SN Computer Science (2023) 4:291
https://doi.org/10.1007/s42979-023-01742-z

ORIGINAL RESEARCH

A Machine Translation Like Approach to Generate Business Process Model from Textual Description

Riad Sonbol[1,2] · **Ghaida Rebdawi**[1,2] · **Nada Ghneim**[1,2]

Received: 28 May 2022 / Accepted: 2 February 2023 / Published online: 28 March 2023

Abstract

Modeling is one of the core tasks in Business Process Management (BPM). It represents the most critical step in the BPM life cycle and is considered as a time consuming and costly task. These challenges raised the question of how researchers can save this cost by building tools that could support modeling experts in their work to reduce the manual workload. In this paper, we propose a Machine Translation (MT) like approach to deal with the problem of generating a business process model based on a textual description. We chose to follow a semantic transfer-based MT approach. Our approach consists of two main phases: The natural Language Analysis phase and BPMN diagram generation. Natural Language Analysis phase aims to analyze the text and extract the required knowledge. One of the main outputs for this phase is a Concept Map which summarizes the concepts of the related domain and the relationships between these concepts. This map represents a background for our processing in the second phase where we try to generate the "translation", which is the BPMN diagram in our case. We achieve our goal in the second phase via a set of semantic, syntactic, and morphological manipulations. The approach has been implemented and evaluated usinga similarity metric based on the Graph Edit Distance. The results show that the proposed approach was able to generate models that are more than 81% similar to those created manually by a human, outperforming the state of the art in this topic.

Keywords Natural language processing · Business process modeling · Model extraction · Machine translation

Introduction

Modeling is one of the core tasks in business process management (BPM) [1]. It aims to create representations (usually called models) of the processes of an organization to understand them, documenting their details, analyzing their performance to determine opportunities for improvements, or representing the target process state [1].

Due to its challenges, modeling represents the most critical step in the BPM life cycle [2]. It is the most time consuming and costly task; it requires conducting a number of meetings, workshops, interviews between modeling experts and process performers to acquire the required knowledge in a highly interactive and repetitive approach. According to Herbst [2], building the as-is model consumes about 60% of the overall time spent in a workflow project.

On the other hand, in most organizations, the required information is available in textual forms; 85% of the information in companies are stored in unstructured documents—mostly textual [3]. This includes policies, reports, forms, manuals, knowledge management systems, and email messages [3]. In addition, most process performers are accustomed to expressing their needs in natural language [4]. These textual information represent potential sources of knowledge needed in building the model.

These facts raised the question of how researchers can save this cost by building tools that could support modeling experts in their manual workload. These systems will not replace modeling expert but will help them in creating models more efficiently in terms of the required time, cost, and quality [5]. According to Friedrich et al. [6], substantial savings are possible by providing such automation tools.

Natural Language Processing (NLP) plays an essential role in dealing with the available textual documents to extract the models: we need to handle many complex

✉ Riad Sonbol
riad.sonbol@hiast.edu.sy

[1] Department of Informatics, Higher Institute for Applied Sciences and Technology, Damascus, Syria

[2] Faculty of Information and Communication Technology, Arab International University, Daraa, Syria

challenges including extracting tasks, ordering them, and dealing with concurrency, and loops [7]. In addition, there are many problems related to the textual input which include the various syntactic and semantic ambiguities [8], irrelevant sections which could be found in the text for different reasons (like giving a detailed example) [6], and complex NLP problems like anaphora resolution. These challenges would be faced even when dealing with a very short text such as the following process "the process of choosing leads:"

"First, the Manager checks the open leads. Afterwards, he selects the top five ones. He then tells his Sales Assistant to call the contact person of the leads. The Sales Assistant calls each customer. If someone is interested, he sends a note to the Manager. The Manager then processes the lead. Otherwise, he calls the next customer."

We will use this simple process later in Sect. "The Proposed Approach: A Machine Translation Like Approach" as an example to clarify the output of each step in our approach.

In the next section, we will give a quick overview of most related works on this topic. Section "Machine Translation: An Overview" presents a short introduction to the main machine translation approaches. Our approach is presented in Sect. "The Proposed Approach: A Machine Translation Like Approach" and its evaluation results are shown in Sect. "Evaluation". Finally, we conclude our paper in Sect. "Conclusion"

Related Works

The problem of generating models from texts has received increasing attention in the last decade. Many approaches have been proposed to deal with this problem based on NLP techniques.

One of the most important contributions in this field is the work of Friedrich et al. [6] who proposed a transformation approach for the automated generation of a business process model in BPMN format from natural language text. The approach can deal with text consisting of full, grammatically correct sentences, ordered in a correct sequential with no irrelevant information. The approach splits up the text into individual sentences, then parse each sentence using Stanford Parser [9]. Depending on the grammatical relations, the approach extracts actors and actions, combines them, and adds them to a data structure called World Model. This model is a variation of the CREWS scenario model [10] and represents intermediate information. The approach uses some semantic resources such as WordNet and FrameNet and includes a self-developed anaphora resolution component. In the last phase of Friedrich's approach, the information contained in the World Model would be transformed into its BPMN representation. Authors claim that the generated models are 76% similar to those created manually by a human-based on the Graph Edit Distance metric.

Gonçalves et al. [11] combine the Group Storytelling technique (which has been originally proposed by Santoro [12]) with Text Mining and NLP techniques. The approach gets a narrative structure that is mainly composed of events (flow and participants). These texts are tokenized, annotated with POS tags, then parsed by a shallow parser. Syntactic-based templates would be applied to extract activities, actors, and actions. These information would be stored in a variation of the CREWS scenario model. The approach uses some keywords (connectors) to determine the process flow. The approach was further tested with a course enrollment process modeled by students. Although the final model uses BPMN format, but might not be complete and could contain undesirable situations in the discovered process models, such as activities without actors [11].

Ghose et al. [13] propose the Rapid Business Process Discovery (R-BPD) framework. This framework can query heterogeneous information resources and rapidly construct proto-models to be incrementally corrected by an analyst. The approach analyzes the text in two different techniques: template-based extraction technique and information extraction technique. The template-based extraction technique uses templates of commonly occurring textual cues for processes, such as if-then pattern. The second technique uses an information extraction-based approach where Natural Language Toolkit (NLTK) [14] was used to annotate the text with POS tags and parse it to conduct the syntax tree. Activities, objects, and actors were extracted based on the resultant syntactic information. Then, sequence flows are discovered through a predefined list of words. The output of this approach is BPMN snippets rather than a fully connected model.

Sinha [15] employs a linguistic analysis engine based on the UIMA framework to extract the model from use cases. Texts are tokenized, lemmatized, tagged then parsed by a shallow parser using a Finite State Transducer. The system annotates each verb with its related concept using a manually created domain dictionary. Next, a specialized version of the anaphora resolution system described in [16] was applied. For every actor, a swimlane is created. Afterward, the process elements were added and the process model was built sentence by sentence.

Another approach presented by Epure et al. [17] analyzes textual input for archaeological processes with a single process instance per text. The text is normalized then analyzed sentence by sentence. This approach generates a parsing tree for each sentence using Stanford parser, then identifies transitive verbs using WordNet and VerbNet to extract activities. Later, pre-defined domain-specific rules were applied to identify the different relationships between activities. The

approach returns chains of relationships without merging them in a complete model.

Honkisz et al. [18] present a concept of a new method for extracting the business process from natural language text through an intermediate process model based on the spreadsheet representation. The method of obtaining this model is based on the syntactic analysis of the business process description and extracting Subject-Verb-Object constructs, which can be later transformed into process activities.

Van der Aa et al. [19] approach for the automatic extraction of declarative process models from natural language. The approach used Natural Language Processing (NLP) techniques that identify activities and their inter-relations from textual constraint descriptions. Later this work has been extended by the same authors in [20] where they presented an interactive approach that takes vocal statements from the user as input and employs speech recognition to convert them into multi-perspective, declarative process models.

Other close works focus on the automatic comparison and alignment of process information in semi-structured and unstructured formats [21]. Henrik et al. [22] proposed a process querying technique that can search repositories of both textual and model-based process descriptions. The approach automatically extracts activity-related and behavioral information from both descriptions types and stores it in a unified data format based on a Resource Description Framework (RDF). Leopold et al. [23] propose the use of natural language processing and machine learning for detecting candidate activities from a textual description of processes. Their technique automatically identifies whether a task described in a textual process description is (1) a manual task, (2) a user task (interaction of a human with an information system) or (3) an automated task.

Recently, many works explored the usage of deep learning approaches to handle process models extraction problems. Feng et al. [24] used a deep reinforcement learning framework to automatically extract action sequences from texts. Their architecture defines two Q-functions associated with CNN networks to extract actions and model actions Sequences. Qian et al. [25] formalized the process model extraction task into the multi-grained text classification problem and design a new hierarchical network to model the conditional relation among multi-grained tasks.

Machine Translation: An Overview

Machine Translation (MT) is one of the common problems in NLP domain. MT systems use different techniques to obtain a target language text automatically from a source language text. Two major approaches have been proposed to deal with this problem: Statistical Machine Translation (SMT) and Rule-Based Machine Translation (RBMT). SMT approach tries to build ML systems by learning from parallel corpora aligned at the sentence level, while RBMT approach applies a set of linguistic rules in three phases: analysis, transfer, and generation which could be performed at different linguistic levels (morphology, syntax or semantics) [26]. Recently, many deep learning based techniques have been proposed to improve machine translation systems [27]. In this section, we will focus on RBMT approaches since it is related to our work.

Traditionally, RBMT approaches could be classified into three categories:direct translation approach, transfer translation approach, and the interlingual approach. The differences between these three approaches could be represented through a triangle called the Vauquois Triangle [28]. As we can see in Fig. 1, the main dissimilarities are related to the levels of analysis which should be done to complete the translation task.

The direct translation approach is based on the word level (the shallowest level of processing). Usually, a single word or a string of words (n-gram) is taken from the source language and looked up in a bilingual dictionary between the source and the target languages to retrieve the translation. This step could include some morphological preprocessing to obtain the base form of words. Next, the translated words would be rearranged to consider the preferred word order in the target language. Typically, systems built using this approach consist of a large bilingual dictionary and a program for analyzing and generating texts [29].

A Transfer-based machine translation approach starts by analyzing the source language text. The analysis could be only at the lexical and syntactic level and called syntactic transfer approach, or could cover the semantic level of processing and called semantic transfer approach. In both cases, the obtained representation is transferred to the target language using rules which map the source language representation into their target language equivalents (such as converting the syntactic tree from the source language to the target language). Finally, the text in the target language is generated.

Interlingual approach is based on extracting an abstract language-independent representation for the meaning of the text in the source language and reproducing it in the target language. The main idea of this approach is that MT must involve an 'understanding'of texts' content in general. The approach starts with lexical, syntactical, and semantical processing for source text and interpreting it into a canonical interlingual. Later, this interlingual would be used to generate the target language text.

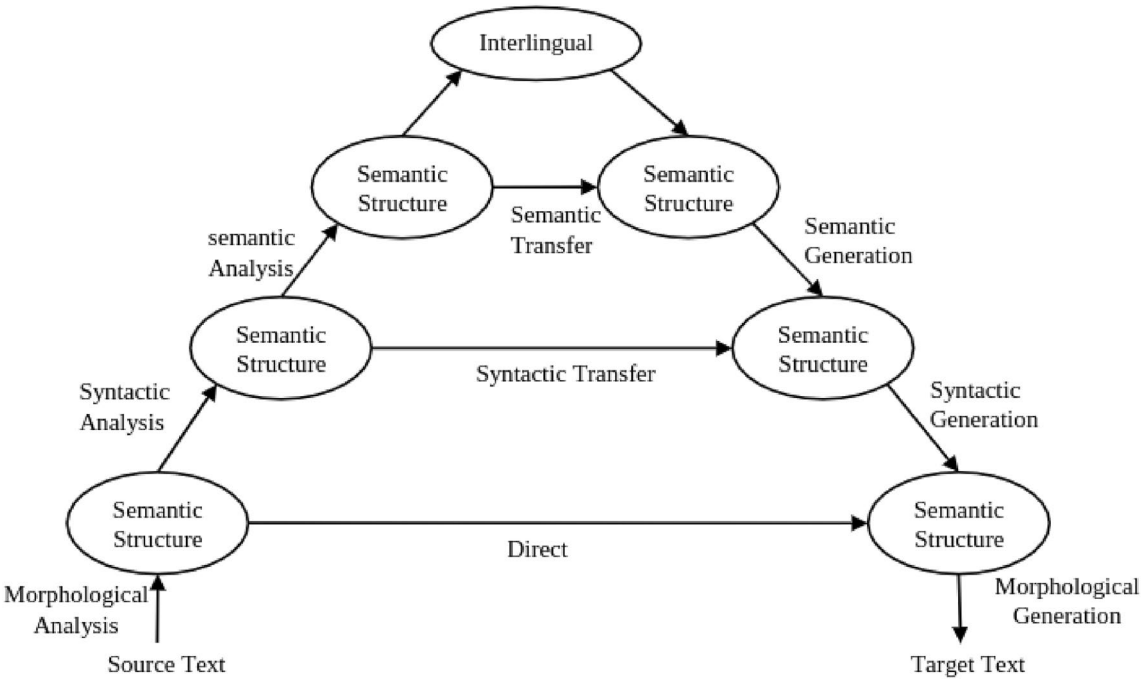

Fig. 1 Vauquois triangle (adapted from [28])

The Proposed Approach: A Machine Translation Like Approach

We propose an MT like approach to deal with the problem of generating a business process model based on a textual description. Since both of the textual description and the BPMN diagram for a process are finally languages to express the process, we can see the work of modeling expert as a translation task from the natural language—which is understood for business domain experts and all the process performers-to BPMN which represents the language of software engineers and modeling experts.

In the next section, we will explain how we can see our problem as a machine translation task by showing the different components for each language (the text and the BPMN) from a language processing point of view, then we will explain our approach in details.

Language Levels: Text vs. BPMN

Traditionally, the processing of any written language goes through three main levels:morphological, syntactic, and semantic. Each of these levels focuses on a specific "context" in the language. In natural language, there are standard definitions of these contexts and levels of processing, but when dealing with visual languages, we find quite different points of view, especially at the syntactic and semantic levels. In the following three paragraphs, we will start with a generic definition of each level, then adapt these definitions to natural and modeling languages. It is worth to mention that the proposed levels are somehow simplistic since the main aim for these levels is (1) to separate what could be considered "morphological" from what is "syntactical" or "semantical" while "translating" the text into a BPMN diagram, (2) and to be used when going throw the descending side of Vauquois Triangle (Fig. 1)

In Morphological Level,morphology refers to the analysis of the smallest meaningful unit of a language. Therefore, the context of processing is the basic element in the language. In natural languages, basic elements are words, while in BPMN basic elements are tasks, events, gateways, data objects, sequence flows, pools, and lanes (Fig. 2). Each of

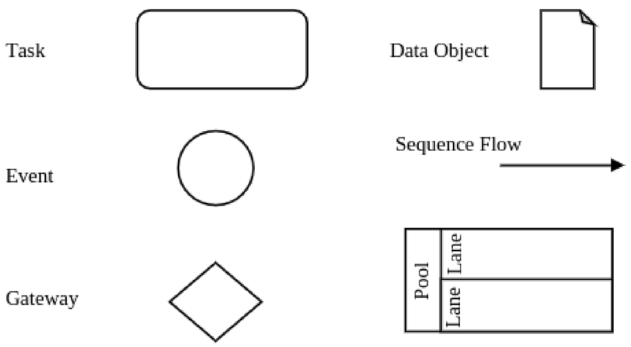

Fig. 2 Generic forms for BPMN components

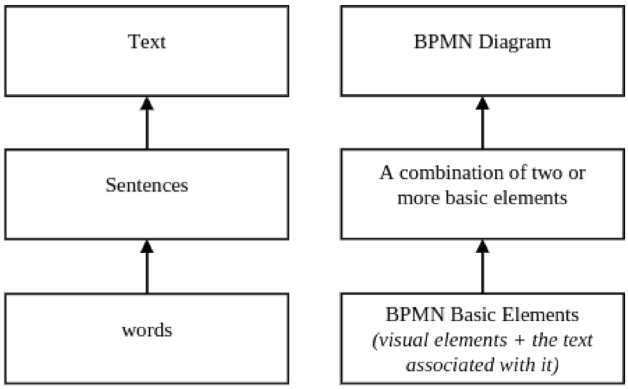

Fig. 3 Main levels in natural languages vs BPMN language

these elements consists of a combination of visual elements and the text that is associated with these elements.

Syntactic Level studies the arrangement of basic elements in the language to create a well-formed independent statement, question, request, command, exclamation, etc. In natural language, the context of processing is the sentence. In BPMN, the "BPMN sentence" is formed by two or more basic elements. For example, a lane combined with a task form a well-formed statement that says who is responsible for doing a task, two tasks with a sequence flow between them form a statement related to the order of two tasks, etc. In natural languages, a text or any part of it is syntactically correct if all of its sentences are syntactically correct. Similarly, a BPMN or any part of it is syntactically correct if all possible BPMN sentences in the diagram are syntactically correct.

Semantic Level deals with the meaning of language units and sentences in natural languages or the whole diagram in the BPMN language. In NLP and BPMN, the key concern in semantic level is how the meaning of larger units (sentence and words) could be obtained from the meaning of smaller units (words and phrases), and how this meaning could be transferred accurately to the target language.

The next figure (Fig. 3) shows a summarization of the main levels in natural languages and the corresponding levels in BPMN diagrams.

The Approach

Generating the BPMN diagram from the textual description of a process is a complex task that needs a deep level of processing in both syntactic and semantic levels of processing. Therefore, the direct transfer approach is not the best solution. In this paper, we use a semantic transfer-based approach. Our approach consists of two main phases: natural language analysis and modeling language generation.

The natural language analysis phase aims to analyze the text and extract the required knowledge from it. In addition to the syntactic analysis results, one of the main outputs for this phase is a Concept Map which summarizes the concepts of the related domain and their relationships. We generate this Concept Map using both syntactic and semantic processing on the whole text. This map represents a background for our processing in the second phase where we try to generate the "translation" i.e. the BPMN diagram in our case. We achieve our goal in the second phase by applying a set of semantic, syntactic, and morphological post processing procedures. At the semantic step, we generate a "Text Graph" that represents the main paths in the model regardless of any consideration for the syntactic and morphological constraints. Later, tuning steps would be applied to consider the grammar and the morphology of the target language (BPMN).

One of the major points in the proposed approach is separating the problem of generating a semantically correct diagram i.e. a diagram that reflects all main paths in the text from the problem of generating a syntactically correct BPMN diagram. The main challenge in the first problem is having an overall understanding of the whole text by recognizing the main paths which could be scattered and overlapped because of the different conditions and cases in the process text. On the other hand, the main challenge in the second problem is translating that overall understanding into a correct BPMN by dividing the different paths into syntactically correct BPMB elements.

To facilitate the process of transferring the semantic from natural language text into the business process model, we proposed the usage of a concept map. We designed a concept map extraction algorithm to overcome the limitation of previously proposed semantic representation. Although we are inspired by the "world model" structure which has been proposed by Friedrich et al. [6], our concept map is rather different from their structure. The "world model" of Friedrich et al. is mainly syntax-oriented, action-centric, and defined in the context of each sentence [30], it is borrowed from the field of robotics where it is usually used to govern robotic actions [31]. It consists of four main elements: Actor, Resource, Action, and Flow. It is constructed based on an algorithm that takes each sentence in the text, decomposes, analyzes it, then the action of each phrase is extracted with its different parts (Actors, and Resources). On the other hand,our concept map is mainly semantic-oriented, concept centric, and defined in the context of the whole text. It is borrowed from the field of education where it is usually used to represent the concepts the learner should know at the end of a specific course or program [32] i.e. we mainly focus on answering the question of "what are the main concepts in the process" then "what are the relations and actions between these concepts.

The next figure (Fig. 4) shows a general overview of our approach. In the next two sections, we will describe each of the two main phases in more detail.

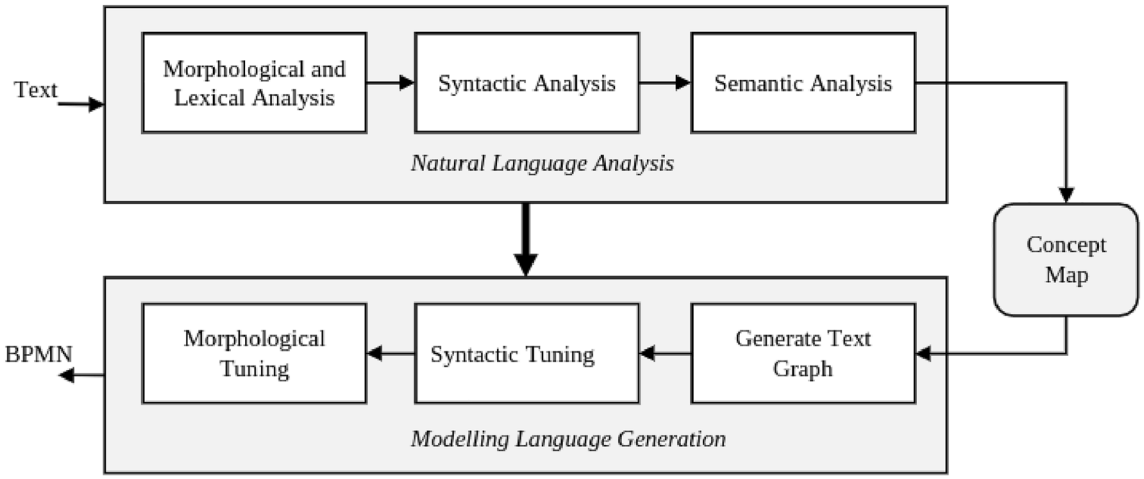

Fig. 4 General overview of the proposed approach

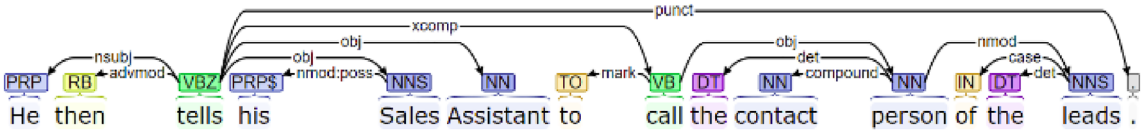

Fig. 5 Sample dependency tree using stanford CoreNLP

Phase 1: Natural Language Analysis

As mentioned above, the main target in this stage is analyzing the text and extracting the required knowledge. We do this through three steps:

1. Morphological and Lexical Analysis: In this step, we split the text into sentences and the sentences into words. Then, we extract words' lemmas using Stanford Lemmatizer. Lemmas will be used in the next steps.
2. Syntactic Analysis: First, we use Stanford pos-tagger [34]to tag each word with its suitable part of speech tag which gives the syntactic role of the word (such as Plural Noun, Singular Noun, Adverb, Adjective...). These syntactic information will help us determine the entities and the concepts in the next semantic processing step. Then, we use Chen and Manning parser [33] to generate the dependency tree for each sentence. According to the authors, the parser outperforms other greedy parsers using sparse indicator features in both accuracy and speed. It can parse more than 1000 sentences per second which makes it suitable for our needs in this work. The following figure (Fig. 5) shows a sample dependency tree:
3. Semantic Analysis:This step of analysis consists of two components: semantic tagging and Concept Map extraction.

   (a) **Semantic Tagging:** One of the main problems in generating BPMN is to connect correctly the different activities. Therefore, we extract semantic tags from sentences to guide us while connecting two sentences (which will represent a set of activities). Tagging process uses a lexicon-based approach to tag words using the following tags

   - "Decision": tags words that represent or trigger a decision taken from one of the actors in the process. This tag could be used for words such as "assess", "check", "test", "determine" or expressions such as "must... or...", "can... or...".
   - "Case": tags words that represent or trigger processing a special case based on a condition, such as "if", "otherwise"...
   - "Go-to": tags words which represent or trigger repeating certain steps i.e. going back to one of the previous steps, such as "sent back", "send something back", "procedure is repeated"...

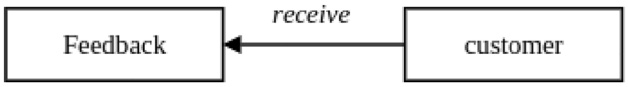

Fig. 6 Simple Concept Map

- "Merge": tags words which represent or trigger merging more than one path, such as "in any of the cases", "in all cases"...
- "Seq": tags words that represent or trigger a direct sequence, such as "then", "after that"...
- "Parallel": tags words which represent or trigger parallelism, such as "while", "meantime"...
- "End": tags words that represent or trigger an end of the process, such as "end of the process"...

The tagging process is implemented using lexicon at lemma level i.e. we tag a word W using the tag T if the lemma of W exists in the lexicon of the tag T. This lexicon is expanded semantically using WordNet to cover all possible synonyms. The final lexicon consists of 43 expressions.

(b) **Concept Map Extraction:** In this step, we will try to generate a "Concept Map" automatically from the text. Concept Map (CM) is a context-dependent knowledge which consists of a set of concepts connected using relationships. Concept Maps have been proposed by Joseph Novak [32] in the context of representing the emerging science knowledge of students [34]. In education, Concept Map is usually used to represent "what the learner knows" or "what the learner should know". In our case, we will use CM as an intermediate semantic representation which describes "what business analyst should know before modeling the described process". It will represent the knowledge in the whole text i.e. a kind of domain ontology.

Concept Map could be represented as a graph where concepts are enclosed in nodes and relations are represented by labeled links between related nodes (i.e. concepts). These relationships with their related concepts form propositions (or semantic units). For example, in the next figure (Fig. 6) a simple Concept Map is represented, where "Feedback" and "Customer" are two concepts related to each other by a relationship labeled "receive". The relationship with the two concepts forms the propositions "customer receives feedback".

This algorithm starts by detecting noun phrases boundaries using a rule-based tagger, then concepts are extracted using a set of templates. After that, the algorithm merges the different forms (aliases) of the same concept in one entity. Next, we extract relationships between concepts based on a set of syntactic rules. Finally, we merge anaphoras with their related concepts. In the following, we will explain in details these steps:

(i) Noun Phrase Boundary (NPBT) Tagging: In this step, we use a rule-based tagging approach to tag words with three possible Noun Phrase Boundary (NPB) Tags. These three tags will be used later to determine the boundaries of noun phrases:

- "SE" tags the first word in the noun phrase.
- "IE" tags the other words in the noun phrase (except the first word).
- "O" tags the remaining words in the text which could not be part of a noun phrase

(ii) Concepts Detection: In this step, we scan the text to extract the noun phrases using the result of NPB-tagger. Any sequence of words with tags SE and IE represents a possible concept. Anaphora cases such as "he", or "she" are considered as possible concepts in the previous tagging step. To distinguish between the different occurrences of the same pronoun in text, we add a location suffix to each anaphora. Forexample, if we have the pronoun "he" in the second sentence, we add the concept he_2 where 2 is the location suffix.

(iii) Alias Detection: Some concepts could appear in different forms in the text. In this step, we merge all these possible forms into one concept. To do that, we detect the possible merges by checking if any concept is an ending of another one, regardless of stopwords or morphological affixes. For example, the two concepts "response comment" and "the comment" are candidates for a merge since the second concept is an ending of the first one. Practically, we found that this heuristic is sufficient to detect the possible merges since users usually use the same words to express the same concept when describing a business process. When there is more than one possible merge, we choose the nearest one based on the number of words. After merging concepts, we consider the longest form (in terms of the number of words) as the main concept, and we consider all other forms as aliases.

(iv) Relationships Generation: after extracting the concepts from the text, we connect them by semantic relationships in three ways:

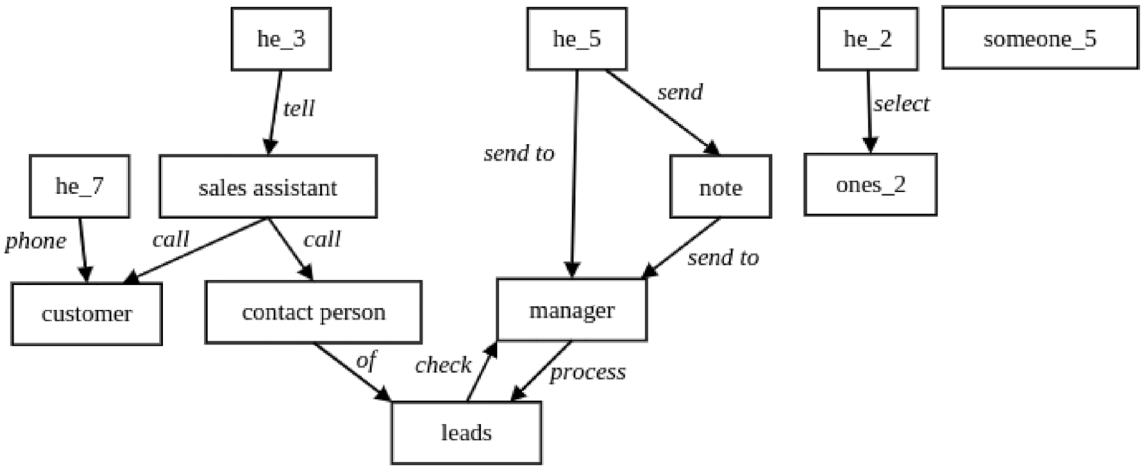

Fig. 7 The Output of steps 1–4 on "the process of choosing leads"

- If a concept X starts with a concept Y, we add a relationship from X to Y labeled by "related to" like the case of "confirmation" and "confirmation document".
- If there is a verb connecting two concepts[1] by subject-object relationships in the dependency tree, we create a relationship between these two concepts labeled by the verb. Here, we consider different cases of conjunctions, such as: "X can accept or reject Y", "X process Y and Z", "X and Y process X".
- If there is a direct relationship between two concepts in the dependency tree, we reflect it as a relationship in the Concept Map. For example, in the phrase "the contact person of leads" we create a relationship between "contact person" and "leads" labeled by "of".

(v) Anaphora Resolver: We resolve anaphora using the extracted concepts and relationships. The next figure (Fig. 7) represents the result of applying the previous steps on "the process of choosing leads":

"First, the Manager checks the open leads. Afterwards, he selects the top five ones. He then tells his Sales Assistant to call the contact person of the leads. The Sales Assistant calls each customer. If someone is interested, he sends a note to the Manager. The Manager then processes the lead. Otherwise, he calls the next customer."

The previous Concept Map contains 6 anaphoras: he_2, he_3, he_5, he_7, ones_2, and someone_5. For each anaphora, we consider all previous concepts, in terms of their occurrence in the text, as possible resolvers. Our objective in this step is to rank these possible concepts and to choose the most suitable one. We do that by comparing the context of the anaphora and the context of each possible concept. The context is defined by its in-relationships (relations from other nodes), and its out-relationships (relations to other nodes). For example, the context of "he_7" could be defined by its relationship with "customer", i.e. he is the person who "phones" the "customer". In the same way, "he_5" is the person who "send" a "note", and "send sth to" the "manager".

We give each possible concept a score for being a resolver for the anaphora. This score is calculated based on the maximum similarity between the relationships of each possible concept and the relationships of the anaphora. WuP similarity (or Wu-Palmer similarity) is used to calculate the similarity between two relations [35].

The final Concept Map for the previous process is shown in Fig. 8.

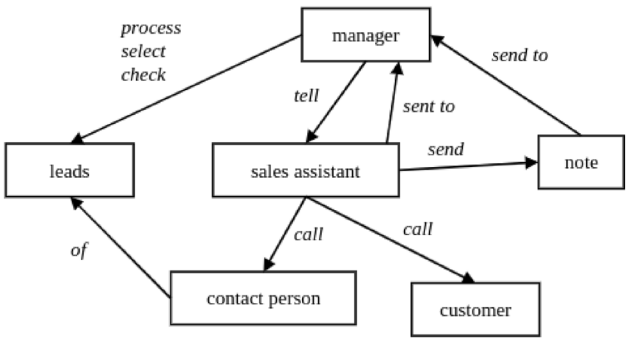

Fig. 8 Final output of applying concept map extraction algorithm on "the process of choosing leads"

[1] We consider that a verb is connecting two concepts if it connects a word from the first concept with a word from the second one.

Phase 2: Modeling Language Generation

After finishing the natural language analysis phase, we generate a semantic representation of the knowledge in the text (i.e. the Concept Map). In addition, we analyze the sentences morphologically and syntactically. Just like what business analysts do while constructing the BPMN diagram, our system will "put the Concept Map in its minds" while generating the model. In the current phase, we use all these information while building the BPMN. This phase could be divided into three levels:

1. Text Graph Generation: At the first level, we focus on the semantic correctness of the generated BPMN graph by discovering the main paths in the process from the text. Therefore, we ignore all details related to activities' labels since we consider it as a morphological task (see Sect. "Language Levels: Text vs. BPMN").

   To achieve this goal, our approach uses sentences as a sub-process (which could simply consist of one activity), then tries to find paths between these sub-processes. The output is a graph connecting these sentences (we will call it "Text Graph").

   Text Graph is generated using the following algorithm:

   (a) Initialization: In this step we initialize a graph G=(V,E) which consists of:

   - A vertex V[i] for each sentence i in the text.
   - An edge between every two vertices representing two consecutive sentences.

   (b) Connecting "Decision" and "Cases" vertices Decision vertices are the vertices where its related sentence contains a word tagged by "Decision" based on the output of the previous semantic tagging step. In the same way, we define Case vertices.

   The aim of this step is connecting each Case vertex with its related Decision vertex. We do that by defining the context of the Case and the context of all previous Decisions. Each context consists of the concepts surrounding the words tagged by "Decision" / "Case". Concept Map helps us in extracting these concepts. We evaluate the similarity between each Case/Decision pair based on the distance between "Decision" context concepts and "Case" context concepts within the concept map. Finally, we connect the Case node with the Decision node with the most similar context using an edge labeled as "Cond".

   The algorithm is detailed in Algorithm 1. As described in step 4 (Algorithm 1), we chose 5 words length after (or before) each "decision" or "case" words to define "decision" context and "case" context. The value 5 has been chosen as a distance threshold by looking at many previous works in natural language processing [36, 37] which chose 5-grams in their applications. The choice of N in n-gram features reflects the maximum meaningful length sequence of words. In our case, this distance threshold reflects the number of words that might be related (with high probability) to the context. For this reason, we have chosen 5 after testing its efficiency experimentally.

Algorithm 1 Detailed Steps for Connecting "Decision" and "Cases" vertices

1: For each case vertex V[c] we get all decision vertices VD_c before it.
2: VD_c=V[d]: V[d] is a decision vertex and d¡c.
3: Find the contextual similarity between the V[c] and each V[d] in VD_c in the following way:
4: sim(V[d], V[c]) = 1.0/ (1+Distance(V[d], V[c])) **Where**:
- Distance(V[d], V[c]) = Min ({len(c1, c2): c1 from $\text{Context}_V[d]$ and c2 from $\text{Context}_V[c]$})
- len(c1, c2) represent the path between c1 and c2 in the Concept Map.
- $\text{Context}_V[d]$= cpt : cpt is a concept in the sentence d and the concept is exist after the word with the tag 'case' with less than 5 words between them

5: Delete all edges from any vertex in the graph to V[c].
6: Connect V[c] with the most contextual similar V[d] with a directed edge from V[d] to V[c].

(c) Adding Repeat edges: The trigger for this step is the existence of a go-to vertex in the graph i.e. a vertex that includes the "Go-to" tag in its sentence. To add repeating edges, we connect each Go-to vertex with one of the previous vertices in terms of the order of their occurrence in the text. Thus, we scan all previous sentences starting from the closest to choose the most similar context. We compare the context of Go-to (i.e. the concepts surrounding the words with the tag go-to) with the context of the destination sentence's head (i.e. concepts which exist at the beginning of the sentence). As a result, a loop is formed by adding an edge between the most contextually similar previous node and the go-to node.

The algorithm is detailed in Algorithm 2.

Algorithm 2 Detailed Steps for Adding Repeat edges

1: For each go-to vertex V[g] scan the vertices V[t] where t goes from g-1 to 1 one by one.
2: $\text{Context}_V[g]$ = {cpt: cpt is a concept in the sentence d and the concept is exist after the word with the tag 'go-to' with less than 5 words between them}.
3: ContextV[t] = {cpt: cpt is a concept in the sentence d and the concept in the first 5 words between them}
4: Distance(V[g], V[t]) = Min ({len(c1, c2): c1 from $\text{Context}_V[g]$ and c2 from $\text{Context}_V[t]$ }) where len(c1, c2) represent the path between c1 and c2 in the Concept Map.
5: if Distance(V[g], V[t]) ==1:
6: Connect V[g] with V[t] with a directed edge from V[g] to V[t]
7: Break the for cycle.

(d) Process End Vertices: For each node containing a word tagged as 'end', we delete all out edges.
(e) Process Merge Vertices: Merge cases are needed mainly as a result of the second step which produces some paths that are not connected with any other node (because of different paths in the flow). For each vertex V_m containing a word tagged as 'merge', we connect each vertex V_t with this vertex when (1) V_t has no out edges, (2) V_t is not an end node, and (3) $t < m$.
(f) Sequence Processing: For each vertex V_s containing a word tagged as 'sequence', we connect V_s with V_{s-1} and delete all incoming edges to V_s. Figure 9 shows a sample TextGraph which represents the result of applying these steps on "choosing leads process".

2. BPMN Syntactic-based tuning

At this step, we convert the TextGraph to a syntactically correct BPMN. We apply the following steps to create the Business Process Diagram:

(a) Change each vertex in TextGraph to a BPMN task and label it with the TextGraph node related sentence.
(b) Change each relation between two vertices in the graph to the BPMN sequence flow between the two related tasks.
(c) Create a BPMN start event and connect it with the first BPMN task.
(d) Create a BPMN end event and connect all open TextGraph vertices (i.e. nodes with no out-edges) with it.
(e) Convert each conditional case in TextGraph to a BPMN exclusive gateway.
(f) Convert each parallel case in TextGraph to a BPMN parallel gateway.
(g) Split complex tasks: Some TextGraph nodes might contain multiple tasks. To simplify a complex task, we apply the following steps:

(i) Extract verbs from node label.
(ii) Extract splitters, such as "otherwise", ",", ":", "and", "if", etc.

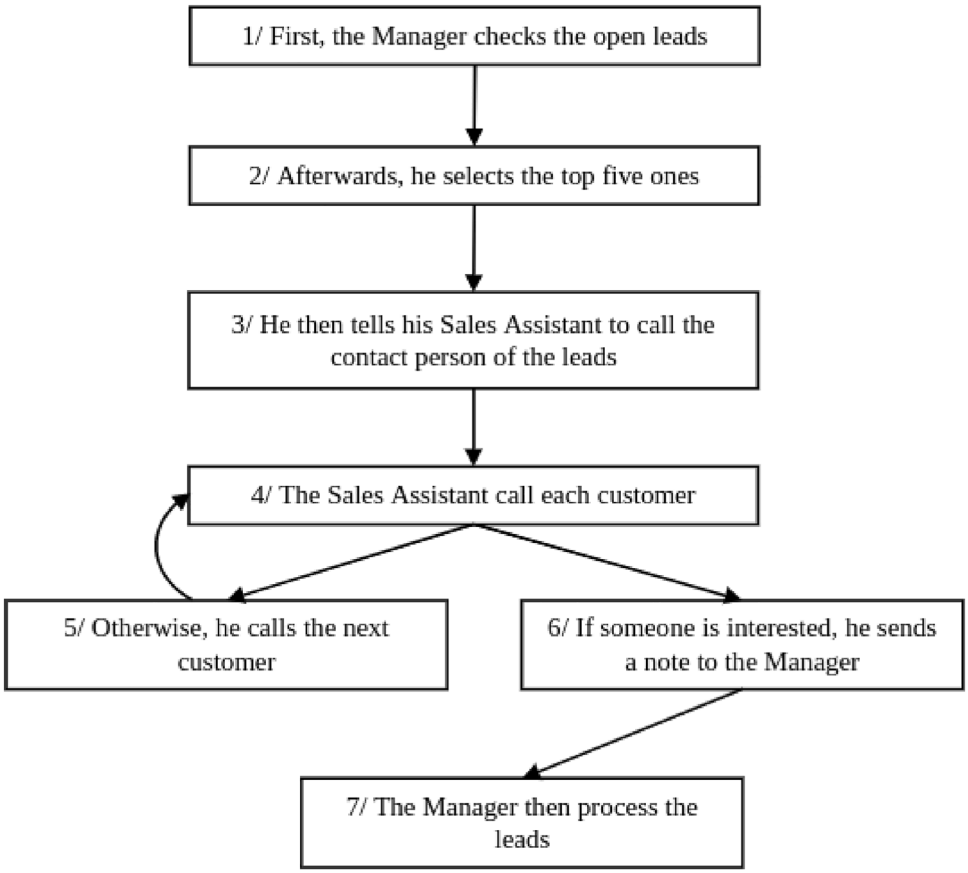

Fig. 9 Applying Text Graph Generation on "the process choosing leads"

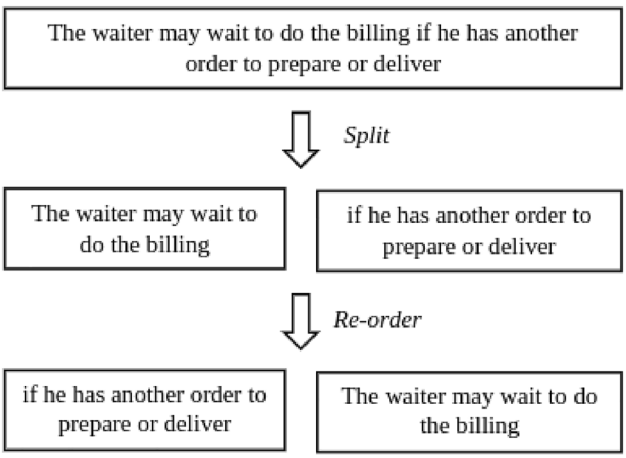

Fig. 10 Splitting Complex Tasks

(iii) Using splitters and verbs, we extract the chunks in this label. A chunk consists of a set of words between two consecutive splitters and contains at least one verb.
(iv) Reorder chunks: we define a set of rules to reorder the extracted chunks. For example: "A if B" contains two chunks "A", "if B". We reorder these two chunks to be "if B", "A" since the condition should appear before the action (see the example in Fig. 10)
(v) Replace the node by creating a task for each chunk, then reconnect the flows with these new tasks.

(h) BPMN Morphological-Based Tuning:
After converting the TextGraphto the syntax of BPMN, we apply a set of procedures to rewrite the different labels (activities, edges, and lanes) in the previously constructed diagram. These procedures focus on the correctness of the simplest units in the BPMN diagram, for this reason,we call it "morphological-based tuning" (See our definition for morphology in BPMN language in Sect. "Language Levels: Text vs. BPMN").

(i) Activities Labels: To enhance the activity labels, we apply a set of syntactical templates to extract a phrase that represents the task and exclude other details. For example, "He selects the top five ones" will be "select the top five ones".
(ii) Edges Labels: Flows after exclusive gateways should be labeled by meaningful

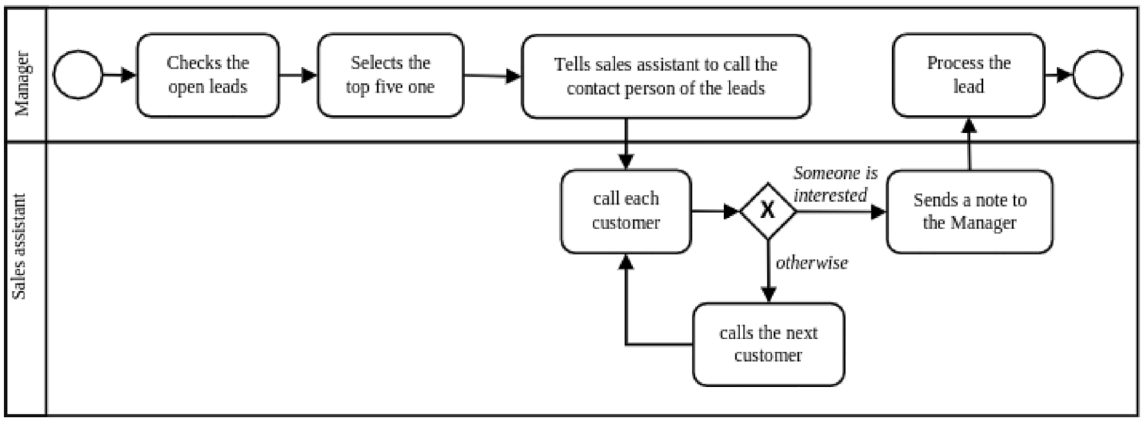

Fig. 11 The generated BPMN for "the process of choosing leads"

conditions. We determine these labels by applying a set of rules which cover these two main cases:

(A) Explicit Conditions: When the condition is expressed clearly like in the example "if A, then B", we get the label directly from the condition part (A part in the last example).
(B) Implicit conditions: When the condition is indirectly mentioned like in the example "it could be accepted or rejected, in the former case.... in the latter case...", we extract the condition using the output of anaphora resolution processing.

(iii) Lanes Labels: Each lane represents an actor in the process. Hence, it should have been extracted as a concept in the Concept Map. Using the Concept Map and the output of the dependency tree, we extract the concept which plays the "subject" role in each activity. Concept Map helps us in (1) standardizing lanes labels (for example, both info department and department should be represented by the same concept i.e. the same lane since we merged all aliases in one concept while building the concept map), (2) dealing with anaphora actors (they, it..), since these anaphoras are "aliases" of other concepts, (3) exclude unreal actors, such as "procedure", "process", "activity", since they are not concepts in the Concept Map.

To deal with activities with no actor, we assign them to the actor of the previous activity in the model.

The output of this phase is the final result of the proposed approach. Figure 11 illustrate the output of applying the approach on "the process of choosing leads".

Evaluation

To evaluate our work, we implemented the proposed approach in java and used it to conduct the experiments. It took in our implementation 150 s to complete processing all the dataset (2 to 6 s per text) and generate the BPMN diagram for each of them (using 4.5 GHz Intel Core i7 and 16GB RAM). We applied two experiments: In this first experiment we evaluated the similarity between the generated models and the manually constructed one, we used Friedrich dataset [30] which consists of 47 textual processes to evaluate the similarity. In the second experiment, we conducted an expert evaluation on a sample of 5 diagrams. In the next two sections, we will discuss the details and the results of each experiment.

Experiment 1: Similarity Evaluation

To measure the similarity between the auto-extracted models and the manually constructed ones, we use Graph Edit Distance (GED) which has been proposed by Dijkman [38] and can be applied for different aspects of a process model [4]. In this experiment, we will use a dataset collected by Friedrich [30]. This dataset is one of the best known datasets in this domain in terms of its size and variety, as it consists of 47 textual process descriptions (in total, 432 sentences) collected from different sources: Academic (15 models), Industry (9 models), Textbook (9 models), and Public Sector (14 models). The dataset represents processes from different domains (computer, hotels, manufacturing, HR, etc.) and is divided into 10 groups according to its source.

Table 1 The detailed results of our approach on Friedrich dataset

Group number	Number of models	Source type	Our approach (%)
1	4	Academic	81.72
2	2	Academic	79.51
3	8	Academic	84.42
4	1	Academic	64.31
5	4	Industry	71.95
6	4	Industry	69.72
7	1	Industry	79.45
8	3	Textbook	77.19
9	6	Textbook	73.77
10	14	Academic	90.77
Total	47	Academic	81.21

The evaluation process starts with matching activities labels in both models (the manual and the generated model) using the edit distance algorithm. This matching produces pairs $M_i = (n, m)$ of similar activities with a similarity value $sim(M_i)$ or $sim(n, m)$. Depending on these pairs the similarity is calculated using the formula:

$simged(P_1, P_2) = 1 - \{snv, sev, sbv\}$

Where:

$snv = \frac{\|sn\|}{\|N_1\|+\|N_2\|}\ sev = \frac{\|se\|}{\|A_1\|+\|A_2\|}\ sbv = \frac{2.\sum_{(n,m)\in M} 1-sim(n,m)}{\|N_1\|+\|N_2\|-\|sn\|}$

M represents the matched pairs based on the matching procedure.

N_i represents the activities in P_i.

A_i represents the edges (flows) in P_i.

sn is the set of all inserted and deleted nodes i.e. it evaluates the unmatched activities between P_1 and P_2.

se is the set of all inserted or deleted edges i.e. it evaluates the unmatched flows between P_1 and P_2.

Dijkman recommended using a weighted average for the three components snv, sev, sbv instead of using a plain average [39]. In our evaluation we used the weights which have been suggested by Friedrich [30]: $w_{sbv} = 0.4, w_{snv} = 0.3, w_{sev} = 0.3$.

The results of our evaluation show that our approach can generate models with more than 81% similarity comparing to manually created diagrams. The detailed results of the 10 groups of processes are shown in Table 1.

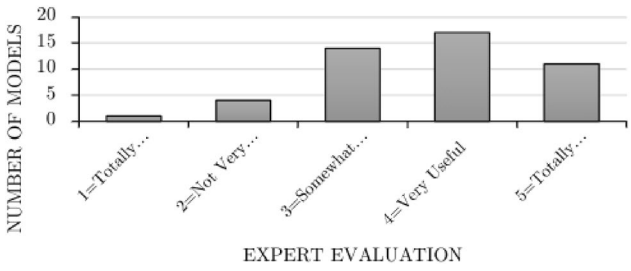

Fig. 12 The distribution of analyst evaluation

Experiment 2: Expert Evaluation

The main objective of this experiment is to see whether the generated models are useful and understandable from an expert point of view. The generated models were evaluated by an independent analyst in the range from 1 to 5, where 1 denotes a totally useless diagram and 5 denotes a totally useful diagram. In particular, value 3 has been used to denote the diagrams that are sufficiently useful to be used as a first draft for the desired model i.e. the expert sees that it is useful to correct the generated diagram instead of building it from scratch. Table 2 provides more details about these five levels.

The result (Fig. 12) shows that the analyst found that 42 out of 47 are useful (with usefulness level 3,4, or 5), while the remaining 10 diagrams are considered not very useful or totally useless. The results of this experiment indicated that our approach could generate a useful diagram in about 89% of tested models. Besides, in about 59% of the cases, analysts considered the output very useful, which means that the generated model gives the correct paths and extract almost all tasks. On the other hand, the analyst decided that the generated models are not good enough to be used in about 11% of the cases.

Table 2 The 5 Usefulness Levels

Rating	Label	Description
1	Totally useless	There is a lot of errors, it is totally useless
2	Not very useful	Some parts are correct. However, there are a lot of error, I prefer to build it from scratch
3	Somewhat useful	There are some major errors, but it is sufficiently useful to be used as a first draft for the desired model
4	Very useful	Most of the diagram is correct. It gives the correct paths in the process and extracts almost all tasks correctly. However, it needs some small changes and improvements
5	Totally useful	It is correct, I just need to add some tiny improvements

Discussion

Our experiments show that the generated BPMN diagrams are 81% similar to manually created ones. Error analysis shows clearly that most error cases do not affect the main flows in the process. In most cases, the process is still "semantically correct" (i.e.the main flows are recognized correctly) which might be the reason why the analyst (in the second experiment) found the generated model useful in 89% of the cases. In these cases, the analyst decided that it is better to correct the generated diagrams comparing to building them from scratch. This indicates that there is a considerable saving of time and effort comparing to building the model manually from the text.

We think that separating the semantic level of processing from the detailed morpho-syntactic levels plays the most important role in improving the "semantically correctness" and avoiding major errors, we "borrow" this idea from semantic transfer-based machine translation approaches where approaches focus on having a semantic representation (TextGraph in our approach) which is transferred to the target language, and only after that the detailed morphological and syntactic processing is done [22, 39].

The effect of using a semantic transfer-based MT approach could seem clear when comparing our detailed results (Table 1) with the detailed results of a syntax-oriented approach like Friedrich et al. approach [30]. Our approach achieves significantly better similarity when the process model text is larger in terms of the number of sentences (or the number of tasks in the targeted model), and complex in term of the number of conditions and cases (or the number of gateways and sequence flow in the targeted model). For instance, our approach achieves significantly better results for groups 2 and 4 which have the largest number of sentences and gateways. This result could be expected since as much as the text is large, the paths would be scattered over the text, and the probability of overlapping would be larger, and the role of semantic-level processing (i.e. the whole text or the whole BPMN diagram, see Fig. 3) would be much important.

TextGraph provides a solution for three main challenges:

1. The spreading of errors over paths: When there is an error in one of the activities, in most cases this error affects previous or next activity without affecting other flows in the process.
2. Recognize the flow when paths are scattered within the text: It is common to introduce the possible paths at the beginning then providing the details of each path in the next sections in the text like in the following example:

   "*The manager of the department can then reject or accept the order. If the order is accepted .....a paragraph explaining this path..... On the other hand, when the manager rejects the order ...a paragraph explaining this path.....*"

   The effect of detecting the main paths over the whole text is expected to increase as much as the number of sentences between these two paths (acceptance and rejection paths in the last example) increases.
3. The ability to handle more complex flows when there is more than one level of gateways and when there are overlapped paths, for example: *"the supervisor can* **accept or reject** *the report. These rules could be automated, to reduce the workload on the supervisor.*textbfIf *the supervisor rejects the report, the employee, who submitted it, is given a chance to edit it, for example, to correct errors or better describe an expense.***If** *the supervisor approves the report, it goes to the treasurer. The treasurer checks that all the receipts have been submitted and match the items on the list.***If** *all is in order, the treasurer accepts the expenses for processing (including, e.g., payment or refund, and accounting).* **If** *receipts are missing or do not match the report, he sends it back to the employee. If a report returns to the employee for corrections, it must again go to a supervisor, even if the supervisor previously approved the report.***If** *the treasurer accepts the expenses for processing, the report moves to an automatic activity that links to a payment system."*

On the other hand, the usage of the concept map provides a kind of taxonomy for the process to avoid:

1. Unreal actors since we accept the only actor which exists in the concept map.
2. Expressing the same actor in different format such as "Supply Chain Management" and "SCM", "informatics department" and "department", "the expense report" and "the report"...
3. Predicting the actors in a better way when it is mentioned indirectly using anaphora such as "He makes the commercial audit and issues the approval for payment."

Conclusion

In this paper, we presented a semantic transfer machine translation-based approach to generate models starting from the text. The approach consists of two phases: Natural Language Analysis and Modeling Language Generation. In Natural Language Analysis phase, we analyze the text, extract the required knowledge, and generate a Concept Map which summarizes the concepts of the related domain and the relationships between these concepts. In

Modeling Language Generation Phase, we focus primarily on transferring the semantic of the text into a diagram (called "TextGraph") that represents the main paths in the process. Later, we apply a set of syntactic, and morphological tuning steps to convert this TextGraph into the final BPMN model.The approach was implemented and evaluated using a similarity metric based on the graph edit distance.

Our evaluation has shown encouraging results. On average, we were able to generate models that are more than 81% similar to manually created models.

Funding This study has received no funding.

Data availability The used data is publically available as mentioned in evaluation section (Friedrich dataset, Reference Number for this dataset is [30]).

Declarations

Conflict of interest Authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Becker J, Rosemann M, Von Uthmann C. Guidelines of business process modeling. In: Business process management. Springer; 2000. p. 30–49.
2. Herbst J, Karagiannis D. An inductive approach to the acquisition and adaptation of workflow models. In: Proceedings of the IJCAI, vol. 99. Citeseer; 1999. p. 52–57.
3. Blumberg R, Atre S. The problem with unstructured data. Dm Rev. 2003;13(42–49):62.
4. Laporti V, Borges MR, Braganholo V. Athena: a collaborative approach to requirements elicitation. Comput Ind. 2009;60(6):367–80.
5. Riefer M, Ternis SF, Thaler T. Mining process models from natural language text: a state-of-the-art analysis, Multikonferenz Wirtschaftsinformatik (MKWI-16), March 2016. p. 9–11.
6. Friedrich F, Mendling J, Puhlmann F. Process model generation from natural language text. In: International conference on advanced information systems engineering. Springer; 2011. p. 482–496.
7. Mendling J, Leopold H, Pittke F. 25 challenges of semantic process modeling. Int J Inf Syst Softw Eng Big Companies (IJISEBC). 2015;1(1):78–94.
8. Bajwa IS. A natural language processing approach to generate sbvr and ocl. Ph.D. thesis, University of Birmingham. 2014.
9. De Marneffe M-C, Manning CD. The stanford typed dependencies representation. In: Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation. 2008. p. 1–8.
10. Achour CB. Guiding scenario authoring. In: EJC. 1998. p. 152–171.
11. de AR Gonçalves JC, Santoro FM, Baião FA. Let me tell you a story-on how to build process models. J Univers Comput Sci. 2011;17(2):276–95.
12. Santoro FM, Borges MR, Pino JA. Tell us your process: a group storytelling approach to cooperative process modeling. In: 2008 12th international conference on computer supported cooperative work in design. IEEE; 2008. p. 29–34.
13. Ghose A, Koliadis G, Chueng A. Process discovery from model and text artefacts. In: IEEE Congress on Services (Services 2007), vol. 2007. IEEE; 2007. p. 167–74.
14. Loper E, Bird S. Nltk: The natural language toolkit. In: Proceedings of the ACL-02 workshop on effective tools and methodologies for teaching natural language processing and computational linguistics. 2002. p. 63–70.
15. Sinha A, Paradkar A. Use cases to process specifications in business process modeling notation. In: 2010 IEEE international conference on web services. IEEE; 2010. p. 473–480.
16. Kennedy C, Boguraev B. Anaphora for everyone: pronominal anaphora resolution without a parser. In: COLING 1996 volume 1: the 16th international conference on computational linguistics. 1996.
17. Epure EV, Martín-Rodilla P, Hug C, Deneckère R, Salinesi C. Automatic process model discovery from textual methodologies. In: 2015 IEEE 9th international conference on research challenges in information science (RCIS). IEEE; 2015. p. 19–30.
18. Honkisz K, Kluza K, Wiśniewski P. A concept for generating business process models from natural language description. In: International conference on knowledge science, engineering and management. Springer; 2018. p. 91–103.
19. van der Aa H, Di Ciccio C, Leopold H, Reijers HA. Extracting declarative process models from natural language. In: International conference on advanced information systems engineering. Springer; 2019. p. 365–382.
20. van der Aa H, Balder KJ, Maggi FM, Nolte A. Say it in your own words: defining declarative process models using speech recognition. In: International conference on business process management. Springer; 2020. p. 51–67.
21. Van der Aa H. Comparing and aligning process representations. In: BPM (Dissertation/Demos/Industry). Springer; 2018. p. 16–20.
22. Leopold H, van der Aa H, Pittke F, Raffel M, Mendling J, Reijers HA. Searching textual and model-based process descriptions based on a unified data format. Softw Syst Model. 2019;18(2):1179–94.
23. Leopold H, van der Aa H, Reijers HA. Identifying candidate tasks for robotic process automation in textual process descriptions. In: Enterprise, business-process and information systems modeling. Springer; 2018. p. 67–81.
24. Feng W, Zhuo HH, Kambhampati, S. Extracting action sequences from texts based on deep reinforcement learning. In: Proceedings of the 27th international joint conference on artificial intelligence. 2018. p. 4064–4070.
25. Qian C, Wen L, Kumar A, Lin L, Lin L, Zong Z, Li S, Wang J. An approach for process model extraction by multi-grained text classification. In: International conference on advanced information systems engineering. Springer; 2020. p. 268–282.
26. Charoenpornsawat P, Sornlertlamvanich V, Charoenporn T. Improving translation quality of rule-based machine translation. In: COLING-02: machine translation in Asia. 2002.
27. Koehn P. Neural machine translation. Cambridge: Cambridge University Press; 2020.
28. Dorr B J, Hovy EH, Levin LS. Machine translation: interlingual methods. 2004.
29. Hutchins J. Machine translation: general overview. In: The Oxford handbook of computational linguistics. 2003.

30. Friedrich F. Automated generation of business process models from natural language input. M. Sc., School of Business and Economics. Humboldt-Universität zu Berli; 2010.
31. Lomas M, Cross E, Darvill J, Garrett R, Kopack M, Whitebread K. A robotic world model framework designed to facilitate human-robot communication. In: Proceedings of the SIGDIAL 2011 conference. 2011; p. 301–306.
32. Novak J, Gowin D. Learning how to learn. Cambridge: Cambridge University Press; 1984.
33. Chen D, Manning CD. A fast and accurate dependency parser using neural networks. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014. p. 740–750.
34. Novak JD, Cañas AJ. The theory underlying concept maps and how to construct them. Florida Inst Hum Mach Cogn. 2006;1(1):1–31.
35. Wu Z, Palmer M. V verbs semantics and lexical selection. In: Proceedings of the 32nd annual meeting on association for computational linguistics. 1994. p. 133–138.
36. Bassil Y, Alwani M. Context-sensitive spelling correction using google web 1t 5-gram information. Comput Inf Sci. 2012;5(3).
37. Brants T. Web 1t 5-gram version 1. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13 (2006).
38. Dijkman R, Dumas M, Van Dongen B, Käärik R, Mendling J. Similarity of business process models: metrics and evaluation. Inf Syst. 2011;36(2):498–516.
39. Jones B, Andreas J, Bauer D, Hermann KM, Knight K. Semantics-based machine translation with hyperedge replacement grammars. In: Proceedings of COLING 2012. 2012. p. 1359–1376.