Process Model Searching: A Corpus of Summary Textual Descriptions

Madiha Khalid, Syed Irtaza Muzaffar Shah, Khurram Shahzad, Faisal Bukhari
Punjab University College of Information Technology, University of the Punjab, Lahore
madiha.khalid| mcsf14m018|khurram|faisal.bukhari@pucit.edu.pk

Abstract— Searching process models is a key feature of process model repositories. The efficiency and effectiveness of searching depends on the underlying matching techniques. Recent studies have proposed the use of textual descriptions of process models alongside process models, for a comprehensive search. However, it is common for organizations to have longer textual descriptions, consequently, the use of full length textual descriptions may negatively affect the efficiency of matching techniques. To overcome this problem, we advocate the use of summary textual descriptions. This however requires, rigorous investigations to prove the efficiency and effectiveness of the use of summary textual descriptions over full length textual descriptions. Conducting these studies require, a corpus of textual descriptions of process models and corpus/corpora of their summary textual descriptions. To fulfill these prerequisites, in this paper we focus on, explaining the process of generating a) a corpus of full length textual descriptions of process models, b) corpora of summary textual descriptions, using two text summarization algorithms. Further, we employ two basic text matching techniques to establish that the summary descriptions in the two corpora are different from each other and thus the choice of the summarization technique is a non-trivial task.

Keywords—Process models; textual descriptions; text summarization; summary textual descriptions;

I. INTRODUCTION

Increasing number of organizations are modeling their business processes to explicitly depict the ordering and dependencies between activities [1]. These models, formally called process models, are useful in a number of contexts, including sharing organizational processes to a new employee and representing requirements of ERP systems [2]. Given that, it is common for organizations to have a collection of hundreds or even thousands of business processes [3], process repositories are used to manage these collection of models [4]. A key feature of process model repositories is, searching relevant process models for a given query process model. However, the effectiveness of these searching techniques rely on the underlying process matching techniques [5]. In fact, several process matching techniques have been proposed [3, 6, 7, 8, 9, 10], which rely on the use of label features, structural features and behavioral features of process models. However, the effectiveness of these matching techniques is too low for use in practice.

To address that problem, a recent study [11] have proposed to store textual descriptions of process models in process repositories, and use these descriptions (alongside process models) for matching process models [12]. The study has established that the use of textual descriptions (alongside process models) indeed increases the effectiveness of process model search. However, it is common for organizations to have long textual descriptions. For instance, an Austrian bank's process collection has 119 textual descriptions of processes, with an average length of 13,130 words and the longest description has a length of 60,558 words [12]. We contend, that the use of these full length textual descriptions will arguable decrease the efficiency of process matching and thereby slowdown the retrieval of relevant models. To overcome that problem, we advocate the use of summary textual descriptions for process matching. However, recommending the use of summary textual descriptions for process matching requires analyses of the tradeoff (in terms of efficiency and effectiveness) between summary and full-length textual descriptions. Such investigations require, a) a corpus of full-length textual descriptions of process models, and b) corpora of summary textual descriptions, of the same set of process models.

Generating a corpus is a non-trivial task, because a number of questions are associated with it, such as, what should be the size of corpus, and what procedures should be employed to generate the corpus? It is to be noted that the idea of generating corpus is not new. In fact, corpora has been developed for various fields of research such as, information retrieval [13], process matching [14], and social network detection [15, 16] etc. In this study we focus on, a) explaining the process of generating a corpus of full length textual descriptions of process models, b) explaining the process of generating corpora of summary textual descriptions of process models, using two auto summarization algorithms. Note, we focus on generating and analyzing corpora of summary textual descriptions of process models. It is because, we believe that analyzing and recommending the use of summary textual descriptions for process matching (instead of complete textual descriptions) formulates a whole separate research problem. Furthermore, we apply two basic techniques, n-gram overlap [17] and Longest Common Subsequent (LCS) [18] to establish that two auto summarization algorithms generate different summaries and therefore a further investigation is required to establish the choice of text summarization algorithms.

The rest of the paper is organized as follows: the section II gives an illustration of the issues associated with generating summary textual descriptions. Section III provides the details of the generated corpus. Section IV discusses the experimental setup and results. Finally the paper concludes in section V.

II. MOTIVATING EXAMPLE

This section introduces the background to this work by providing an example process model, its complete textual description and two possible summary textual descriptions. Fig. 1 and 2 shows an example loan application process model and its equivalent full-length textual description, respectively. The example process model, shown in Fig. 1, is designed in Business Process Modeling Notation (BPMN) [19] – the de facto standard process modeling language and Signavio [20] – an online process modeling tool that follows most of the process modeling guidelines [21]. The model contains, one pool (labeled as ACME Inc.), two XOR gateways (represented by a diamond sign with a X sign inside it), one start event (represented by a circle), one end event (represented by a solid circle), two lanes (labeled as finance department and employee), and 9 activities (labeled as, submit application for loan, receive request, etc.).

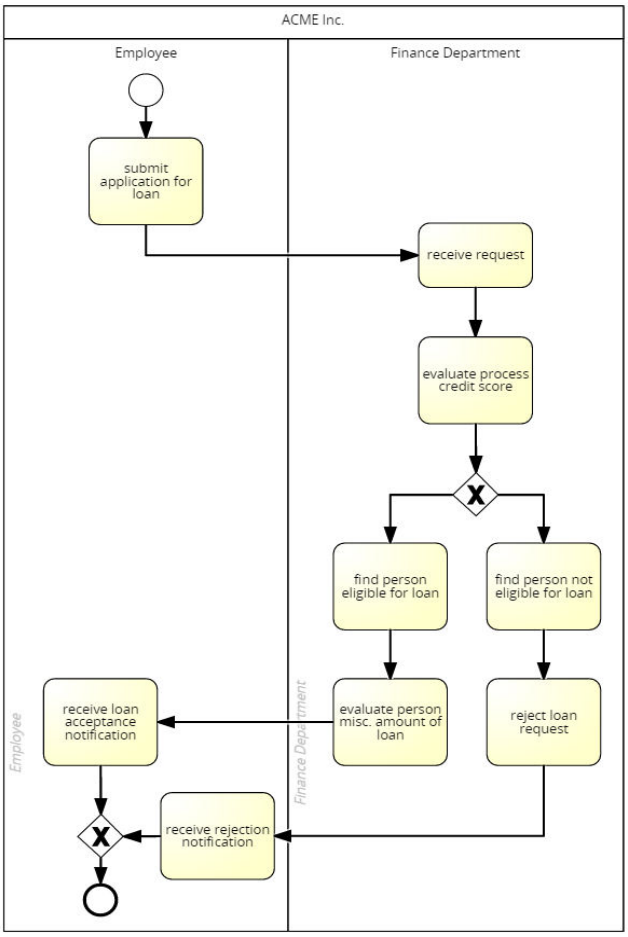

Fig. 1. An example of notation based business process model designed using BPMN.

> The loan application process at ACME Inc. starts when an employee submits application for loan. Finance department receives that request. Finance department then evaluates person's credit score if it finds that the person is not eligible for loan then the loan request will be rejected and the employee receives a rejection notification. On the contrary, if the finance department finds the person eligible for loan it will evaluate the person's misc. amount of loan and sends the loan acceptance notification to customer.

Fig. 2. Full length textual description of the example process model.

Summary 1	*Summary 2*
The loan application process starts when finance department receives the loan request from employee. It evaluates the application and send back the decision.	At ACME Inc., if an employee seeks loan he/she submits a loan application that is evaluated by finance department. Finance department accesses loan seeker's credit score and sends the acceptance or rejection notification as per the assessment result.

Fig. 3. Summary textual descriptions of the example process model

It is worth noting that there is a clear correspondence between the process models presented in Fig. 1 and its corresponding textual description presented in Fig. 2. For instance, from process model as well as from the textual description it can be seen that the process belongs to ACME Inc., and it starts when an employee submits an application for loan. Similarly, it can also be observed that the process ends with a rejection notification or acceptance notification.

In order to demonstrate that more than one summary descriptions can be generated from one input textual description, two researchers were asked to independently generate summary descriptions of the complete textual description presented in Fig. 1. The generated summary textual descriptions, produced by the researchers, are presented in Fig. 3. Two key observations can be made about the summary descriptions produced by the researchers, despite the fact they share same baseline textual description: a) the two summary descriptions are different from each other, b) the length of two descriptions are different i.e. the length of summary 1 is less than that of summary 2. It is thus unclear, which one of the two textual descriptions truly represent the summary of the full length textual description. Similarly, it can be demonstrated, a single researcher can produce difference summaries of the same textual description on different occasions. This justifies the use of automatic text summarization algorithms while generating the summary textual descriptions.

III. THE CORPORA

In this section, we explain the process of generating a corpus of full-length textual descriptions and corpora of summary textual descriptions.

A. *Full length Textual Descriptions*

We have access to a collection of 669 process models that are designed using BPMN – the de facto process modeling language. The collection is modeled in Signavio – the most recommended process modeling tool [21], and it is available as JSON and PDF formats. The choice of the collection is motivated by the fact that the collection contains process models with diverse label and structural features [14]. More precisely, the collection includes 150 Original (O) process models and three other handcrafted variants of these models, Near Copy (NC), Light Revision (LR) and Heavy Revision (HR). The variants are crafted by employing a systematic and rigorous procedure that we deem necessary to impart diversity in labels and structure of models with an aim to challenge the abilities of process matching techniques. The NC variant of a model is generated by slightly changing the formulation of each label of the model, whereas the LR variant is generated by substantially changing the formulation of each label of the model. HR variant is generated by changing the formulation of each label as well as the control flow between activities of the models.

The smallest model in the collection contains 11 activities and the largest model contains 54 activities. Another unique feature of the collection is, the models included in the collection follow most of the process modeling guidelines, presented by Mendling et al. [22]. For instance, there is no process model in our collection that contains a split gateway node without a corresponding join gateway node. The human effort involved in generated the collection can be understood by the number of operations performed while generating model variants i.e. a) 24092 insertion, deletion and substitution or words were performed to generate label variants of process models, and b) 1764 structural change operations were performed to generate structural variants of process models.

There were two ways of generating textual description of the collection of process models, manual or automatic [23]. A number of factors limit the available choices. Manual generation of textual descriptions require understanding of BPMN, the modeling notation used for generating the collection i.e. a user having no or limited knowledge of BPMN may not correctly comprehend a BPMN process model and thereby may not generate a truly representative textual description of the model. Further, manually generating textual description of such a large number of process models is a time consuming and error pruned task. Also, the quality of description is dependent on the writing skills of the involved humans i.e. the description of the same process models may differ from person to person.

Due to the challenges associated with the manual way of generating textual description, the Natural Language Generation System (NLGS) developed by Leopold et al. [11] is used to generate textual description of 669 process models. As far as we are aware, NLGS is the only available tool that can automatically generate textual description of a process model. It uses a well-established technique that takes a process model in JSON format as input and generates its textual description. Note, an empirical evaluation of the textual description generated by NLGS has established that the NLGS generated textual description is superior than the human generated textual description, in terms of completeness, structure and linguistic complexity [11]. The study has also established that the texts are understandable by naïve users and effectively allow the reader to interpret the process model semantics [11]. Furthermore, the textual description generated by the tools has been used in previous studies for verification of process models. We yielded a corpus of textual descriptions of 669 process models using NLGS.

B. *Generating Summary Descriptions*

In order to generate summary descriptions of process models, by taking input the full length textual descriptions generated by NLGS, we have used two widely used summarization algorithms, TextRank [24] and LexRank [25]. A large number of natural language processing communities rely on the performance of these two algorithms. Both of these algorithms employ an extractive approach to generate summary descriptions [26]. In extractive approaches, the importance of each phrase/sentence is computed with respect to the complete document. Subsequently, a summary description is generated based on the importance of phrases/sentences in the text. The key strength of the approaches that fall in this category is, they purely rely on the content of the source description and do not change the order of words within a phrase or a sentence i.e. these techniques do not induce new phases/sentences that does not exist in the source text. Below, we provide an overview of the two summarization algorithms.

TextRank: TextRank [24] is an expansion of page rank algorithm in which the source text is tokenized into sentences and represented as a vertex in the graph. Subsequently, edges between these sentences are marked on the bases of overlapping between them. Subsequently, it iteratively compute the scores of each vertex using graph based ranking algorithm, until convergence. Finally, vertices are sorted on the bases of their final scores.

LexRank: LexRank [25] first tokenize the document into sentences and represents each sentence as vertex. Then, it adds edges between these vertices on the bases of inverse document frequency (idf) cosine similarity [25]. Note, we changed the concept of inverse document frequency (idf) at the collection level, to inverse sentence frequency i.e. log of, total number of sentences in the process description divided by the number of sentences in which the word occurs". Subsequently, if the generated similarity score between two sentences is above a certain threshold value, a value 1 is stored in respective index of these sentences matrix and increment 1 in degree values otherwise store 0 and no increment in degree value. Lastly, final score of each sentence is computed using power method followed by vertices sort.

C. *Characteristics of the Summary Corpora*

Table I summaries some of the main statistics of the TextRank and LexRank generated summary descriptions. From the table, it can be seen that the average length of the TextRank generated summary descriptions are more than that of LexRank generated summary descriptions (i.e. 105 > 95 for 75% summary, 71 > 63 for 50% summary, and 34 > 30 for 25%

TABLE I. CHARACTERISTICS OF TEXTRANK AND LEXRANK GENERATED SUMMARY CORPORA

			Total Words				Unique Words				Stop Words			
			Min	*Max*	*Avg*	*Total*	*Min*	*Max*	*Avg*	*Total*	*Min*	*Max*	*Avg*	*Total*
Complete Description (669 Models)			48	376	131	87772	24	121	4	2883	21	143	53	35637
Summary Descriptions	*75%*	***TextRank***	37	321	105	70706	19	107	3	2652	14	119	41	27680
		LexRank	37	273	95	63674	19	97	3	2625	15	110	38	25743
	50%	***TextRank***	23	224	71	48031	12	72	3	2243	9	81	27	18432
		LexRank	23	186	63	42738	14	74	3	2203	11	75	25	17323
	25%	***TextRank***	8	110	34	22767	5	44	2	1591	4	45	12	8680
		LexRank	8	91	30	20151	7	44	2	1499	4	38	12	8330

summary) and less than the full-length description. From the quantity of unique words used in generating summary, it can be seen that as the summary decreases to 25%, the number of unique words used by TextRank are significantly more than LexRank (total number of unique words: 1591 > 1499 and maximum number of unique words: 74 > 44), compared to the 50% and 75% summaries. These numbers show that for generating 25% summary TextRank algorithm uses more unique words than LexRank.

IV. COMPUTING SIMILARITY

To compare the two corpora generated by TextRank and the other by LexRank we rely on the use of two well-known similarity estimation models: n-gram overlap and Longest Common Subsequence.

A. *Similarity Estimation Models*

The estimation models used for similarity computation are briefly discussed below.

N-gram Overlap: N-gram overlap is the simplest quantitative similarity measure to compute the similarity between two strings [27]. Historically, n-gram overlap is widely used and proven for similarity detection in textual documents, such as, plagiarism detection and detecting the fraction of reused content in journalism [28, 29]. In computational linguistics, the term n-gram refers to a contiguous sequence of n language units, where these language units can be letters, words or syllables. The selection on the language unit depends upon the application. In our case we used n-gram of words and considered uni-gram (n-gram of size 1) as most representational size of n-gram. N-gram overlap determines the amount of n-grams that are common in a pair of texts. The overlapping fraction using n- gram can be measured in a number of ways, for example using Jaccard coefficient or overlap coefficient.

Jaccard coefficient (as defined in equation 1) computes similarity as the size of intersection of common n-grams divided by the number of shared n-grams. Whereas, overlap coefficient (equation 2) estimates the similarity by dividing the size of intersection of common n-grams by the size of one of the strings.

$$JC(X,Y) = \frac{|X_n \cap Y_n|}{|X_n \cup Y_n|} \quad (1)$$

$$OC(X,Y) = \frac{|X_n \cap Y_n|}{\min(|X_n|, |Y_n|)} \quad (2)$$

Where, X_n is the n-grams in first text and Y_n is the n-grams of second text.

In our experiments, a pair of summary documents are compared using overlap coefficient. The resulting similarity score ranges between 0 and 1. Where 0 indicates no similarity and 1 indicates complete similarity. Since n-gram overlap considers only fixed sized grams and does not preserve the ordering of the grams, therefore, we have also used an order preserving algorithm, Longest Common Subsequence.

Longest Common Subsequence: Longest common subsequence (LCS) [18] is another widely relied upon approach by natural language processing community. LCS uses slightly different approach than n-gram. It represents text as tokens of words or characters while preserving the order of the tokens. The similarity between a pair of documents is computed on the basis of string edit distance i.e. the number of string operations (insert, delete, alter) required to convert one string token into another.

In our experiments, we computed LCS between two textual descriptions and divided the resultant value by one of the source texts to get a normalized similarity score, called LCS_{norm}.

$$LCS_{norm}(X,Y) = \frac{|LCS(X,Y)|}{\min(|X|, |Y|)} \quad (3)$$

B. Similarity Scores between Summary Descriptions

Table II shows the average similarity score of TextRank generated summary descriptions with LexRank generated summary descriptions, using n-gram overlap (unigram) and LCS. In order to generate the average similarity scores, at first, we created 669 pairs of summary descriptions generated by TextRank and LexRank (TSD:LSD) by adjusting the summary threshold to 75%. Subsequently, the similarity score of each pair was computed using n-gram overlap and LCS. Thereafter, the average of the 669 similarity scores and their standard deviation was calculated for each estimation technique, separately. Similarly, 669 pairs were generated for each of the following: 50% summary descriptions generated by TextRank and LexRank, and 25% summary descriptions generated by TextRank and LexRank.

TABLE II. AVERAGE SIMILARITY SCORES BETWEEN TSD:LSD PAIR

	Summary size					
	75%		***50%***		***25%***	
Estimation Model	***Avg.***	***Stdev***	***Avg.***	***Stdev***	***Avg.***	***Stdev***
N-gram	0.93	0.04	0.79	0.08	0.59	0.14
LCS	0.83	0.08	0.59	0.12	0.43	0.14

The similarity score of 0.93, using n-gram overlap represents that, an average 93% of the vocabulary used by TextRank to generate summary textual descriptions is also used by LexRank to generate summary textual descriptions. Note, this higher score does not represent that the TextRank and LexRank generated summary descriptions are 93% similar; rather it represents that 93% of the vocabulary (unique words) used by the two summarization algorithms, overlap. Further, the slight variation in the standard deviation indicate the little change in the similarity score across 669 pairs.

From the table it can be observed that, as the summary size decreases, the vocabulary overlap (average unigram overlap score) also decreases. This consistently decreasing score shows that the summary descriptions generated by the two algorithms is different, in terms of the vocabulary used for generating summaries. A deeper examination of the generated summaries reveal that the reason for these decreasing scores is rooted in the underlying techniques used by the two algorithms to rank sentences i.e. TextRank uses a variant of PageRank to compute the importance of each sentence in the document, whereas LexRank uses inverse sentence frequency (see preceding section), to compute the importance of each sentence in the document. As a result, the two algorithms compute different rank for each sentence.

From the table it can also be observed that the average LCS similarity scores are relatively low. These lower scores indicates that, in the longest subsequence that is common between the two summary descriptions, there is a significant change in the ordering of tokens/words. A deeper examination of summaries revealed the reasoning for this lower score i.e. since the two algorithms use different techniques to rank sentences, therefore the sentences chosen for generating summary are also different. This reduces the size of the longest subsequence that is common between the summaries generated by LexRank and TextRank.

From this discussion we conclude that the summary textual descriptions generated by the two algorithms are significantly different from each other and thus the choice of the summarization technique is non-trivial.

C. Similarity Scores between Summary and Full length Descriptions

Table III shows the average similarity score of TextRank generated summary descriptions with the full-length textual description and LexRank generated summary descriptions with full-length textual description, using n-gram overlap (unigram) and LCS. In order to generate the average similarity score, at first we created 669 pairs of full-length textual descriptions and summary descriptions generated by TextRank (FTD:TSD) by adjusting the summary threshold to 75%. Subsequently, the similarity score of each pair was computed using n-gram overlap and thereafter, the average of the 669 similarity scores was calculated. Accordingly, the 0.93 similarity score in the table represents, that on an average 93% of the vocabulary available in the full length textual description is also available in the 75% summary description generated by TextRank. Similarly, 669 pairs were generated between of full-length textual description with the following, 50% summary descriptions and 25% summary descriptions generated by TextRank. Also, the process was repeated by creating 669 pairs of full-length textual descriptions and LexRank generated 75%, 50% and 25% summary descriptions (FTD:LSD).

TABLE III. AVERAGE SIMILARITY SCORES BETWEEN FTD:TSD AND FTD:LSD

Estimation Model	**Average similarity score**					
	FTD: TSD			***FTD:LSD***		
	75%	***50%***	***25%***	***75%***	***50%***	***25%***
N-gram	0.93	0.80	0.64	0.92	0.83	0.65
LCS	0.99	0.99	0.99	0.99	0.99	0.98

From the table it can be observed that, as the summary size decreases the vocabulary overlap (average unigram overlap score) decreases for both pairs, FTD:TSD and FTD:LSD. This consistently decreasing score shows the expected behavior i.e. a reduced set of vocabulary is used to generate summary description. Another observation is, there is no significant difference in the average score of the two pairs, FTD:TSD and FTD:LSD. We contend, these slight differences in the average overlap scores of the two pairs does not represent that the TextRank & LexRank generated summary descriptions are identical to the full-length textual descriptions. It rather represents that the vocabulary used by the two algorithms to generate summary descriptions significantly overlap with the vocabulary used in the full-length textual descriptions.

To our surprise the LCS based similarity score is 0.99 for both pairs, FTD:TSD and FTD:LSD. Also, the value does not

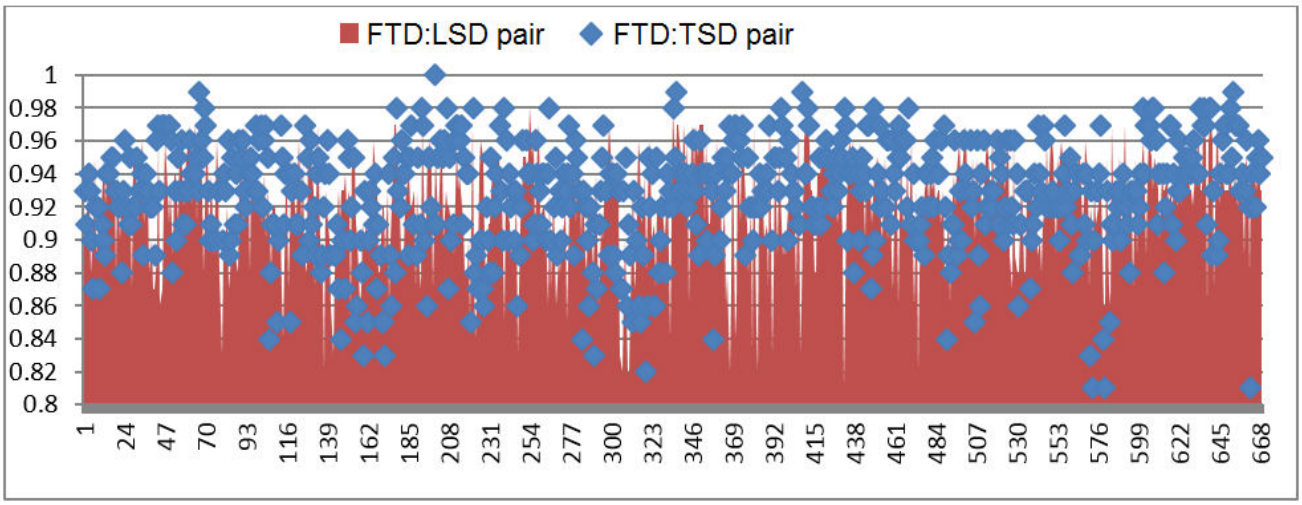

Fig. 4. A comparison of N-gram overlap between FTD:LSD pair and FTD:TSD pair at 75% summary.

change significantly, even when the summary size is changed. This higher similarity score indicates, in the longest subsequence, that is common between the two descriptions, there is no change in the ordering of tokens of words. A deeper examination of summaries revealed the reason that, both TextRank and LexRank decompose the full-length textual description into sentences prior to ranking each sentence. The decomposition is based on the punctuations, in particular fullstop. Subsequently, the sentences are ranked. When it is required to generate a summary of a text, say 50% summary, the 50% sentences with the higher ranking are selected to compose summary textual description. Note, the order of sentences, as maintained in full-length textual description, is retained by both algorithms while composing summary descriptions, i.e. the vocabulary within the sentence as well as the ordering in the sentences does not change. Due to that, the LCS similarity score is very high.

We return to investigate whether the higher average similarity score represent that TextRank and LexRank generated summary descriptions are identical to the full-length textual description or not. In Fig. 4 we plot a comparison of n-gram overlap between both pairs, (FTD:LSD pair at 75%; FTD:TSD pair at 75%). From the graph, two observations can be made: a) the FTD:LSD pair and FTD:TSD pair similarity scores do not overlap and therefore the identical average scores do not give a true representation of the similarity score. This can be observed from the fact that majority of the lines representing the similarity score of the FTD:LSD pair, does not touch the dots representing the similarity score of the FTD:TSD pair, in the graph. Secondly, the dots representing the FTD:TSD pair similarity score are closer to 1 compared to FTD:LSD pair. This shows, TextRank generated summary descriptions are more similar to full-length textual descriptions than LexRank generated summary descriptions.

V. CONCLUSION

In this paper we advocate the use of summary textual description of process models, to enhance the efficiency and effectiveness of process matching. However, a rigorous investigation is required to analyze the tradeoff (in terms of efficiency and effectiveness) between the use of summary textual descriptions and full length textual descriptions. To fulfil this requirement, we have generated a corpus of textual descriptions of process models, and corpora of summary textual descriptions of the same set of process models. In particular, in this paper we have discussed the process of generation of these corpora and specification of each corpus. Further, we used two algorithms TextRank and LexRank to summarize the textual descriptions. We applied two basic techniques, N-gram overlap and LCS to establish that two auto summarization algorithms generate different summaries. Our statistical analysis has shown that the datasets produced by two algorithms are significantly different and thus the choice of the summarization technique is non-trivial.

REFERENCES

[1] M. Weske, “Business process management - concepts, languages, architectures,” 2nd ed., Springer, 2012.

[2] A. Scheer, and F. Habermann, “Making ERP a success: Using business process models to achieve positive results,” Communications of ACM, vol. 43, pp. 57-61, 2000.

[3] Z. Yan, R. Dijkman, and P. Grefen, “Fast business process similarity search,” Dist. & Paral. Data., vol. 30, pp. 105-144, 2012.

[4] M. Rosa, H. Reijers, W.M.P Aalst, R. Dijkman, J. Mendling, M. Dumas, and L. Banuelos, “APROMORE: An advanced process model repository,” Expt. Sys. Appl. vol. 38, pp. 7029-7040, 2011.

[5] M. Becker and R. Laue, “A comparative survey of business process similarity measures”, Computer in Industry. 63, pp. 148 – 167, 2012.

[6] M. Kunze, M. Weidlich and M. Weske, “Querying process models by behavior inclusion,” Soft. Sys. Mod., vol. 14, pp. 1105-1125, 2013.

[7] M. Dumas, L. Banuelos, and R. Dijkman, “Similarity search of business process models,” In Proc. of IEEE Data Eng. Bullt., vol. 32, pp. 23-28, 2009.

[8] W.M.P. Aalst, A. Medeiros, and A. Weijters, “Process equivalence: Comparing two process models based on observed behavior.” In Proc. of BPM, Springer LNCS 4102, pp. 129-144, 2006.

[9] C. Ekanayake, “Consolidation of business process model collections,” PhD thesis at QUT, 2014.

[10] A. Awad, A. Polyvyanyy and M. Weske, “Semantic querying of business process models,” In Proc. of the IEEE-EDOC, pp. 85-94, 2008.

[11] H. Leopold, J. Mendling, and A. Polyvyanyy, “Supporting process model validation through natural language generation,” IEEE Transactions on Software Engineering, vol. 40, pp. 818-840, 2014.

[12] H. Leopold, H.V.D. Aa, F. Pittke, M. Raffel, J. Mendling, and H.A. Reijers, “Integrating textual and model-based process descriptions for comprehensive process search,” In Proc. of 17th International Conference on Business Process Modeling, Development, and Support, Springer LNBPI, vol. 248, pp. 51-65, 2016.

[13] O. Kurland and L. Lee, “Corpus structure, language models, and ad hoc information retrieval,” ACM Transactions on Information Systems, vol. 27, no. 3, pp. 13:1--13:39, 2009.

[14] K. Shahzad, K. Shareef, R. F. Ali, R. M. Adeel and A. Abid, “Generating Process Model Collection with Diverse Label and Structural Features,” unpublished.

[15] R. Abascal-Mena, R. Lema, and F. Sèdes, “Detecting sociosemantic communities by applying social network analysis in tweets,” Social Network Analysis and Mining, 5(1), pp. 1-17, 2015.

[16] J. Diesner, and K. M. Carley, “Exploration of communications networks from the Enron email corpus,” In Proc. of the Workshop on Link Analysis, Counterterrorism and Security, SIAM Intl. Conf. on Data Mining, pp. 3–14, 2005.

[17] A.B. Cedeno, P. Rosso, and J.M. Benedi, “Reducing the Plagiarism Detection Search Space on the Basis of the Kullback-Leibler Distance,” In Proc. of 10th International Conference on Computational Linguistics and Intelligent Text Processing, Springer LNCS, vol. 5449, pp. 523-534, 2009.

[18] [18] L. Bergroth, H. Hakonen and T. Raita, “A Survey of Longest Common Subsequence Algorithms,” SPIRE. IEEE Computer Society. pp. 39–48, 2000.

[19] Object Management Group (OMG). Business Process Model and Notation (BPMN) Version 2.0, http://www.omg.org/spec/BPMN/2.0/. last accessed on June 13, 2016

[20] Signavio www.signavio last accessed on 30 June 2016

[21] M. Snoeck, I. Oca, T. Haegemans, B. Scheldeman, and T. Hoste, “Testing a selection of BPMN tools for their support of modelling guidelines,” In Proc. of the Working Conference on Practice of Enterprise Modeling, Springer LNBIP, vol. 235, pp. 111-125, 2015.

[22] J. Mendling, H. Reijers, and W.M.P Aalst, “Seven Process Modeling Guidelines (7PMG),” Inf. & Soft. Tech., vol. 52, pp. 127-136, 2010.

[23] S. Zaheer, K. Shahzad and R. M. Adeel, “Comparing Manual- and Auto-Generated Textual Descriptions of Business Process Models,” In Sixth International Conference on Innovating Computing Technology., unpublished.

[24] R. Mihalcea and P. Tarau. “TextRank - bringing order into texts,” In Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), Barcelona, Spain, pp. 404-411, 2004.

[25] G. Erkan and D. Radev, “LexRank: graph-based centrality as salience in text summarisation,” Journal of Artificial Intelligence Research, 22, pp. 457-479, 2004.

[26] V. Gupta and G. S. Lehal, “A survey of text summarization extractive techniques,” J. Emerg. Technol. Web Intell., vol. 2, no. 3, pp. 258–268, 2010.

[27] B. Siniša and V. Štefanec, “N-gram overlap in automatic detection of document derivation,” The Future of Information Sciences, pp. 373-382, 2011.

[28] P. Clough, R. Gaizauskas, S. Piao, and Y. Wilks, “Measuring text reuse,” In Proc. of 40th Annual Meeting on Association for Computational Linguistics, pp. 152-159, 2002.

[29] A.B. Cedeno, P. Rosso, and J.M. Benedi, “Reducing the Plagiarism Detection Search Space on the Basis of the Kullback-Leibler Distance,” In Proc. of 10th International Conference on Computational Linguistics and Intelligent Text Processing, Springer LNCS, vol. 5449, pp. 523-534, 2009.