Central Punjab, Lahore 54000, Pakistan

Corresponding author: Khurram Shahzad (khurram@pucit.edu.pk)

ABSTRACT Process Model Matching (PMM) aims to automatically identify corresponding activities from two process models that exhibit similar behaviors. Recognizing the diverse applications of process model matching, several techniques have been proposed in the literature. Typically, the effectiveness of these matching techniques has been evaluated using three widely used performance measures, Precision, Recall, and F1 score. In this study, we have established that the values of these three measures for each dataset do not provide deeper insights into the capabilities of the matching techniques. To that end, we have made three significant contributions. Firstly, we have enhanced four benchmark datasets by classifying their corresponding activities into three sub-types. The enhanced datasets can be used for surface-level evaluation, as well as a deeper evaluation of matching techniques. Secondly, we have conducted a systematic search of the literature to identify an extensive set of 27 matching techniques and subsequently proposed a taxonomy for these matching techniques. Finally, we have performed 432 experiments to evaluate the effectiveness of all the matching techniques, and key observations about the effectiveness of the techniques are presented.

INDEX TERMS Business process management, developing benchmarks, taxonomy, text matching.

I. INTRODUCTION

The conceptual models that depict the workflow of an organization are called business process models [1], [2]. It is abundantly established that these models are valuable assets for organizations, due to the broad range of application areas ranging from documenting software developed requirements to configuring ERP systems [3]. Process Model Matching (PMM) refers to the identification of the corresponding activities between two process models that represent the same or similar behavior [4]–[6]. The examples of the corresponding activities of two process models are: ‘Assess candidate’ - ‘Evaluate applicant’ and ‘Get birth certificate’ - ‘Collect letter of birth registration’.

The application scenarios of PMM techniques are manifolds which include computing similarity between process models, querying process models, and harmonizing variants of process models [7]–[10]. Recognizing the importance of identifying corresponding activities, several attempts have been made to develop new matching techniques, as well as to use string matching techniques for PMM [11]. Typically, the effectiveness of these techniques is evaluated in terms of three measures, Precision, Recall, and F1 scores, using benchmark datasets from the process model matching contests [11]–[13]. In this study, we have synthesized these available benchmark datasets to reveal two interrelated problems, a) the F1 score achieved by PMM is inflated due to the presence of trivial corresponding pairs, and b) the three measures merely establish the effectiveness of a PMM technique at surface-level and do not provide deeper insights into PMM techniques.

In the following subsections, firstly, we illustrate the PMM problem, secondly, we establish the limitations of the existing approaches that we have identified. Finally, a summary of the contributions is presented.

A. ILLUSTRATION OF PROCESS MODEL MATCHING PROBLEM

Figure 1 shows two process models, A and B, separated by horizontal boxes. The corresponding activities between A and B process models are highlighted in gray colour.
Where, the changes in the intensity of the colour, from light gray to dark gray, represents the varying levels of differences between the formulation of labels. More specifically, the corresponding activities having identical or similar labels are highlighted with light gray. The examples of such pairs are: ‘submit application’ - ‘submit application’ and ‘Rank student’ - ‘Ranking of student’. In contrast, the corresponding activities having the same semantics but a significantly different formulation of labels are highlighted with dark gray shade. The examples of such pairs are: ‘Check academics’ - ‘Verify education documents’ and ‘Appoint applicant’ - ‘Send offer letter’.

In this study, we contend that in the presence of such varying levels of differences in the labels of corresponding activities, it is desirable that the PMM techniques should be able to identify all types of corresponding activities.

B. ESTABLISHING THE PROBLEM

There are three publicly available datasets that have been widely used to evaluate the effectiveness of PMM techniques [4], [13]. These datasets include real-world process models from three different domains: university admissions, registration of newborns, and asset management. The datasets are formally referred to as University Admissions (UA), Birth Registration (BR), and Asset Management (AM) datasets [12]. Ontological representations of these datasets have also been generated to define an ontology mapping task, formally called OAEI’17 dataset [14].

The first two datasets are composed of 9 process models, whereas the last dataset is composed of 72 process models. Furthermore, all the three datasets include 36 pairs of process models, and collection of gold-standard annotations between 1575, 633, and 799 activity pairs. Furthermore, to the three datasets, we have also used another large dataset, Multi-Genre (MG) dataset, that is manifolds larger than UA, BR, and AM datasets. The MG dataset that we have used in this study is composed of 600 process models, 406 process model pairs, and gold-standard annotation between 89558 activity pairs.

For this study, we have synthesized all the four datasets by re-assessing their gold-standard annotations. For that, two researchers having expertise in natural language processing were involved in the re-assessment process. Both researchers independently reviewed each corresponding pair in the four datasets and reached the same conclusion that all these datasets contain a substantial amount of corresponding pairs that are either identical or similar i.e. having a slight variation in the formulation of the labels. The results of our synthesis are presented in Table 1. In the table, the corresponding pairs having identical labels or slight variations in the formulation of labels are represented as trivial corresponding pairs. Based-on the results, we have identified the following two interrelated issues regarding the evaluation of PMM techniques.

1) INFLATED SCORES

All the four datasets include a substantial amount of trivial corresponding pairs, which artificially inflates the Precision, Recall, and F1 scores achieved by a matching technique. Therefore, any value of the evaluation measures achieved by the complete datasets does not correctly represent the effectiveness of a PMM technique.
II. ENHANCED BENCHMARK DATASETS

This section presents our first contribution, enhanced benchmark datasets which include classification of corresponding pairs into three sub-types. In particular, firstly, we discuss the conceptual bases that we have used for enhancing the datasets. Subsequently, the details of our enhancements, as well as the specifications of the enhanced datasets are presented.

A. CONCEPTUAL BASES FOR ENHANCING DATASETS

It is widely acknowledged that the level of similarity between a given pair of strings can be defined in terms of three types: Near Copy, Light Revision, and Heavy Revision [16]–[18]. Where, Near Copy represents a slight variation in the formulation of two strings in the pair by changing the form of a word or adding stop words. An example pair of this type of pair is ‘best student’ - ‘the best student’. Light Revision represents substantial variation in the formulation of two strings, which can be achieved by adding new words or replacing words with their synonyms. ‘Identify best student’ is a light revision of ‘find the outstanding student’. Heavy Revision represents significant variation in the phrasing of the two strings in such a way that the semantics are not changed. The change operations may involve the use of alternative words or reordering existing words. Examples of heavy revision of an activity label ‘find the best student’ are ‘perform class topper search’ and ‘find the student having the highest GPA in the class’.

B. ENHANCING THE BENCHMARK DATASETS

This study has used the conceptual bases presented above to enhance the four benchmark datasets. To that end, firstly, the three types of pairs that we have used, are introduced, which is followed by a discussion about the pre-processing of the datasets that we have performed. Finally, the procedure of enhancements and the specifications of the enhanced datasets, are presented.

1) DIVERSITY IN CORRESPONDING PAIRS

This study has used three types of pairs, Verbatim, Modified Copy, and Heavy Revision, to represent the diversity in the corresponding pairs. These types stem from the levels of string similarity discussed in the preceding subsection. Below, we provide an overview of our three types of pairs and our criteria for deciding the category of a given pair. Note, the criteria are developed based on our synthesis of the four datasets keeping in view the definitions of NC, LR, and HR. The criteria of each type of pair, as well as their respective examples from the benchmark datasets, are presented in Table 2.

- **Verbatim (VB)**: Two activities in a pair are verbatim of each other if their labels are either identical or almost the same. In particular, a corresponding pair is verbatim if the two labels are identical labels, identical labels without stop words, or re-ordered pairs.
Table 2. Criteria and examples of declaring a pair VB, MC, and HR [15].

| Pair Type       | Criteria                        | Example Pair                        |
|-----------------|---------------------------------|-------------------------------------|
| Verbatim        | Identical labels                | Create birth certificate            |
|                 | Identical labels without words  | Receive notification birth          |
|                 | Reordered, but Identical words  | Check nationality of parents        |
| Modified Copy   | Adding/deleting a few words     | Register child                      |
|                 | Replacing synonyms              | Confirm identity                    |
|                 | Switching labeling style        | Register child                      |
| Heavy Revision  | Substantially revised labels    | Receive documents                   |
|                 | Subsume the other label         | Register child                      |

- **Modified Copy (MC):** The participating activities in a pair are modified copy of each other if the formulation of labels is substantially different from each other but their semantics are the same. This category stems from the notion of Light Revision. The criteria used for considering an activity pair as a modified copy if it involves additional words in a label, use of synonyms, or switching labels style.

- **Heavy Revision (HR):** The participating activities in a pair are the heavy revision of each other if the labels are significantly different from each other or one label subsumes the other label. The criteria used for this type of pair and their corresponding examples are presented in Table 2.

2) PRE-PROCESSING DATASETS

As a first formal step towards enhancing the benchmark datasets, we also analyzed the publicly available results,\(^1\) as well as the gold-standard annotations,\(^2\) of all the twelve matching systems from the matching contest. The analysis revealed the following three discrepancies:

- **UA dataset contains at least 188 corresponding activity pairs that are not meaningful as they include logical labels that do not belong to an activity. An example of such an activity label is ‘IntermediateCatchEvent’ - ‘ExclusiveGateway’.

- **All the four datasets (UA, BR, AM, MG) contain activity pairs that have identical or similar labels but they are declared as unequivalent in the benchmark datasets. The four datasets includes 13, 42, 213, and 1704 such pairs.**

- **Each dataset contains duplicate pairs which are likely to skew the overall performance scores, as well as scores of each type of pair, after classification.**

To overcome the discrepancies, firstly, the meaningless pairs were removed from the UA dataset. Secondly, The identical or similar labels that were declared as unequivalent were marked as equivalent pairs. That is, 13, 42, 213, and 1704 pairs were marked in the UA, BR, AM, and MG datasets, respectively. Finally, 189, 164, 78, and 789 duplicate pairs were omitted from the four datasets leaving 171, 259, 378 and 7358 corresponding pairs in the four datasets.

3) CLASSIFYING CORRESPONDING PAIRS

The corresponding pairs in the pre-processed datasets were used for the classification task. That is, each corresponding pair was classified as VB, MC, or HR, based on the criteria presented in Table 2. In particular, two experts independently classified all the corresponding pairs into one of the three types, using the developed criteria. Subsequently, all the conflicts resolved by discussing each pair and developing consensus about its type. Finally, each dataset was enhanced by annotating the respective concurred type of each pair.

The distribution of the types of pairs in each dataset is presented in Table 3. A key observation from the table is that the number of pairs in each type is significantly different. For instance, the number of HR pairs in UA datasets is more than twice the number of VB and MC pairs. This lack of uniformity, in terms of the number of pairs of a specific type, reinforces that the use of overall effectiveness scores merely provides a surface level evaluation of a matching technique, and it does not provide any deeper insights about the capabilities of a matching technique. Hence, this study computes the overall effectiveness scores, as well as the effectiveness score of each type of pair.

---

1. The results can be downloaded from [https://ai.wu.ac.at/emisa2015/contest.php](https://ai.wu.ac.at/emisa2015/contest.php)
2. Gold standard refers to the benchmark correspondences generated by BPM experts
TABLE 3. Types of corresponding pairs in the four datasets.

| Datasets | VB Pairs | MC Pairs | HI Pairs | Total |
|----------|----------|----------|----------|-------|
| UA       | 33       | 28       | 110      | 171   |
| BR       | 46       | 62       | 151      | 259   |
| AM       | 255      | 24       | 99       | 378   |
| MG       | 812      | 2540     | 4006     | 7358  |
| Total    | 1146     | 2654     | 4366     | 8166  |

III. TAXONOMY OF THE MATCHING TECHNIQUES

In this section, we present our second contribution which includes identifying a comprehensive set of matching techniques and building a taxonomy of the process model matching techniques that exist in literature. Additionally, an overview of each matching technique is presented in this section, hence, providing a unified and comprehensive source of knowledge for the research community.

A. SEARCH STRATEGY

Starting from a survey of process model matching techniques, we used the snowballing technique to identify existing studies. In particular, we started with a notable study [5] and applied backward and forward tracing techniques to identify relevant studies. As a result of the backward tracing, we found twenty six studies that have discussed one or more similarity measures. Whereas, as a result of the forward tracing, we initially collected seven research studies from which five studies either did not discuss any similarity measure or they discussed one additional similarity measure is already collected during the backward tracing step. At this step, the latest studies which relied on the use of three state-of-the-art text similarity techniques, named word embeddings, for process model matching. Accordingly, we have identified twenty seven similarity measures that are included in this study. Note that we only considered research articles published in journal and proceedings of international conferences and workshops. The developed taxonomy is presented in the following subsection.

B. THE TAXONOMY

In this study, we have included all the twenty seven similarity techniques and designed a taxonomy for these measures. The taxonomy is composed of more abstract categories, as well as specific categories, based on the commonalities between the techniques. Figure 2 shows the first level of our taxonomy.

It can be observed from the figure that there are three categories of similarity techniques: syntactic, semantic, and word embeddings. Where, the syntactic techniques compute the similarity between a label pair based on the exact content of the involved labels. These are very basic measures that even do not consider simple relations between words of labels, such as inflected form, inclusion or exclusion of stop-words, synonym, etc. In contrast, the semantic similarity measures take into consideration the different forms of words and relationships between words for computing similarity between labels. The third category, word embeddings, includes a set of state-of-the-art techniques that go beyond synonyms and considers the context of words during the comparison of activity labels in a pair.

1) SYNTACTIC SIMILARITY

The syntactic similarity measures are further divided into two categories: word-based similarity measures and string-based similarity measures. The word-based similarity measures compare the two labels in a pair based on individual words that exist within both labels. Whereas, the string-based similarity measures compare label pair by considering the complete label as a string and then using string matching operations to compute the similarity between the two labels. Word-based similarity measures are further divided into two categories named as overlap based similarity measures and vector-based similarity measures. The overlap based similarity measures compare the label pair on the bases of the common words between both labels. Whereas, vector-based similarity measures firstly generate the word vectors for both labels either based on term frequency or based on term frequency-inverse document frequency. Subsequently, cosine between these two vectors is calculated to decide the similarity of both labels. String-based similarity measures are also divided into two categories that are distance-based similarity measures and substring-based similarity measures. If any similarity measure computes the similarity of label pair by observing that one label is part of other labels or not it is called sub-string based similarity measure. In contrast, distance-based similarity measures compare both labels based on different types of distances between both labels, such as edit distance and hamming distance, etc.

2) SEMANTIC SIMILARITY

The semantic similarity measures are divided into two categories: word similarity based similarity measures and word
relatedness based similarity measures. Word-similarity based similarity measures compares the label pair by also considering the synonyms of both labels’ words. While word relatedness based similarity measures consider other relations between words, such as Lin, Lesk, etc., whereas word similarity based similarity measure considers synonyms of individual words. These measures are further divided into two categories based on the weights assignment to identical words and synonyms words. Weighted similarity measures assign weights to identical words and synonyms words that exist between both labels, whereas, non-weighted similarity measures do not assign weights to different types of word pairs.

As shown in Figure 4, word relatedness based similarity measures are further divided into two categories which are individual and ensembled. Those similarity measures which only use single word relatedness measures are placed into an individual categories, such as Lin, Lesk, etc.

C. THE MATCHING TECHNIQUES
We have selected 27 similarity measures that are previously used in the process model matching literature. A brief overview of each similarity measure is as follows:

1) JACCARD SIMILARITY
Jaccard similarity computes the similarity between labels based on the number of words that are common between them. Formally, Jaccard similarity is defined as follows:

\[ S_{\text{jaccard}}(L_1, L_2) = \frac{W_{\text{com}}(L_1, L_2)}{|L_1| + |L_2| - W_{\text{com}}(L_1, L_2)} \]

where, \( L_1 \) and \( L_2 \) are the two labels, \( |L_n| \) is the length of the \( n_{th} \) label, and \( W_{\text{com}} \) is the number of words of the first label that is also present in the second label. This similarity measure returns similarity score between 0 and 1, where 1 represents that both labels are identical, and 0 represents that both labels are entirely different. For instance, consider two labels, “conduct oral exam” and “conduct interview”. There is only one word “conductor” in the first label that is also available in the second label. Therefore, Jaccard similarity between the labels becomes:

\[ \text{Sim}(L_1, L_2) = \frac{1}{3 + 2 - 1} = 0.25 \]

2) DICE SIMILARITY
Dice similarity is a variant of Jaccard similarity that also computes the similarity between a label pair based on the
common words between the two labels. However, it computes the similarity of both labels based on the number of words that are common between them. Formally, Dice similarity is defined as follows:

\[
Sim_{dice}(L_1, L_2) = \frac{2 \times |W_{com}(L_1, L_2)|}{|L_1| + |L_2|}
\]

where, \(L_1\) and \(L_2\) are the two labels and \(|L_n|\) is the length of the \(n_{th}\) label, in terms of number of words. Furthermore, \(W_{com}\) is the count of common words from the first label that can also be found in the second label. This similarity measure returns similarity score between 0 and 1, where 1 represents that both labels are identical, whereas 0 represents that both labels are entirely different. For the two labels, “conduct oral exam” and “conduct interview”, the Dice similarity becomes

\[
Sim(\text{conductoralexam}, \text{conductinterview}) = \frac{2 \times 1}{3 + 5} = 0.4
\]

3) COSINE TF
To compute cosine similarity, first labels vectors should be generated. There are different ways to generate the labels vectors; Term Frequency is the simplest one. Here word count in a label is used as term frequency. All unique words of both labels are used as a corpus for vectors where words are in an ordered form. After the successful generation of labels vectors, cosine similarity is calculated based on the equation that is given below.

\[
similar(L_1, L_2) = \frac{\sum_{i=1}^{n}(L_{1i} \times L_{2i})}{\sqrt{\sum_{i=1}^{n}(L_{1i})^2} \times \sqrt{\sum_{i=1}^{n}(L_{2i})^2}}
\]

where \(L\) is used for Label representation. The above equation returns similarity value between 0 and 1, where 1 means both labels are identical, and 0 means both labels are completely dissimilar. For example, “conduct oral exam” and “conduct interview” are two candidate labels. Therefore, Cosine similarity value with Term Frequency is \((1) = \frac{1}{3+5} = \frac{1}{5} = 0.2\)

4) COSINE TF/IDF
Cosine TF/IDF differs from cosine TF. The equation for the computation of Cosine similarity with TF/IDF is identical to Cosine similarity with term frequency. However, the difference lies in the way label vectors are generated, i.e. it generates vectors based on TF-IDF, instead of Term Frequency. TF/IDF discredit common words that are present in both labels. For example, if “conduct oral exam” and “conduct interview” are two candidate labels, the Cosine similarity value with TF/IDF becomes \(\frac{0}{0.181+0.091} = \frac{0}{0.272} = 0\)

5) SUBSTRING
Two labels are similar if they are identical or one label is the substring of the other label. This binary decision means either the two labels are similar or dissimilar. For example, “conduct oral exam” and “conduct interview” are dissimilar labels as per the above criteria. Furthermore, in the example pair “conduct interview” and “conduct interview with panel”, these both labels are similar as one is a complete substring of the other label.

6) EDIT DISTANCE 17
There are several variations to compute similarity using Edit Distance. Given below equation is mostly used variation of Edit Distance to calculate the similarity between two labels. According to the given below equation value of edit distance is normalized with string length of larger label.

\[
similar(L_1, L_2) = 1 - \frac{ed(L_1, L_2)}{\max(|L_1|, |L_2|)}
\]

Edit Distance is also known as Levenshtein Distance. Edit Distance of two labels \(L_1\) and \(L_2\) is the required the minimal number of atomic string operations to transform one label to another. The atomic operations are insertion, deletion, and substitution of a character. The above equation returns a similarity score between 0 and 1, where 1 means both labels are identical and 0 means both labels are completely dissimilar. For example, “documentation” and “document” are two labels for which five atomic operations are required to transform one label into the other, therefore the similarity score becomes \(1 - \frac{5}{7} = 0.615\).

7) EDIT DISTANCE 16
The equation given below is another variant of Edit Distance to calculate the similarity between two labels. According to the technique, the value of edit distance is normalized by the sum of both labels’ length.

\[
similar(L_1, L_2) = 1 - \frac{ed(L_1, L_2)}{|L_1| + |L_2|}
\]

The above equation returns a similarity score between 0 and 1, where 1 means both labels are identical and 0 means both labels are completely dissimilar. For example, “documentation” and “document” are two labels, the five atomic operations are required to transform one label into the other. The similarity score then becomes \(1 - \frac{5}{7} = 0.761\).

8) EDIT DISTANCE 15
The equation given below is another variant of Edit Distance to calculate the similarity between two labels. According to the given below equation value of edit distance is weighted with the length of shortest label.

\[
similar(L_1, L_2) = \max \left(0, \frac{\min(|L_1|, |L_2|) - ed(L_1, L_2)}{\min(|L_1|, |L_2|)} \right)
\]

This equation returns a similarity score between 0 and 1, where 1 means both labels are identical and 0 means both labels are completely dissimilar. For example, “documentation” and “document” are two labels having edit distance of five. The similarity score then becomes \(\frac{3-5}{8} = \frac{-2}{8} = 0.375\).

9) HAMMING DISTANCE
Earlier, we discussed the three similarity techniques that are based on Edit Distance. This technique based on the Hamming Distance. The equation given below is used to calculate
the similarity between two labels based on their Hamming Distance. According to the technique, Hamming distance is normalized with string length of the larger label.

\[
similar(L_1, L_2) = 1 - \frac{hd(L_1, L_2)}{\max(|L_1|, |L_2|)}
\]

 Typically, Hamming distance is calculated on equal sized labels, however activity labels may not be of comparable size, therefore, Hamming distance is extended to extra lengths of the larger label. Hamming distance is calculated by inspecting every index of both labels. If characters of both labels at any specific index are not the same then the Hamming distance is incremented by one. The above equation returns similarity value between 0 and 1, where 1 means both labels are identical and 0 means both labels are completely dissimilar. For example, consider the two labels “documentation” and “document”, the Hamming distance of these labels is five. The similarity score therefore becomes \( 1 - \frac{5}{13} = 0.615 \).

10) LONGEST COMMON SUBSEQUENCE

LCS finds the longest common subsequence from two labels instead of substrings. As we know substring should be in consecutive indexes in the label, but characters in common subsequence might exist at any index. Below mentioned first equation is used to calculate LCS Distance.

\[
lcsDistance(L_1, L_2) = |L_1| + |L_2| - 2 * lcs(L_1, L_2)
\]

LCS Distance is similar to Edit Distance if Edit Distance allows only insertion and deletion operation or considers the cost of substitute operation as double. The equation given below is used to calculate the similarity between two labels based on their LCS Distance. According to the equation, the value of LCS distance is normalized by the sum of both labels’ length.

\[
similar(L_1, L_2) = 1 - \frac{lcsDistance(L_1, L_2)}{\sum(|L_1|, |L_2|)}.
\]

The above equation returns a similarity score between 0 and 1, where, 1 means both labels are identical and 0 means both labels are completely dissimilar. For example, consider the two labels “Send letter of acceptance” and “Send acceptance”, the LCS distance of the above mentioned labels is ten, therefore, the similarity value becomes \( 1 - \frac{10}{40} = 0.75 \).

11) JARO SIMILARITY

Jaro similarity is considered as the best syntactic similarity technique for shorter labels. As compared to Edit Distance, the Jaro similarity uses the concept of common characters between label pairs. Consider two labels \( L_1 \) and \( L_2 \) having length of \( n \) and \( m \), respectively. Suppose \( i \) is used as index for \( L_1 \) and \( j \) is used as index for \( L_2 \), at some index \( i \) in \( L_1 \) and \( j \) in \( L_2 \) the character is same, such as \( L_{1i} = L_{2j} \), it is a common character if and only if the two characters are not far enough or at least not far from the half length of maximum length label. Formally, it can be written as follows:

\[
j - i \leq \frac{\max(n, m)}{2} - 1.
\]

The above mentioned technique is used to find the common characters between label \( L_1 \) and label \( L_2 \), and it is denoted by \( cc(L_1, L_2) \). As a result, we get an ordered vector of common characters for \( L_1 \) by with comparing \( L_2 \), we could name this vector as \( L_1' \). Similarly, a ordered vector of common characters between \( L_2 \) and \( L_1 \) is represented as \( L_2' \). Formally, Jaro similarity is defined as follows:

\[
similar(L_1, L_2) = \frac{1}{5} \left( A + A + A - t(L_1, L_2) \right)
\]

where

\[
A = cc(L_1, L_2)
\]

In the above equation, \( t(L_1, L_2) \) is the transposition number between \( L_1 \) and \( L_2 \) which is calculated by the half of hamming distance of \( L_1' \) and \( L_2' \) vectors. The above equation returns a similarity score between 0 and 1, where 1 means both labels are identical and 0 means both labels are completely dissimilar. For example, “Send letter of acceptance” and “Send acceptance” are two labels, the Jaro similarity value for the above mentioned labels becomes 0.76.

12) JARO SIMILARITY VARIANT

This technique is a variant of Jaro similarity with small changes. Formally, it is defined as follows:

\[
similar(L_1, L_2) = \frac{1}{4} \left( A + B + C + D \right)
\]

where,

\[
A = \frac{cc(L_1, L_2)}{n}
\]

\[
B = \frac{cc(L_2, L_1)}{m}
\]

\[
C = \frac{cc(L_1, L_2) - t(L_1, L_2)}{cc(L_1, L_2)}
\]

\[
D = \frac{cc(L_2, L_1) - t(L_1, L_2)}{cc(L_2, L_1)}
\]

This technique also uses common characters in opposite direction because of the fact that \( cc(L_1, L_2) \) is not equal to \( cc(L_2, L_1) \). The above equation returns a similarity score between 0 and 1, where 1 means both labels are identical and 0 means both labels are completely dissimilar. For the two labels “Send letter of acceptance” and “Send acceptance”, the similarity score of the labels become 0.73.

13) MEAN OF STATIC WEIGHTED WORD COMPARISON

This technique assigns 1.0 weight to common words in both labels and assigns 0.75 weight to those uncommon words pairs that are synonyms of each other. Subsequently, the score is normalized by the length of labels that has maximum words. As this technique assign a higher value to common words and lower to synonymous words, therefore, this technique favors the common words. Formally, it is defined as follows:

\[
similar(L_1, L_2) = \frac{1.0 * A + 0.75 * B}{\max(|L_1|, |L_2|)}
\]
where,

\[ A = |L_1 \cap L_2| \]
\[ B = \sum_{(a,b) \in L_1 \times L_2 \setminus L_1} \text{synonym}(a,b) \]

where, \(|L_1 \cap L_2|\) is the count of common words in both labels. Where \text{synonym} function checks whether the pair of words from uncommon words of both labels are synonym to each other. If this is the case then this function returns 1 otherwise return 0. For word synonyms, WordNet dictionary is used.

14) MEAN OF DOUBLED WEIGHTED WORD COMPARISON
The formal specification of the techniques is given below in the equation. In the equation, \(w_1\) and \(w_2\) are weights for common words and synonym words, respectively. The values for these weights could be dynamically assigned between 0 and 1, but 1 and 0.75 values are recommended for common words and synonym words, respectively. This technique assigns 1.0 weight to common words in both labels and assigns 0.75 weight to those uncommon words pairs that are synonym of each other. \(w_1\) is doubled by multiplying it with 2 and \(w_2\) is doubled by counting synonyms for both labels instead of synonyms. The equation for this technique is as follows.

\[ \text{similar}(L_1, L_2) = \frac{2.0 \cdot w_1 \cdot A + w_2 \cdot (B + C)}{|L_1| + |L_2|} \]

where,

\[ A = |L_1 \cap L_2| \]
\[ B = \sum_{(a,b) \in L_1 \times L_2 \setminus L_1} \text{synonym}(a,b) \]
\[ C = \sum_{(a,b) \in L_2 \times L_1 \setminus L_2} \text{synonym}(a,b) \]

In the equations, \(|L_1 \cap L_2|\) is the count of common words in both labels. Where, \text{synonym} function checks whether the pair of words from uncommon words of both labels are synonyms of each other. If this is the case, then this function returns 1 otherwise it returns 0.

15) MONGE ELKAN
Monge Elkan technique takes a word from the first label and compare it with all words of the second label but only records maximum value from these comparisons. The procedure is repeated for all words of the first label. The resultant values are summed and normalized with the total number of words in the first label. Formally, it is defined as follows:

\[ \text{similar}(L_1, L_2) = \sum_{a \in L_1} \max_{b \in L_2} \text{WordSimilarity}(a, b) \]

In the equation, the \text{WordSimilarity} function return one if both words “a” and “b” are identical. In case both words are synonyms, a predefined value is returned. This value is less than one and greater than zero, however, typically, 0.5 or 0.75 values are used. In case both words are not identical nor synonyms, then this function returns zero.

16) MONGE ELKAN LIN
Monge Elkan technique takes a word from the first label and compares it with all words of the second label but only records maximum value from these comparisons. The procedure is repeated for all words of the first label. The resulting values are summed and normalized by the total number of words in the first label. Formally, it is defined as follows:

\[ \text{similar}(L_1, L_2) = \frac{\sum_{a \in L_1} \max_{b \in L_2} \text{Lin}(a, b)}{|L_1|} \]

As compared to the earlier technique, this technique computes the Lin similarity value between two words instead of synonym similarity.

17) DICE WITH SYNONYMS
It’s a syntactic technique which considers only common words in both labels. This technique also considers the synonyms. The equation for this technique is as follows.

\[ \text{similar}(L_1, L_2) = \frac{2 \times \text{SCW}(L_1, L_2)}{\text{WC}(L_1) + \text{WC}(L_2)} \]

where, \(L\) is used for Label representation and \(\text{WC}(L)\) means Word Count in the specific label. Additionally, \(\text{SCW}\) is the count of words from the first label that have common or synonym in the second label.

18) LIN WITH GREEDY PAIRS
Lin value of the label pair is calculated by averaging of Lin value of words pairs that are selected by greedy pairing. Greedy pairing works as Lin value is computed for each word of the first label with all words of the second label and chooses the maximum Lin value for each word of the first label. Then these values are multiplied by their respective IDF values. After that, all these values are added and then normalized with the word count of the first label. The same steps are repeated for the second label. Finally, both values are averaged and resultant value is the Lin similarity of both labels. Furthermore, word pair Lin value is based on the Information Content of LCS (Least Common Subsumer) divided by the sum of Information Content of both synsets of words. Formally, it is defined as follows:

\[ \frac{2 \times \text{IC}(lcs)}{\text{IC}(s_1) + \text{IC}(s_2)} \]

19) LIN WITH OPTIMAL PAIRS
This technique also computes Lin similarity between two candidate labels by employing a similar approach. However, this technique makes the words pair by the optimal way to gain the overall maximum Lin similarity value for both labels as compared to Greedy pairing where pair is selected based on the maximum Lin value. The Lin value of the label pair is calculated by averaging Lin value of words pairs that
are selected by optimal nature. Furthermore, it is based on the Information Content of LCS (Least Common Subsumer) divided by the sum of Information Content of both synsets of words. Formally, it is defined as follows:

\[
2 \ast \frac{IC(lcs)}{IC(s1) + IC(s2)}
\]

20) LESK WITH GREEDY PAIRS

LESK value of a label pair is calculated by averaging of LESK value of word pairs that are selected by greedy pairing. Similar to the greedy pairing of Lin, each word of the first label with all words of the second label and chooses the maximum LESK value for each word of the first label. Then these values are multiplied by their respective IDF values. After that, all these values are added and then normalized with the word count of the first label. The same steps are repeated for the second label. In the end, both values are averaged and resultant value is computed.

21) LESK WITH OPTIMAL PAIRS

This technique also computes LESK similarity between two candidate labels. The difference is that this techniques employs optimizing pairing to gain the overall maximum LESK similarity value for both labels as compared to Greedy pairing, where the pair is selected based on maximum LESK value. The LESK value of label pair is calculated by averaging LESK value of words pairs that are selected by optimal pairing.

22) PATH SIMILARITY

This technique does not consider the common words in both labels, because the inclusion of common word increases the similarity of both labels and could influence the overall similarity decision. This technique calculates path similarity using the WordNet dictionary and normalizes this score by the count of words of the first label that are uncommon with words of the second label, and path similarity is based on shortest path between both words in the WordNet taxonomy. Formally, this technique is defined as follows:

\[
similar(L_1, L_2) = \frac{\sum_{w_1 \in L_1 \setminus L_2} \max(\partial(w_1, w_2)|w_2 \in L_2 \setminus L_1)}{|L_1 \setminus L_2|}
\]

where, \(\partial(w_1, w_2)\) is path similarity value of words pair \(w_1\) and \(w_2\) from the WordNet dictionary.

23) INTERSECTION OF SENS

This technique only considers the uncommon words of both labels. Common words of both labels ignored by this technique. Then for each label, a synonym set is maintained that contains all the WordNet Synonym Sets of uncommon words of that Label. After that, the length of intersection of these two sets is divided by the length of the maximum set. Equation for this technique is as follows.

\[
similar(L_1, L_2) = \frac{|\partial(L_1) \cap \partial(L_2)|}{\max(|\partial(L_1)|, |\partial(L_2)|)}
\]

where \(\partial(L_i)\) is Synonym Set of all uncommon words of the first Label.

24) LIN OF SEMANTIC COMPONENTS

This technique extracts three semantic components (action, business object, additional information) from both labels and subsequently Lin value is calculated for each component of both labels. Finally, these values are summed up and the maximum number of semantic components in any label normalizes resultant value. Formally, it is defined as follows:

\[
similar(L_1, L_2) = \frac{A + B + C}{\max(|L_1|, |L_2|)}
\]

where,

\[
A = Lin_a(L_1, L_2) \\
B = Lin_{bo}(L_1, L_2) \\
C = Lin_{add}(L_1, L_2)
\]

In the equation, \(Lin_a(L_1, L_2)\) calculates the Lin value between action components of both labels. If any label does not have a component, then 0 is returned for that specific component instead of Lin value. Here, \(a, bo, add\) are used as abbreviations for action, business object, additional information components. Whereas, \(|L_1|\) means the number of semantic components that are present in label \(L_1\). This technique assumes the action semantic component must be present in both labels.

25) WORD EMBEDDINGS

In addition to the above, we have also used three state-of-the-art representations of words, called word embeddings. In particular, we have used three established techniques for generating word-embeddings, Word2vec, fastText, and GloVe. Where, Word2vec embeddings were proposed by Google Inc. [19], fastText embeddings were proposed by Facebook Inc. [20], and the GloVe embeddings were proposed by the leading research group at Stanford University [21]. Among these techniques, Word2vec and fastText are based on skipgram with the negative-sampling training method, whereas, GloVe is based on global word-to-word co-occurrence statistics.

In the subsequent sections, a comprehensive evaluation of these measures is presented.

IV. COMPREHENSIVE EVALUATION OF TECHNIQUES

This section presents our third contribution, which is a comprehensive evaluation of all the similarity techniques identified in the preceding section. Below, we introduce the evaluation measures and experimental setup used for evaluating the similarity techniques. Subsequently, a comprehensive analysis of the results is presented.

A. EVALUATION MEASURES

The three widely established measures to evaluate the effectiveness of matching techniques are Precision, Recall, and
F1 scores. Where, Precision is the ratio between true positive pairs and true positive and false positive pairs [22], Recall is the ratio between true positive pairs and true positive and false negative pairs, and F1 score is the harmonic mean of precision and recall.

\[ F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \]

**B. EXPERIMENTATION**

In this study, we have performed 432 matching experiments. That is, experiments are performed using 27 matching techniques for all the four datasets (UA, BR, AM, MG) to generate overall performance scores, as well as for the three types of pairs (VB, MC, and HR). For the experimentation, all the matching techniques were implemented in such a way that each implementation takes input an activity pair and returns a normalized similarity score between 0 and 1. Subsequently, each similarity score is converted into a binary decision of equivalent and non-equivalent represented by 1 and 0, respectively. These binary decisions are used to compute Precision, Recall, and F1 score. Note, for the overall results, the values of P, R, and F1 were computed. On the contrary, for VB, MC, and HR pairs only Recall scores were computed, due to the absence of un-equivalent pairs.

Several matching techniques involve computing semantic similarity between words. In this study, we have used WordNet to determine the semantic similarity between words from two labels. WordNet is a well-established and widely used lexical database for the English language that is composed of over 150,000 concepts, including verbs, adverbs, nouns, and adjectives [23]. The concepts are related with other concepts in the database, called synsets [24], [25]. Furthermore, the concepts can have additional relations, such as is-a and part-of relationships, to form a network of concepts.

**C. RESULTS AND ANALYSIS**

The results of all the experiments performed using the 27 techniques and four datasets are presented in Table 4-7. In particular, each table includes overall Precision, Recall, and F1 scores of all the techniques that provides a surface level evaluation of matching techniques. Furthermore, the Recall scores of each type of pair: Verbatim, Modified Copy, and Heavy Revision, provides essential insights about the capabilities of each matching technique. Based on the results presented in Tables 4 - 7, we present the following three key observations: comparison of matching techniques, comparison of pair types, and comparison of datasets. Subsequently, we present the results of the error analysis.

1) COMPARISON OF MATCHING TECHNIQUES

It can be observed from the results of the UA dataset presented in Table 4, that fastText outperformed all other techniques in terms of overall scores, as well as for each type of pair. Similarly, it can be observed from the results of BR and MG datasets presented in Table 5 and 7, the scores achieved by fastText are higher than all the other 26 techniques. A possible reason for the higher effectiveness of fastText is that these embeddings have generated vector representation of each word in a high-dimensional space using a very large collection of tokens, which is useful to calculate the similarity between label pairs. A notable observation from the results of AM dataset presented in Table 6 is that the F1 scores achieved by some matching techniques, such as Jaro and Path similarity are higher than that of fastText. However, the recall scores achieved by these techniques are lower than the other techniques indicating the effectiveness of fastText to identify a higher number of corresponding pairs.

2) COMPARISON OF TYPES OF PAIRS

To gain deep insights into the performance of matching techniques, we computed the average Recall score for each type of pair: VB, MC, and HR. The generated average scores for the three types of pairs using UA, BR, AM, and MG datasets are presented in Figure 6, 7, 8 and 9, respectively. It can be observed from the results of the UA dataset presented in Figure 6 that all the types of techniques achieved a very high. Recall score for VB pairs, a lower score for MC pairs, and even lesser scores for HR pairs. These
results indicate: a) a deeper insight about the effectiveness of matching techniques is desired, b) the overall F1 score achieved by each matching technique is inflated due to the VB pairs, and c) the HR corresponding pairs are harder-to-detect than the other two-types of pairs. It can also be observed from the figure that the difference in the performance of techniques for VB pairs is minuscule. A manual review of these pairs revealed that the differences between VB pairs are so small that even the syntactic measures can identify these corresponding pairs. Similar observations can be made about BR, AM and MG datasets, from the results presented in Figure 7, 8 and 9.

3) COMPARISON OF DATASETS

Figure 10 shows a comparison of the Recall scores of all the four datasets. Each bar in the graph represents the average Recall score of all the matching techniques. It can be observed

| Technique     | Overall results     | Individual results |
|---------------|---------------------|--------------------|
|               | P   | R   | F1  | VB | MC | HR  |
| Dice          | 0.98| 0.22| 0.36| 0.91| 0.26| 0.00|
| Jaccard       | 1.00| 0.15| 0.26| 0.83| 0.00| 0.01|
| Cosine TF     | 0.98| 0.22| 0.35| 0.87| 0.26| 0.00|
| Cosine TFIDF  | 0.95| 0.21| 0.35| 0.87| 0.21| 0.01|
| Substring     | 0.97| 0.15| 0.26| 0.72| 0.08| 0.00|
| ED-15         | 0.94| 0.19| 0.32| 0.91| 0.10| 0.00|
| ED-16         | 0.72| 0.32| 0.45| 0.93| 0.45| 0.09|
| ED-17         | 0.95| 0.20| 0.34| 0.91| 0.13| 0.02|
| Hamming       | 0.96| 0.17| 0.29| 0.87| 0.06| 0.00|
| LCS           | 0.93| 0.27| 0.42| 0.93| 0.37| 0.03|
| Jaro          | 0.81| 0.33| 0.47| 0.98| 0.44| 0.09|
| Jaro Variant  | 0.77| 0.38| 0.51| 0.98| 0.45| 0.17|
| Mean of DW    | 0.98| 0.22| 0.36| 0.89| 0.26| 0.01|
| Mean          | 1.00| 0.17| 0.28| 0.83| 0.08| 0.00|
| Monge         | 0.95| 0.24| 0.38| 0.87| 0.26| 0.04|
| Dice          | 1.00| 0.13| 0.23| 0.72| 0.00| 0.00|
| intersec. of senses | 0.98| 0.17| 0.28| 0.83| 0.08| 0.00|
| Greedy Lin    | 1.00| 0.26| 0.42| 0.91| 0.37| 0.02|
| Optimum Lin   | 1.00| 0.25| 0.40| 0.93| 0.32| 0.01|
| Greedy Lesk   | 1.00| 0.24| 0.39| 0.91| 0.32| 0.01|
| Optimum Lesk  | 1.00| 0.23| 0.37| 0.93| 0.24| 0.01|
| Monge Lin     | 0.86| 0.32| 0.47| 0.65| 0.29| 0.23|
| Path Similarity| 0.97| 0.22| 0.35| 0.85| 0.19| 0.03|
| Lin of components| 0.93| 0.22| 0.35| 0.59| 0.26| 0.09|
| fastText      | 0.88| 0.46| 0.60| 0.98| 0.56| 0.25|
| GloVe         | 0.88| 0.35| 0.50| 0.93| 0.45| 0.13|
| Word2Vec      | 0.87| 0.36| 0.51| 1.00| 0.41| 0.15|

| Technique     | Overall results     | Individual results |
|---------------|---------------------|--------------------|
|               | P   | R   | F1  | VB | MC | HR  |
| Dice          | 0.96| 0.71| 0.82| 0.99| 0.42| 0.07|
| Jaccard       | 1.00| 0.67| 0.80| 0.98| 0.08| 0.00|
| Cosine TF     | 0.96| 0.71| 0.82| 0.99| 0.38| 0.07|
| Cosine TFIDF  | 0.99| 0.70| 0.82| 0.97| 0.33| 0.09|
| Substring     | 0.99| 0.59| 0.74| 0.85| 0.08| 0.04|
| ED-15         | 0.96| 0.69| 0.80| 0.99| 0.17| 0.05|
| ED-16         | 0.80| 0.79| 0.80| 0.99| 0.58| 0.33|
| ED-17         | 0.94| 0.70| 0.80| 0.99| 0.21| 0.06|
| Hamming       | 0.99| 0.66| 0.79| 0.97| 0.04| 0.02|
| LCS           | 0.91| 0.74| 0.82| 0.99| 0.54| 0.15|
| Jaro          | 0.87| 0.81| 0.84| 0.99| 0.67| 0.37|
| Jaro Variant  | 0.85| 0.81| 0.83| 0.99| 0.58| 0.41|
| Mean of DW    | 0.96| 0.71| 0.82| 0.99| 0.38| 0.07|
| Mean          | 0.96| 0.69| 0.80| 0.99| 0.29| 0.02|
| Monge         | 0.92| 0.72| 0.81| 1.00| 0.42| 0.10|
| Dice          | 1.00| 0.65| 0.79| 0.96| 0.00| 0.00|
| intersec. of senses | 0.97| 0.69| 0.81| 0.98| 0.17| 0.08|
| Greedy Lin    | 0.94| 0.74| 0.83| 0.99| 0.42| 0.15|
| Optimum Lin   | 0.91| 0.74| 0.82| 1.00| 0.38| 0.16|
| Greedy Lesk   | 0.95| 0.72| 0.82| 0.99| 0.42| 0.11|
| Optimum Lesk  | 0.92| 0.74| 0.82| 1.00| 0.38| 0.14|
| Monge Lin     | 0.76| 0.63| 0.69| 0.73| 0.58| 0.37|
| Path Similarity| 0.99| 0.71| 0.82| 0.97| 0.29| 0.12|
| Lin of components| 0.83| 0.59| 0.69| 0.77| 0.42| 0.15|
| fastText      | 0.69| 0.86| 0.77| 1.00| 0.67| 0.55|
| GloVe         | 0.74| 0.83| 0.79| 1.00| 0.67| 0.44|
| Word2Vec      | 0.81| 0.82| 0.81| 1.00| 0.71| 0.37|
from the figure that the overall Recall score achieved by techniques for AM dataset is highest whereas it is lowest for the MG dataset. A manual analysis of the AM dataset revealed that a majority of the corresponding pairs, almost 67% pairs, lie in VB category, and hence they are easily detectable. It can also be observed that there is a slight difference between the effectiveness of techniques for VB pairs, whereas the difference is significant for MC and HR pairs. The reason for the VB pairs is that the labels involved in these pairs are slightly different from each other, hence, these corresponding pairs are identified by all the techniques.

D. ERROR ANALYSIS

We have performed an error analysis of the results of matching techniques. In particular, Table 8 presents the quantitative analysis of the results of word-embeddings, which are the most effective category of matching techniques. The first row in the table shows that the number of corresponding pairs that were identified by at least one type of word embeddings but not identified by any syntactic or semantic technique. These numbers represent the effectiveness of word embedding techniques as compared to other techniques.

The second row in Table 8 shows the number of corresponding pairs in each dataset which were not identified by any matching technique, including word embeddings. It can be observed from the table that 3893 pairs in the MG dataset were not identified by any matching technique, which is 47% of total corresponding pairs. This higher percentage of the corresponding pairs represents the challenging nature of the MG dataset.

We also performed a qualitative analysis of the corresponding pairs that were not identified by any matching technique. In particular, we manually analyzed each corresponding pair and grouped them together to identify the causes of errors. A brief overview of these causes are as follows:

- Firstly, the activity pairs in which the length of one label is significantly greater than the other label in the pair were not identified by any matching technique. The
significant the difference the similarity score leads to a reduced similarity score, even if the synonym or all the words in the smaller label are available in the larger label. Consequently, these label pairs are declared as non-corresponding pairs due to the cutoff threshold of 0.75. The risk with lowering this threshold is that several non-corresponding pairs are declared as corresponding pairs, hence, reducing the Precision score. The examples of the corresponding pairs having significantly different lengths are presented in Table 9.

- Secondly, the activity pairs that were declared as corresponding pairs in a business context were not identified by any matching technique. That is, these pairs include words having no semantic relationship between each other, according to the WordNet. However, the two labels have the same business meanings in the context of the business process. The example of such an activity pair is “Initiate discussion if on due date - Start debate through message”.

- Thirdly, the corresponding activity pairs in which one label subsumes the other label were not correctly identified by any matching technique. For instance, in the corresponding pair “Reject - Send letter of rejection”, the later label subsumes the former label. On the other hand “Update the applicant status - Mark applicant as suitable” is an example of the corresponding pairs in which one label semantically subsumes the other. That is, both the options of marking suitable and unsuitable are represented in the label “update the application status”, whereas, the other label semantically subsumes in the first label as it only represents one option.

- Finally, the corresponding activity pairs in which the number of semantically similar words are less than the dissimilar words were not identified by matching techniques. Our manual investigation of the errors identified that it is due to the cutoff threshold of 0.75.

V. CONCLUSIONS AND DISCUSSION

The effectiveness of PMM techniques is evaluated using three widely-used benchmark datasets, in terms of Precision, Recall, and F1 scores. Our comprehensive analysis of these datasets, as well as the results of existing techniques have revealed two interrelated issues: a) the datasets include a substantial amount of identical or similarly labels which inflates the performance scores, and b) the overall Precision, Recall, and F1 scores are useful for a surface-level evaluation of a technique, however, it does not provide the necessary insights about the capabilities of a matching technique.

In this study, we have made three key contributions. Firstly, we have refined four existing datasets containing 92565 activity pairs, by omitting their discrepancies. Subsequently, we have enhanced these datasets by classifying the corresponding pairs into three types, based on the level of differences in the participating labels. The notable features of this contribution are that these types are well-rooted in literature, and we have employed a systematic procedure for the extension involving clearly defined classification criteria and multiple researchers.

Secondly, we have identified an extensive set of 27 matching techniques from a literature survey and developed a taxonomy of these techniques. This study includes the procedure for developing the taxonomy along with an overview of each matching technique. As a third contribution, we have performed 432 matching experiments using the enhanced datasets for a comprehensive evaluation of 27 matching techniques.

The results of the experiments revealed that the state-of-the-art word embeddings exhibit are somewhat promising for
identifying two types of activity pairs, Modified Copy and Heavy Revision Pairs. Whereas, both syntactic and semantic techniques almost completely failed in the identification of Heavy Revision Pairs. Based on the results we recommend the need for developing the next generation of matching techniques that can show the promising results for all the three types of pairs.

REFERENCES

[1] B. List and B. Korherr, “An evaluation of conceptual business process modelling languages,” in Proc. ACM Symp. Appl. Comput. SAC, 2006, pp. 1532–1539.

[2] M. La Rosa, H. A. Reijers, W. M. P. van der Aalst, R. M. Dijkman, J. Mendling, M. Dumas, and L. García-Bañuelos, “APROMORE: An advanced process model repository,” Expert Syst. Appl., vol. 38, no. 6, pp. 7029–7040, Jun. 2011.

[3] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, vol. 2. Heidelberg, Germany: Springer, 2018.

[4] E. Kuss, H. Leopold, H. Van der Aa, H. Stuckenschmidt, and H. A. Reijers, “Probabilistic evaluation of process model matching techniques,” in Proc. Int. Conf. Conceptual Modeling, Gifu, Japan: Springer, 2016, pp. 279–292.

[5] F. Jabeen, H. Leopold, and H. A. Reijers, “How to make process model matching work better? An analysis of current similarity measures,” in Proc. Int. Conf. Bus. Inf. Syst. Poznań, Poland: Springer, 2017, pp. 181–193.

[6] C. Rodriguez, C. Klinkmüller, I. Weber, P. Daniel, and F. Casati, “Activity matching with human intelligence,” in Proc. Int. Conf. Bus. Process Manage. Rio de Janeiro, Brazil: Springer, 2016, pp. 124–140.

[7] A. Awad, A. Polyvyanyy, and M. Weske, “Semantic querying of business process models,” in Proc. 12th Int. IEEE Enterprise Distrib. Object Comput. Conf., Sep. 2008, pp. 85–94.

[8] M. Dumas, L. García-Bañuelos, M. La Rosa, and R. Uba, “Fast detection of exact clones in business process model repositories,” Inf. Syst., vol. 38, no. 4, pp. 619–633, Jun. 2013.

[9] M. La Rosa, M. Dumas, R. Uba, and R. Dijkman, “Business process model merging: An approach to business process consolidation,” ACM Trans. Softw. Eng. Methodol., vol. 22, no. 2, p. 11, Mar. 2013.

[10] U. Cayoglu, R. Dijkman, M. Dumas, P. Fettke, L. García-Bañuelos, P. Hake, C. Klinkmüller, H. Leopold, A. Ludwig, P. Loos, J. Mendling, A. Oberweis, A. Schoknecht, E. Sheerit, T. Thaler, M. Ullrich, I. Weber, and M. Weidlich, "Report: The process model matching contest 2013,” in Proc. Int. Conf. Bus. Process Manage. Beijing, China: Springer, 2013, pp. 442–463.

[11] C. Meilicke, H. Leopold, E. Kuss, H. Stuckenschmidt, and H. A. Reijers, “Overcoming individual process model matcher weaknesses using ensemble matching,” Decis. Support Syst., vol. 100, pp. 15–26, Aug. 2017.

[12] G. Antunes et al., “The process model matching contest 2015,” Enterprise Modelling Inf. Syst. Archit., vol. 248, pp. 127–153, Sep. 2015.

[13] E. Kuss, H. Leopold, H. van der Aa, H. Stuckenschmidt, and H. A. Reijers, “A probabilistic evaluation procedure for process model matching techniques,” Data Knowl. Eng., vol. 117, pp. 393–406, Sep. 2018.

[14] M. Achichi et al., “Results of the ontology alignment evaluation initiative 2017,” in Proc. 12th Int. Workshop Ontology Matching co-Located with 16th Int. Semantic Web Conf., vol. 2032. Vienna, Austria: CEUR-WS, 2017, pp. 61–113.

[15] M. Ali and K. Shahzad, “Enhanced benchmark datasets for a comprehensive evaluation of process model matching techniques,” in Proc. Workshop Enterprise Organizational Modeling Simulation, Tallinn, Estonia: Springer, 2018, pp. 107–122.

[16] P. Clough, R. Gaizauskas, S. S. L. Piao, and Y. Wilks, “METER: Measuring Text reuse,” in Proc. 40th Annu. Meeting Assoc. Comput. Linguistics - ACL, 2001, pp. 152–159.

[17] P. Clough and M. Stevenson, “Developing a corpus of plagiarised short answers,” Lang. Resour. Eval., vol. 45, no. 1, pp. 5–24, Mar. 2011.

[18] S. Sameen, M. Sharjeel, R. M. A. Nawab, P. Rayson, and I. Muneeb, “Measuring short text reuse for the urdu language,” IEEE Access, vol. 6, pp. 7412–7421, 2018.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 3111–3119.

[20] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” 2016, arXiv:1607.04606. [Online]. Available: http://arxiv.org/abs/1607.04606

[21] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2014, pp. 1532–1543.

[22] C. D. Manning, P. Raghavan, and H. Schütte, Introduction to Information Retrieval. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[23] G. A. Miller, “WordNet: A lexical database for English,” Commun. ACM, vol. 38, no. 11, pp. 39–41, 1995.

[24] S. Patwardhan, S. Banerjee, and T. Pedersen, “Using measures of semantic relatedness for word sense disambiguation,” in Proc. Int. Conf. Intell. Text Process. Comput. Linguistics. Mexico City, Mexico: Springer, 2003, pp. 241–257.

[25] A. Budanitsky and G. Hirst, “Evaluating WordNet-based measures of lexical semantic relatedness,” Comput. Linguistics, vol. 32, no. 1, pp. 13–47, Mar. 2006.

MUHAMMAD ALI received the M.Sc. degree from the Punjab University College of Information Technology (PUCIT), University of the Punjab, Lahore, and the master’s degree from the University of Management and Technology. He is currently pursuing the Ph.D. degree with the PUCIT, University of the Punjab. He is a Systems Analyst with the Directorate of IT Services, University of Gujrat, Gujrat. He has over a decade of software development experience.

KHURRAM SHAHZAD received the master’s and Ph.D. degrees from the KTH Royal Institute of Technology, Stockholm. He has been associated with the Information Systems Group, Technical University Eindhoven, Eindhoven, and the University of Fribourg, Fribourg. He is currently an Assistant Professor with the Punjab University College of Information Technology (PUCIT), University of the Punjab, Lahore. He has published more than 35 articles in international conferences and journals.

SYED IRTAZA MUZAFFAR received the M.Phil. degree in computer science from the Punjab University College of Information Technology (PUCIT), University of the Punjab, Lahore, where he is currently pursuing the Ph.D. degree with the PUCIT. He has three years of teaching and development experience. He is currently a Lecturer of computer science with the University of Central Punjab, Lahore.

MUHAMMAD KAMRAN MALIK received Kaman holds a the Ph.D. degree in computer science. He has more than 15 years of teaching and development experience. He holds one has authored one U.S. patent. He has provided consultancy to many multinational firms on natural language processing, machine learning, and data science. He has authored 25 journal articles and conference papers. He is an Assistant Professor with the at Punjab University College of Information Technology (PUCIT), University of the Punjab, Lahore, Pakistan. His research interests include natural language processing, machine learning, and data science.