FB2BPMN: An End-to-End Pipeline for Translating Unstructured User Feedback into BPMN Using LLM and Fuzzy Matching

Uce Indahyanti[1], Arif Djunaidy[2*], Daniel Siahaan[3]
[1,3]Department of Informatics, Faculty of Intelligent Electrical and Informatics Technology, Institut Teknologi Sepuluh Nopember (ITS), Surabaya 60111, Indonesia
[2]Department of Information Systems, Faculty of Intelligent Electrical and Informatics Technology, Institut Teknologi Sepuluh Nopember (ITS), Surabaya 60111, Indonesia
E-mail: 7025211023@student.its.ac.id, arif.djunaidy@its.ac.id, daniel@if.its.ac.id
[*]Corresponding author

Keywords: BPMN, fuzzy string matching, NLP, LLM translation, user feedback text

Received: September 24, 2025

Translating unstructured user feedback into Business Process Model and Notation (BPMN) is challenging due to informal language, contextual ambiguity, and the lack of explicit structural cues. We present FB2BPMN, an end-to-end pipeline that combines natural language processing (NLP), large language models (LLMs), and fuzzy string matching to automatically generate BPMN elements from raw feedback. The pipeline comprises four stages: sentence structuring, fact extraction, role-activity mapping, and fuzzy-based semantic alignment. We evaluate FB2BPMN on 125 annotated feedback instances sampled from academic journal management systems. Using expert-authored BPMN as reference, FB2BPMN attains precision 0.97, recall 0.88, and F1 0.91 on element identification and accuracy 0.85 on process flow construction, outperforming a rule-based baseline. Results indicate strong structural and semantic correspondence, showing that FB2BPMN effectively bridges informal feedback and formal process representations.

Povzetek: Študija uvaja metodo FB2BPMN, ki samodejno generira BPMN modele iz ne-strukturiranih besedilnih povratnih informacij uporabnikov. Rezultati evalvacije na 125 primerih potrjujejo visoko natančnost in zanesljivost pristopa, kljub manjšim težavam pri poimenovanju elementov zaradi leksikalnih razli.

1 Introduction

Business Process Model and Notation (BPMN) is a widely adopted standard for graphically representing business processes in a domain-independent format understandable to diverse stakeholders [1], [2]. It bridges process design and implementation [3], enabling a shared understanding of system functionalities and operational flows [4]. Organizations typically maintain large collections of documents, ranging from formal operating procedures to unstructured textual feedback, that may contain process-relevant knowledge. While structured documentation facilitates BPMN generation in many real-world settings, formal sources are unavailable, and process knowledge exists primarily in unstructured form. BPMN models can be derived from various source documents, including structured sources such as business rules, standard operating procedures, and use case diagrams, as well as semi-structured or unstructured sources such as event logs, user stories, and user feedback [5]. Whereas structured sources enable straightforward BPMN generation, real-world user feedback is informal and ambiguous, making direct transformation more challenging. Generating BPMN from unstructured documents is particularly challenging due to inconsistent grammar, incomplete descriptions, and lack of formalized process elements, which are common issues when formal documentation is unavailable.

Prior research has explored SBVR-based approaches have been used to translate the interview transcripts into formal process models. BPMN generation from both structured and semi-structured sources. For example, SBVR-based approaches have been used to translate interview transcripts into formal process models [6], while user story-driven methods have been employed to construct BPMN diagrams through predefined semantic templates [7], [8]. These methods demonstrate high accuracy when processing well-formed input but struggle to handle the variability of informal, real-world feedback. User feedback can serve as a valuable asset for business process improvement. It can support incremental changes and explorative redesign initiatives [9], [10]. Today, the improvement or redesign of business process models still relies on expert expertise, often disregarding user feedback. Consequently, users often propose process redesigns, but these suggestions are frequently disregarded [11]. On the other hand, many users access app repositories and social media platforms to provide feedback on their digital experiences. This vast textual

corpus has the potential to be mined to extract detailed information regarding user expectations and feature-oriented suggestions [12].

Recent advances in natural language processing (NLP) and large language model (LLM) have opened new opportunities for generating process models directly from textual descriptions, including those derived from informal or unstructured sources such as user feedback, an area that remains underexplored. Fatawi et al. [13] conducted a bibliometric review underscoring the pivotal role of prompt engineering and LLM in NLP applications and their increasing relevance in automation systems. Studies have shown that LLM can extract structured information from natural language, map it to domain-specific ontologies [14], and support model-to-model transformation through semantic interpretation [15].

Recent studies highlight the potential of LLM in semantic understanding. Prasad et al. [16] demonstrated their effectiveness in interpreting unstructured text, supporting our use of LLM to translate user feedback into BPMN representations. Sonbol et al. [17] proposed a translation-inspired method to transform structured process descriptions into BPMN models, and the BPMN Chatbot tool [18] has demonstrated the ability to produce BPMN diagrams from text. Most existing methods for BPMN generation rely on structured inputs, such as formal descriptions or domain-specific templates. In contrast, using large language models to generate BPMN models from unstructured user feedback remains underexplored.

Research on business processes generally revolves around generating, redesigning, extracting, and measuring the similarity of business processes. Fehrer et al. [19] conducted process redesign and similarity measurement studies. The study proposed a redesign method utilizing the aBPR approach. Delias et al. [20] explored business process prototype improvements through evidence-based approaches, while [21] proposed a roadmap for process model improvement. Another study introduced a method for measuring business process similarity using role relation networks and role hierarchy graphs [22]. Furthermore, [23] present an approach to refine business process models by examining syntax and semantics. Work related to business process generation can be categorized into two main aspects: the source document type and the format of the resulting model. The following studies concentrate on building process models from structured text source documents, transforming them into formats or notations. Several researchers have proposed techniques for deriving business processes from source documents such as business rules and use cases, which are converted into SBVR [24], [25].

Other studies investigated the transformation of business rules and event logs into BPMN [11], [26], [27]. Methods for generating model processes from business rules and using case documents into SBVR were discussed [24]. SBVR translation into UML using an NLP approach was explored [25]. Other approaches transform textual business rules and event log documents into BPMN [11], [27]. Zeng, Duan et al. [28] describe top-down process mining as a method for generating process models from activity logs. Similarly, [29] propose an approach for constructing process models based on a set of refinement rules. In a related study, [30] investigate how Decision Model and Notation (DMN) can accurately represent data modeled within BPMN. Recent research on BPMN generation increasingly incorporates advanced technologies to improve automation and accuracy. Several studies [17], [31], [32] utilized textual requirements or textual description document sources. Sholiq et al. [33] demonstrated BPMN generation from Indonesian functional requirements using tailored syntactic patterns and parsing rules. These works combined machine learning techniques, including deep learning, with specific tools to improve the extraction and modeling processes from textual descriptions into BPMN diagrams. Textual business process documents can generate textual BPMN using a deep learning approach known as A-BPS [34].

Other approaches include a machine translation method to construct BPMN in a text-graph format [17] and the use of NLP and deep learning techniques, leveraging GPT language models and Graphviz tools [31]. Another method combines NLP techniques, SBVR definitions, rule sets, and spreadsheet-based descriptions to generate BPMN diagrams from textual requirements [32]. Danenas et al. [25] proposed an NLP-enhanced approach for extracting business vocabularies and rules from natural language, enabling formal representation aligned with SBVR standards. Tangkawarow et al. [6] proposed the ID2SBVR method to extract SBVR operational rules from informal document sources.

User feedback represents a valuable resource for improving business processes. Mustansir et al. [35] assessed user satisfaction by applying sentiment analysis to feedback related to process execution. In a subsequent study, Mustansir et al. [11] proposed classifying user feedback into three levels using deep learning and natural language processing techniques. Ahmed, Shahzad et al. [36] integrated user feedback into process models by analyzing relationships among feature vectors, data balancing strategies, and word embeddings in a different approach. A study in this domain presented a model-based approach for transforming user stories into process model notations [37]. Ahmed & Mustansir [38] introduced a novel NLP-based concept to map user feedback to relevant process model elements. An automated approach for generating business process models from user stories was proposed by [8]. Other research utilized user stories to construct business process models in UML format [7],[39]. Further studies focused on identifying and mapping user feedback into process elements [25], [31].

As discussed in the introduction, recent advances in NLP and LLM have opened opportunities for generating process models from text, particularly through tasks such as information extraction and semantic transformation [14],[15]. In the context of business process modeling, approaches such as the translation-inspired method by [17] and the BPMN Chatbot tool developed by [18] have demonstrated the feasibility of generating BPMN models from textual input. Conventional approaches to BPMN generation typically rely on structured inputs, limiting their adaptability to informal or ambiguous text. SBVR-

based methods require strict syntactic rules, making them sensitive to linguistic variability. User story techniques depend on predefined semantic templates, reducing flexibility in real-world scenarios. Deep learning models offer greater adaptability but often require extensive preprocessing and lack transparency. These constraints highlight the need for methods capable of handling unstructured inputs with minimal assumptions.

However, the automatic generation of BPMN from unstructured user feedback remains largely unexplored. Unlike user stories or formal requirements, real-world feedback often contains fragmented sentences, colloquial expressions, and multiple intertwined intents. Such characteristics make it difficult for template-based or rule-based approaches to produce accurate models. Existing LLM-based BPMN generation methods have primarily been tested on clean, syntactically consistent input, leaving a gap in applicability to noisy feedback data.

To our knowledge, no existing study has directly addressed the translation of unstructured user feedback into BPMN models. To fill this gap, we propose a method that converts informal feedback into BPMN representations without requiring prior structuring. The approach integrates natural language processing techniques, large language models, and fuzzy string matching to extract and map process elements from free-form text, enabling a more responsive and user-driven process modeling paradigm. This translation pipeline is designed to handle the variability and informality of user input, often containing fragmented expressions or symbols, and to produce structured textual representations suitable for BPMN model generation.

Several recent studies have explored BPMN generation from semi-structured text such as user stories, interviews, and business rules. These approaches typically rely on well-formed sentences and domain-specific vocabulary. For instance, SBVR-based methods depend on clearly defined fact types and logical connectors [25], while user story approaches require structured input and semantic templates [8]. In contrast, real-world user feedback is often unstructured, informal, and ambiguous, posing challenges for traditional models [8], [33]. These methods tend to perform poorly when faced with noisy input.

The proposed method addresses this gap by integrating large language models, fuzzy string matching, and syntactic patterning to extract BPMN elements from free-form text. Unlike prior approaches that depend on structured templates or curated rule sets, our method offers a flexible and scalable mechanism for modeling user-driven business processes in practical settings. This study addresses the above gap by introducing a method that integrates NLP techniques, LLM, and fuzzy string matching to translate unstructured user feedback into BPMN diagrams. The main contributions are as follows: (1) An end-to-end translation pipeline capable of processing diverse sentence structures without prior manual structuring, (2) A syntactic pattern–based preprocessing stage to improve interpretation of informal text before BPMN element extraction, (3) A comprehensive evaluation on 125 annotated feedback instances covering four sentence complexity levels (simple, compound, complex, compound-complex), and (4) The release of an annotated dataset to support reproducibility and future research in automated process modeling.

Table 1 summarizes representative studies on process model generation. Earlier works (e.g., [24]) relied on structured inputs, while later approaches [8],[33] addressed semi-structured requirements using rule-based or NLP methods. More recent efforts (e.g., [17]) employed LLMs but still depended on curated or domain-specific input formats. In contrast, the proposed FB2BPMN method uniquely handles unstructured feedback through an LLM-guided transformation with fuzzy string matching, thereby improving robustness and adaptability in processing noisy textual data. This advancement clearly fills the research gap left by previous approaches and establishes a foundation for scalable, data-driven process modeling.

The remainder of this paper is structured as follows: Section II discusses the methods, Section III presents the results and discussion, and Section IV provides the conclusion.

2 Feedback-to-BPMN method

This section describes the FB2BPMN (Feedback-to-BPMN) method developed in this study. Building on the identified research gap, the FB2BPMN framework is as an end-to-end pipeline that transforms unstructured user feedback into executable BPMN models. The approach integrates natural language processing (NLP), large language models (LLMs), and fuzzy string matching to ensure semantic alignment between informal text and formal process representations.

To guide the methodological design and evaluation, We formulated two research questions in line with the study objectives. These questions evaluate the effectiveness of the FB2BPMN pipeline in processing informal user feedback. In addition, they explore the role of fuzzy string matching in improving the consistency and semantic alignment of the generated BPMN models:

RQ1: How effectively can the FB2BPMN pipeline generate BPMN representations from informal and unstructured user feedback?

RQ2: To what extent does the integration of fuzzy string matching enhance the semantic consistency and naming alignment of BPMN elements compared to using an LLM alone?

To address these questions, the subsequent subsections describe the proposed FB2BPMN translation pipeline, which consists of preprocessing, pattern-based sentence structuring, BPMN element extraction, and fuzzy-based semantic alignment.

2.1. Architecture

FB2BPMN is designed as an end-to-end method designed to accommodate informal and syntactically diverse inputs. The method systematically generates

Table 1: Comparison of representative studies on process model generation.

Study	Dataset	Method / Approach	Performance	Limitation
Tangkawarow et al. (2022)	Informal documents	ID2SBVR: Rule-based NLP for SBVR vocabulary extraction	Qualitative validation only (syntactic and vocabulary correctness)	Limited to semi-structured inputs; requires formal grammar
Nasiri et al. (2023)	User stories	Template-driven NLP	Precision ≈ 0.94	Not applicable to free-form feedback; limited semantic flexibility
Köpke & Safan (2024)	Chat-based process descriptions	LLM-based dialogue transformation	Not evaluated quantitatively	Relies on structured prompts; conversational scope
Sholiq et al. (2022)	Textual requirements	Pattern-based / rule-based NLP	Avg. F1 ≈ 0.87	Limited adaptability to informal sentences
Arshad et al. (2019)	Controlled Natural Language (CNL) conforming to SBVR specifications	SBVR2XML: CNL→SBVR→XML transformation	Precision ≈ 0.949; Recall ≈ 0.925 (tool eval.)	Assumes controlled NL/SBVR; targets XML generation (not BPMN); does not address noisy, unstructured feedback
This study	User feedback (unstructured text)	FB2BPMN: NLP + LLM + fuzzy matching	Avg. F1 ≈ 0.91; Precision ≈ 0.92	Evaluation based on a representative 125-entry subset; future work includes broader domain and dataset expansion.

structured BPMN elements without requiring manual restructuring, thereby enabling organizations to leverage user feedback more effectively, as a foundation for business process improvement.

Figure 1 illustrates the role of the proposed FB2BPMN method within the broader cycle of business process improvement based on user feedback. FB2BPMN functions as a critical stage that translates informal textual input into structured process models, enabling systematic business process refinement. By positioning FB2BPMN in this context, the figure highlights its contribution as an integral component of the end-to-end improvement pipeline rather than an isolated technique. To the best of our knowledge, no prior study has directly addressed the improvement of business process models in such an integrated manner [40].

Building on this foundation, Figure 2 presents the internal workflow of FB2BPMN in greater detail. Specifically, the method operates through four sequential stages: (1) converting user feedback into structured text, (2) extracting relevant fact types, (3) mapping these fact types to process constructs, and (4) generating BPMN elements.

Each stage incrementally transforms unstructured input into a formal process representation, ensuring syntactic consistency and semantic interpretability. This stepwise procedure provides a clear methodological pathway for translating diverse feedback into BPMN models that can directly support business process redesign.

The method relies on LLM to analyze large volumes of previously unseen text and capture contextual nuances in user feedback [41]. By extracting semantic elements such as actions, roles, and objects, the LLM enhances the interpretation of unstructured input and addresses challenges of linguistic variability, implicit references and fragmented expressions. Rather than a standalone generator, the LLM functions as a semantic filter that enriches and disambiguates user inputs before subsequent processing.

Complementing this, NLP techniques and fuzzy string matching are employed to identify process elements through fact type extraction and to map semantically related terms to BPMN constructs [31], [42]. This combination effectively addresses both lexical variation and semantic similarity, thereby supporting the reliable generation of BPMN elements from highly diverse feedback data

2.2. Step 1: Conversion of user feedback into structured text

Algorithm 1 illustrates how a large language model (LLM), such as OpenAIGenerator, is employed with the PromptBuilder class from the Haystack library to transform raw text into sentences that follow specific syntactic patterns. Each sentence is first split into individual units (line 1) and checked for coordinating conjunctions and clauses (line 2), which determines its classification as compound (line 3), compound complex (lines 4–6), complex (line 8), or simple (line 10). These types are then mapped to corresponding prompt templates (line 12) that guide the model in generating structured representations (lines 13–14). Valid outputs are stored (lines 15–16), while invalid ones are logged (lines 17–18)

before the final set of structured sentences is returned (line 21).

```
Algorithm 1: Conversion of User Feedback into Structured Text
Data: T = user feedback text
Result: ST = structured text {Actor, Action, Object, Condition}
1.  for each sentence ∈ splitSentences(T) do
2.      if contains(sentence,["and","but","or"]) and containsClause(sentence) then
3.          type ← "compound"
4.      else if contains(sentence,["if","although","because","when","while"]) then
5.          if containsConjunction(sentence, ["and", "but", "or"]) then
6.              type ← "compound-complex"
7.          else
8.              type ← "complex"
9.          end
10.     else
11.         type ← "simple"
12.     end
13.     prompt ← TEMPLATE[type].Replace({query ← sentence})
14.     structuredSentence ← LLM.generate(prompt)
15.     if isValid(structuredSentence) then
16.         ST.add(structuredSentence)
17.     else
18.         logError("Invalid structured sentence for input: " + sentence)
19.     end
20. end
  // Each sentence tᵢ ∈ T is transformed into a structured sentence sᵢ using a
  // type-specific template.
  // ST = f_LLM(T) = { sᵢ ∈ TEMPLATE(tᵢ) | tᵢ ∈ T, type(tᵢ) ∈ {simple, compound,
  // complex, compound complex} }
21. return ST
```

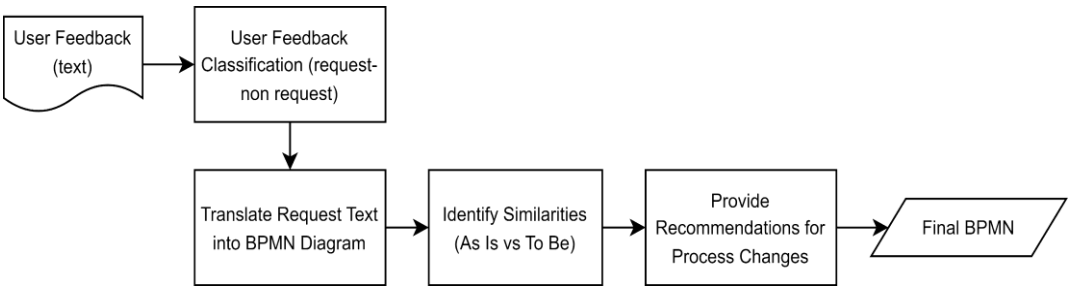

Figure 1: Overall architecture of the integrated process improvement approach.

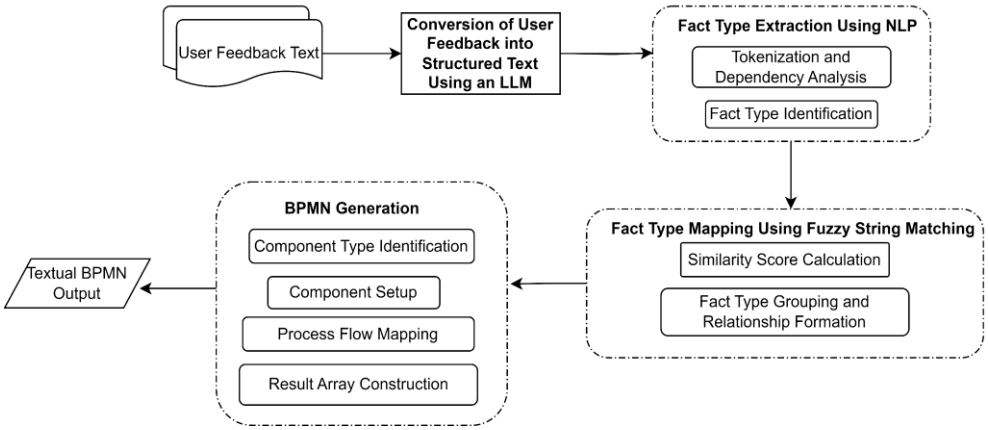

Figure 2: The FB2BPMN framework

Within the FB2BPMN method, this structured text is subsequently used for fact type extraction, mapping, and constructing BPMN elements. Figure 2 presents the architecture that supports this text to BPMN translation. Identifying such structures is essential for downstream processing, as compound and complex sentences often introduce greater syntactic ambiguity [43]. Sentence type classification reduces ambiguity and improves fact type extraction accuracy, particularly for compound and complex sentences.

Each sentence type is mapped to a specific prompt template that guides the LLM in generating a well-structured input version. For example, a simple sentence such as "*Allow tagging of articles*" might be processed

using a template: Identify the actor, action, and object in the following instruction: {sentence}, producing the structured form [System – allows – tagging of articles]. Similarly, a complex sentence such as "*If the user is logged in, allow tagging of articles*" would use a conditional template to produce separate condition and activity structures. These structured outputs improve downstream fact extraction accuracy by reducing syntactic ambiguity.

To improve clarity, the templates omit unnecessary introductory phrases. The algorithm classifies each sentence, selects the appropriate template, and submits it to the LLM. If the output meets structural criteria, it is retained; otherwise, it is flagged for review. For example, the model produces coherent and meaningful sentences reflecting the original intent from an unstructured input containing fragmented expressions and informal phrasing. While minor stylistic inconsistencies may occur, the transformation effectively enhances the readability and supports further processing in the BPMN generation pipeline.

The LLM was guided using a structured prompting template designed to transform informal feedback into syntactically consistent sentences. The prompt specified three output patterns: activity, conditional, and parallel sentences, each following predefined templates (e.g., "<subject> <verb> <object>" or "If <subject> <condition>, <activity>"). The model was instructed to rewrite user feedback from a system-centric perspective, ensuring the use of formal English, consistent grammatical subjects, and clear syntactic relations. This guided transformation produced normalized text that could be systematically parsed for fact type extraction in subsequent steps.

To illustrate, consider the input sentence "*Authors cannot track the review progress easily after submission*." The corresponding structured output generated by the LLM is {Actor: *authors*, Action: *track*, Object: *review progress*, Condition: *after submission*}.

This step classifies sentences according to their grammatical type and applies a corresponding type-specific prompt template to guide the LLM transformation. The resulting output is a normalized structured representation that preserves the original meaning while improving readability and syntactic consistency for subsequent BPMN element extraction in Algorithm 2.

The complete LLM prompt template used in this step is provided in the supplementary material available on Figshare at https://doi.org/10.6084/m9.figshare.29023172.

2.3. Step 2: fact type extraction

This step analyzes the structured text and categorizes sentences into two primary fact types: activities and conditions. These categories form the basis for mapping textual information to BPMN components, such as tasks and gateways. The process comprises two subprocesses: (1) tokenization and dependency analysis, and (2) fact type extraction.

The first subprocess, tokenization and dependency analysis, involves breaking each sentence into tokens (words) and labeling their grammatical roles, such as subject, predicate, and object. This analysis uses the spaCy Python library, which provides a detailed grammatical structure for each sentence. For example, the sentence "*The system processes user data*" is parsed into tokens: "The," "system," "processes," "user," and "data", with dependencies identified as "system: Subject," "processes: Predicate," and "data: Object." This step ensures that the system can accurately capture syntactic relationships within the text.

Sentences that lack a complete subject predicate object structure are flagged for manual review or passed through a secondary LLM based reconstruction. For instance, the incomplete feedback "*Need option for article tagging*" omits an explicit subject; the secondary pass infers "*System*" as the default actor before proceeding to extraction. This dual pass mechanism minimizes data loss and maintains the integrity of the fact type dataset.

This procedure is summarized in Algorithm 2, which integrates tokenization, dependency analysis, and fact type extraction. The algorithm processes each structured sentence to extract the subject, action, and object, forming the grammatical basis for classification as either activity or condition fact types.

To illustrate, consider the structured input {Actor: *authors*, Action: *track*, Object: *review progress*, Condition: after submission}. From this representation, the algorithm extracts two fact types: activity (*authors, track, review progress*) and condition (*after submission*).

This step converts structured sentences into semantic triples that capture either actions or conditions, providing the input for Algorithm 3.

2.4. Step 3: fact type mapping

This step applies fuzzy string matching, using the Levenshtein distance metric, to align text segments that are similar but not identical. The process consists of two subprocesses: (1) identifying similarities between fact types, and (2) generating sequence flows based on the relationships inferred from the similarity scores, as illustrated in Algorithm 3.

Fuzzy string matching is applied to align semantically similar fact types that differ in wording. This study uses the Levenshtein distance metric, with a similarity threshold of 0.85. This value was selected empirically to balance sensitivity (capturing legitimate variations) and specificity (avoiding incorrect merges). Although alternative measures such as Jaro–Winkler and cosine similarity were considered, Levenshtein distance was chosen for its robustness to short text variations and token-level edits, common in user feedback.

The Levenshtein distance measures the minimum number of operations required to transform one string into another, namely deletions, insertions, or substitutions. It is computed by initializing a matrix to represent the prefix lengths of both strings and filling it with the minimal number of operations needed at each position. The

```
Algorithm 2: Fact Type Extraction
Data: ST = structured text produced by Algorithm 1
Result: FT = Extracted fact types
for each sentence ∈ structuredText do
  //Tokenization and dependency analysis
  tokens ← tokenize(sentence)
  dependencyLabels ← parseDependency(tokens)
 // Fact type extraction
if containsKeyword(tokens, "if")then
  condition ← extractCondition(sentence)
  factType ← createFact("condition", condition)
 else
 //Identify core elements
   subject ← findTokenByLabel(dependencyLabels, "nsubj")
   verb ← findTokenByLabel(dependencyLabels, "ROOT")
  object ← findTokenByLabel(dependencyLabels, "dobj")
 if subject ≠ null and verb ≠ null and object ≠ null then
   activity ← createActivity(subject, verb, object)
   factType ← createFact("activity", activity)
 else
   logWarning("Incomplete sentence structure: " + sentence)
   continue
  end
 end
   factTypes.append(factType)
end
  // Each structured sentence sᵢ ∈ ST is mapped to a fact type fᵢ
  // FT = f_extract(ST) = { fᵢ ∈ MAPPED(sᵢ) | sᵢ ∈ ST }
return factTypes
```

resulting distance is converted into a similarity score (simScore) using the formula presented in Eq. (1)

$$simScore(x_i, x_j) = \left(1 - \frac{distance(x_i, x_j)}{\max\text{len}(x_i, x_j)}\right) \times 100 \qquad (1)$$

As shown in Eq. (1), the distance (x_i, x_j) measures dissimilarity or difference between x_i and x_j, and maxlen (x_i, x_j) is the length of the longer sequence between x_i and x_j. For example, given the sequences x_i = “user” and x_j = ”use”, we can calculate that the distance (x_i, x_j) is the number of edits required to align the sequences. In this case, deleting 'r' from "user" results in distance (x_i, x_j) = 1, while maxlen (x_i, x_j) represents the length of the longer sequence, in this case "user," yielding maxlen = 4. Therefore, the similarity score between x_i and xj is computed based on these values.

Similarity scores are used to identify relationships among fact types. Fact types exhibiting high similarity are consolidated into a single BPMN element, whereas those with low similarity are treated as distinct entities. The BPMN model encapsulates clusters of similar fact types and represents their interrelations through sequence flows. Contextually related but distinct elements are connected accordingly, while highly similar ones preserve their original source and target associations. This step employs a fuzzy string matching algorithm to analyze and map semantic relationships across fact types.

To illustrate, consider the following input fact types derived from Algorithm 2: activity (authors, track, review progress) and condition (after, submission). The temporal cue after submission establishes a dependency relation between the two facts, indicating that the activity occurs subsequent to the condition. Consequently, the algorithm produces a relation in which the completion of submission precedes the activity of tracking the review progress.

This step identifies semantic and temporal relationships among extracted fact types, which serve as the foundation for BPMN element generation in Algorithm 4.

2.5. 4: BPMN generation

In this step, two subprocesses, namely generating BPMN elements and mapping relationships, construct BPMN models from fact types and their semantic links. Fact types are first transformed into BPMN tasks or gateways, depending on whether they represent activities or conditions. The relationships between these components are then mapped using sequence or conditional flows, ensuring logical coherence in the resulting model. A textual representation of the BPMN

The algorithm in Step 4 consists of two subprocesses: generating BPMN elements and mapping relationships. Activity fact types are transformed into tasks, and condition fact types into gateways, while relationships between them are mapped using sequence or conditional flows. These subprocesses result in a textual BPMN representation supporting visualization and implementation, as illustrated in Algorithm 4. The resulting BPMN XML adheres to the BPMN 2.0 specification, enabling direct import into standard modeling tools such as bpmn.io or Camunda Modeler.

To illustrate, consider the following example derived from the previous step. The input fact types are (*authors, track, review progress*) and (*after submission*), with the relation (*submission - track review progress*). From these inputs, the algorithm generates BPMN elements in both XML and JSON formats, as shown below.

Generated BPMN elements (XML):

<bpmn:task id="T1" name="Submit Manuscript"/>
<bpmn:exclusiveGateway id="G1" name="After Submission"/>

```
Algorithm 3: Fact Type Mapping
Data: FT = extracted fact types
Result: R = relationships between fact
  //Identify similarities between fact types
1.for each factType1 ∈ factTypes do
2. for each factType2 ∈ factTypes do
3.  if factType1 ≠ factType2 then
4.   score ← fuzzy_match(factType1.name,factType2.name)
5.    if score ≥ threshold then
6.     relation ← createRelation(factType1,factType2)
7.    relations.append(relation)
8. end
9.end
10.for each relation ∈ relations do
  // Generate sequence flows from relationships
11. source ← relation.source
12. target ← relation.target
13. sequenceFlow ← createSequenceFlow source.id, target.id)
14.bpmnSequenceFlows.append(sequenceFlow)
15.end
  // Formal representation:
  // R = f_map(FT) = { (f_i, f_j) | sim(f_i, f_j) ≥ θ, f_i ≠ f_j }
  // Where sim is a fuzzy similarity function and θ is the threshold
16.return bpmnSequenceFlows
```

```
Algorithm 4: BPMN Generation
input: FT = Extracted fact type, R = Relationship between facts types
Result: BPMN textual, BPMN XML
  //Genareating BPMN elements
1.Initialize BPMN_textual ← []
2.Create BPMN_XML root element (e.g.,<bpmn:definitions>)
3.Add <bpmn:process> element to BPMN_XML
4.for each factType ∈ FT do
5. if factType.type == "activity" then
6.  task ← createTask(factType.subject, factType.action, factType.object)
7.  BPMN_textual.add(task)
8.  Add <bpmn:task> to BPMN_XML with task attributes
9. else if factType.type == "condition" then
10.  gateway ← createGateway factType.condition)
11.  BPMN_textual.add(gateway)
12.  Add <bpmn:exclusiveGateway> to BPMN_XML with gateway attributes
13.  end
14.end
   //Mapping relationships
15.for each relation ∈ R do
16.  if relation.type == "dependency" then
17.    flow ← createSequenceFlow (relation.source, relation.target)
18.   BPMN_textual.add(flow)
19.   Add <bpmn:sequenceFlow> to BPMN_XML connecting source and target
20.  else if relation.type == "condition" then
21.   flow ← createConditionalFlow (relation.source, relation.target)
22.   BPMN_textual.add(flow)
23.   Add <bpmn:sequenceFlow> with condition to BPMN_XML
24.  end
25. end
    // BPMN = f_generate(FT, R) = {elements ∪ flows|elements from FT, flows from R}
26. return BPMN
```

<bpmn:task id="T2" name="Track Review Progress"/>
<bpmn:sequenceFlow sourceRef="T1" targetRef="T2"/>

Generated BPMN elements (JSON):

{"bpmn_shapes": [{"id": 0, "type": "task", "name": "Submit Manuscript"}, {"id": 1, "type": "gateway", "name": "After Submission"}, {"id": 2, "type": "task", "name": "Track Review Progress"}], "sequence_flows": [{"source": 0, "target": 1}, {"source": 1, "target": 2}]}

This step converts semantic fact types and their identified relationships into BPMN-compliant elements in both textual and XML representations, completing the FB2BPMN transformation pipeline.

3 Results and discussion

This section presents the results of translating user feedback into BPMN models (BPMN1) using the proposed method and compares them with expert-authored reference models (BPMN2). The evaluation covers dataset composition, experimental procedure, and performance analysis based on standard metrics, followed by an interpretation of the findings and their implications.

3.1. Experiments

A total of 1,027 user feedback entries were initially collected from the Public Knowledge Project's Open Journal Systems (PKP OJS) platform. These entries comprised diverse forms of user-generated text, including feature suggestions, bug reports, and general comments. Following expert annotation, entries that did not contain explicit requests for new features or improvements to

existing functionalities were excluded, resulting in a refined dataset of 656 relevant feedback instances.

The dataset exhibits substantial structural variability, ranging from 30 to 50 words and from 2–5 sentences per entry. Typical noise patterns include @-mentions, URLs, issue or pull request references, markup tags (e.g., <h2>), and interface-specific tokens such as menu labels or configuration paths. These heterogeneous characteristics motivated the design of the sentence structuring process (Algorithm 1) and subsequent fact extraction and alignment steps designed to handle noisy and irregular input.

From this filtered dataset, 125 entries were selected using stratified random sampling to ensure proportional representation of major sentence types. The sample size was determined by practical constraints, as manual evaluation required significant domain expertise and effort. Nevertheless, the selected subset provides sufficient linguistic diversity to support a reliable assessment of the proposed FB2BPMN method.

All 125 entries were successfully processed, demonstrating the system's capability to interpret unstructured and variable user input. The dataset includes four sentence types: simple, compound, complex, and compound–complex, providing a representative foundation for evaluating the method's effectiveness in translating natural language into structured BPMN models. Prior studies have shown that sentence complexity can significantly affect performance in downstream transformation tasks [43].

The final sample consists of 28 complexes, 33 compound, 14 compound-complex, and 50 simple sentences, corresponding to 22.4%, 26.4%, 11.2%, and 40% of the dataset. This distribution ensures that each sentence type is adequately represented, allowing for a nuanced evaluation of the method's robustness across varying levels of syntactic complexity. The supplementary dataset, including all 125 annotated instances and evaluation outputs, is available in the Figshare repository (DOI: 10.6084/m9.figshare.29023172).

Each feedback instance was translated into structured text and then converted into BPMN models (BPMN1), which were evaluated against expert-generated reference models (BPMN2). The comparison was conducted based on three main evaluation criteria: (1) element identification, which assesses the completeness of BPMN2 elements relative to BPMN1; (2) element naming, which evaluates terminological consistency; and (3) process flow, which examines the alignment of activity sequences. Performance was measured using standard metrics, including accuracy, precision, recall, and F1 score, to assess the model's effectiveness in producing BPMN representations that accurately reflect user feedback.

Each feedback case was processed individually, and in certain instances, multiple translation iterations were applied to improve contextual fidelity. For example, the feedback "currently dependent files can only be added/administered on the galleys grid. Allow them elsewhere" as refined through several iterations to capture the intended process semantics more accurately. The expanded evaluation using 125 datasets enabled a broader and more representative analysis of system performance across varying levels of feedback complexity.

Table 2: The distribution of sentence types.

Sentence Type	Dataset ID	Entries
Complex	1, 9, 21, 43, 58, 76, 81, 85, 95, 97, 98, 99, 101, 102, 104, 105, 106, 107, 108, 110, 111, 112, 114, 115, 116, 117, 118, 124	28
Compound	10, 13, 15, 24, 32, 35, 36, 38, 39, 40, 41, 42, 44, 46, 47, 49, 51, 55, 57, 61, 63, 68, 71, 72, 75, 77, 83, 84, 86, 89, 91, 92, 93	33
Compound-Complex	45, 62, 80, 90, 94, 96, 100, 103, 109, 119, 120, 121, 123, 125	14
Simple	2, 3, 4, 5, 6, 7, 8, 11, 12, 14, 16, 17, 18, 19, 20, 22, 23, 25, 26, 27, 28, 29, 30, 31, 33, 34, 37, 48, 50, 52, 53, 54, 56, 59, 60, 64, 65, 66, 67, 69, 70, 73, 74, 78, 79, 82, 87, 88, 113, 122	50

Table 2 summarizes the distribution of sentence types within the dataset, providing a representative foundation for evaluating the performance of the proposed method in the subsequent section.

3.2. Evaluation

The evaluation aims to measure the translation model's performance using standard metrics, including accuracy, precision, recall, and F1 score [8], [44], [45]. It compares the reference textual BPMN with the generated textual BPMN from the translation process using the same dataset.

The dataset was derived from a platform that developed an Open Journal System (OJS), containing user feedback related to business process improvement requests. Reference textual BPMN models (BPMN2) were manually created by experts, specifically online journal administrators at a university. The generated models (BPMN1) were produced by the proposed translation method. Structured feedback text was provided to the experts, along with a BPMN writing template aligned with the array-based data structure of the generated BPMN. This template ensured a uniform format for comparison. The evaluation focused on three aspects: element identification, element naming, and process flow:

(1) Element Identification (tasks, gateways): Does BPMN2 include all the elements that appear in BPMN1?
(2) Element Naming: Are there discrepancies in the naming of elements? This assessment considers exact matches and semantic similarity, measured against a predefined threshold. For example, "if it meets the standards" and "if it meets the basic standards" would

be considered equivalent if their meanings are aligned, despite minor differences in phrasing.

(3) Process Flow: Does the sequence of activities in BPMN2 follow the same logical order as in BPMN1?

The following definitions are applied to calculate the evaluation metrics, namely accuracy (Acc), precision (Prec), recall (Rec), and F1 score (F1). True Positive (TP) refers to elements that correctly appear and match in BPMN2 as expected based on BPMN1, including both the name and the sequence. False Positive (FP) refers to additional or mismatched elements in BPMN2 that do not exist in BPMN1 or are incorrect. False Negative (FN) refers to elements that should be present according to BPMN1 but are missing or incorrect in BPMN2. For example, in one feedback case, the generated BPMN1 contained the task "Allow tagging of articles" and the gateway "If the user is logged in", both of which also appeared in BPMN2. These were counted as true positives. However, BPMN2 included an additional task "Display confirmation message" not present in BPMN1 (false negative), while BPMN1 included "Check user role" which was absent in BPMN2 (false positive).

Table 4 presents the average performance of the FB2BPMN system across four sentence types: simple, compound, complex, and compound complex, evaluated using accuracy, precision, recall, and F1 score. The assessment covers three fundamental BPMN modeling tasks: element identification (A), element naming (B), and process flow construction (C).

Sentences with greater syntactic complexity, particularly complex and compound complex types, consistently yield higher scores in element identification and process flow tasks. This trend is likely driven by richer syntactic cues and more explicit relational structures. In contrast, simple sentences have the lowest scores, especially in the naming task, due to limited contextual and lexical information.

Table 3: Summary statistics of evaluation metrics

Aspect	Accuracy	Precision	Recall	F1-score
Mean:				
Identification	0.87	0.97	0.88	0.91
Naming	0.77	0.91	0.77	0.83
Process Flow	0.85	0.96	0.86	0.90
Std. Deviation:				
Identification	0.15	0.08	0.15	0.11
Naming	0.17	0.15	0.16	0.14
Process Flow	0.19	0.10	0.19	0.13

Among the three tasks, element naming (B) demonstrates the greatest variability across metrics. Precision decreases for complex and compound complex sentences, indicating challenges in assigning consistent labels when multiple clauses or entities are present. Meanwhile, process flow performance (C) improves with sentence complexity, as reflected in higher F1 scores, suggesting that complex syntactic structures support more accurate reconstruction of procedural logic.

These results have direct practical implications. As presented in Table 3, high precision in element identification (97%) suggests the method is reliable when accuracy is critical, such as in safety-critical domains or compliance auditing. However, lower recall in element naming (77%) indicates that the method may omit some valid elements or produce inconsistent labels, which could require post-processing or human validation in applications demanding high completeness. Overall, the results indicate that increased sentence complexity enhances the system's ability to extract structured representations, leading to more accurate and coherent BPMN model generation.

The results presented in Table 3 indicate that the model performs well in satisfying syntactic expectations across key evaluation dimensions. Metrics were obtained through a fully manual evaluation, in which the BPMN elements generated by the model were compared against expert-labeled ground truth. The assessment focused on semantic adequacy, label appropriateness, and contextual relevance, ensuring that the evaluation captured structural accuracy, semantic fidelity, and domain-specific validity. Preliminary findings suggest that the model's high precision in element identification and flow construction aligns with human judgment. However, variations remain in the naming aspect, where semantic interpretation often diverges due to paraphrasing or lexical ambiguity in user inputs. These results underscore the need for further qualitative assessment to better understand the trade-off between syntactic accuracy and semantic fidelity in real-world scenarios.

While Table 3 summarizes the overall performance across identification, naming, and process flow aspects, further analysis was conducted to examine how sentence structure influences these outcomes. The goal of this complementary evaluation was to assess whether syntactic complexity affects the model's ability to accurately extract BPMN elements from unstructured feedback.

As shown in Table 4, the FB2BPMN pipeline maintained consistent performance across all sentence types, with only a slight decrease observed for compound-complex sentences due to increased syntactic ambiguity.

Table 4: Performance by sentence type

Sentence Type	Precision	Recall	F1-Score
Simple	0.95	0.90	0.92
Compound	0.94	0.89	0.91
Complex	0.93	0.88	0.90
Compound-Complex	0.91	0.86	0.88
Average	0.93	0.88	0.90

To illustrate specific challenges, Dataset 14 achieved perfect element identification (1.00), yet naming recall declined to 0.60 due to lexical variation. For example, BPMN1 generated "*Export all button will be applied*" and "*Select all option would be preferable on a single grid page*," whereas BPMN2 produced "*Export all button will be added*" and "*Select all option is preferred for single grid pages.*" Although semantically equivalent, these differences reduced the naming score despite minimal

impact on process logic. Dataset 35 demonstrated structural compression, where BPMN1 specified four distinct activities ("*The system provides a content editor for HTML versions," "The system saves HTML versions to the database or as flat files," "Some journals request this functionality*," *and "The system acknowledges the difficulties that this functionality presents*"). At the same time, BPMN2 abstracted them into two generalized tasks ("*Provide and save HTML content editor output" and "Handle journal requests and acknowledge editor difficulties*"). This abstraction reduced recall for both identification and naming (approximately 0.50–0.60) and lowered flow accuracy to 0.33, preserving overall logic but limiting granularity and traceability.

In contrast, Datasets 64 and 65 achieved full alignment across all metrics, confirming robustness when the input was unambiguous. For instance, in Dataset 64, BPMN1 produced tasks such as "Authors upload revised figures separately" and "Allow partial update of figures only," with a gateway "Should this happen without changing full submission?"; BPMN2 generated semantically equivalent elements "Let authors upload revised figures separately," "Simplify figure-only updates in workflows," and a gateway "Does this avoid full resubmission?" Likewise, in Dataset 65, BPMN1 specified "Import metadata from external databases" and "Reduce errors in manual metadata entry," with a gateway "Can this reduce manual entry errors?"; BPMN2 produced corresponding elements "Enable metadata import from external sources," "Reduce manual input errors," and a gateway "Does this minimize entry mistakes?" All elements were consistently identified, named, and sequenced, yielding complete alignment between model outputs and expert references. These examples highlight two recurring sources of error: omission or merging of elements, which primarily affects recall and flow reconstruction, and semantic compression, where multiple activities are combined into broader elements, thereby reducing naming precision and structural detail. While the generated BPMN models preserve high-level process logic, further improvements are required to support use cases demanding high semantic fidelity and fine-grained process validation. Complete results for all 125 cases are provided in the Figshare repository.

An aggregated evaluation complemented the qualitative analysis and examined overall consistency. Table 5 presents a comparative summary of BPMN evaluation results using standard classification metrics (Precision, Recall, F1, Accuracy). The rule-based baseline method [32] was analyzed separately, as it employed MMRE-based error evaluation rather than element-wise classification metrics. The reported values represent the mean and standard deviation of accuracy, precision, recall, and F1-score across all datasets, confirming the model's robustness under varying syntactic and semantic complexity levels. The complete dataset-level evaluation table, including detailed metric values for all 125 instances, is provided in the Figshare repository for transparency and reproducibility. A comparative analysis with existing BPMN generation approaches is presented to contextualize these findings. Although the overall accuracy achieved by the proposed method (87%) is slightly lower than that reported by [8] (94%), it demonstrates higher precision, reaching 97% in element identification. It reflects a conservative extraction strategy that minimizes false positives, making the approach suitable for scenarios that require high-confidence mapping from natural language feedback. Unlike Nasiri's method [8], which assumes structured and well-formed input, FB2BPMN prioritizes robustness under noisy and unstructured conditions. This design choice emphasizes high precision while trading off some recall, ensuring reliability in real-world scenarios where feedback is highly variable.

To assess computational efficiency, we evaluated the average runtime of each component using Google Colab Pro with an NVIDIA T4 GPU and 32 GB RAM. The sentence restructuring phase completed within a few seconds per input on average, while fact extraction and mapping were executed even faster. BPMN diagram generation was near-instantaneous due to its template-driven logic. These observations indicate that the method provides consistent and responsive performance, ensuring practical scalability for processing user feedback in real-world settings.

To further assess the effectiveness of the proposed method, we provide a qualitative comparison with existing BPMN generation approaches that rely on structured inputs. For example, [8] proposed a user story-based approach that utilizes predefined semantic patterns. Danenas et al. [25] extracted SBVR-based elements from UML use case diagrams using natural language processing techniques. These methods typically require domain-specific templates and are limited in handling informal or ambiguous user expressions.

In contrast, our method demonstrates greater robustness in processing unstructured user feedback, which frequently contains irregular syntax, vague references, and multiple intents. While structured input methods often fail to generate valid BPMN components without extensive preprocessing, our approach achieves up to 95 percent accuracy in element identification, even when applied to free-form text.

Although a direct quantitative comparison was not feasible due to differing input assumptions and data representations, the proposed method demonstrates clear advantages in handling informal and unconstrained input sources, broadening the practical applicability of automated business process modeling. The findings suggest that the method is particularly suited for domains where feedback is abundant but formal documentation is scarce, such as public service systems, e-commerce

Table 5: Comparative summary of BPMN evaluation methods.

Aspect	This Study	Nasiri et al., [8]
Evaluation Method	Precision, recall, F1 score, and accuracy	Precision, recall, and accuracy based on true/false positive and negative counts
Evaluation Focus	Element detection, naming consistency, and process flow correctness	Activity correctness, control flow validation, and transition accuracy
Key Results	Precision = 97%, Recall = 88%, Accuracy = 87%	Precision = 94%, Recall = 97%, Accuracy = 94%
Strengths	Reliable element identification under informal and variable input structures	High semantic accuracy in modeling control flow and behavioral transitions

platforms, and citizen reporting tools. In such contexts, generating high-precision process models from informal text can accelerate process redesign, facilitate bottleneck identification, and reduce reliance on domain experts for initial modeling.

The proposed method shows strong potential for application in real-world settings, particularly within public service systems, customer support platforms, and business process improvement teams. By enabling the transformation of informal user feedback into structured BPMN representations, the method allows business analysts to identify recurring issues, recommend service redesigns, and automate routine modeling activities. For example, in a government complaint handling system, free-text reports from citizens could be converted into process models to reveal bottlenecks or procedural inefficiencies. Recent studies further support the practical viability of this approach. Licardo et al. [31] demonstrated that BPMN models can be accurately extracted from informal textual descriptions using a combination of large language models and natural language processing techniques, achieving up to 96 percent accuracy. Furthermore, a systematic review by [46] highlights the increasing effectiveness of machine learning and deep learning methods, particularly LLM-based approaches, compared to traditional rule-based techniques when processing unstructured and fragmented business process descriptions. These findings affirm the applicability of our method in handling real-world input scenarios where formal documentation is often lacking. This ability to extract structured process knowledge from unstructured narratives supports evidence-based decision making and promotes continuous improvement in dynamic organizational environments.

As presented in Table 5, the proposed method achieves an overall accuracy of 87%, significantly lower than the 94% reported by [8]. However, it demonstrates higher precision, particularly in element identification (97%, see Table 3). This trade-off indicates that the method identifies fewer total elements but with greater consistency, reducing false positives. Such precision-oriented behavior is especially valuable in domains where model clutter or misinterpretation must be minimized. In contrast, Nasiri's method [8] yields higher accuracy but at the cost of lower precision, increasing the risk of redundant or inconsistent elements.

These findings also highlight several limitations that inform future research directions. First, the variability in naming accuracy indicates the need for semantic enhancement, such as contextual embeddings or domain-specific ontologies, to improve consistency. Second, while the method demonstrates high precision, the trade-off with recall suggests that hybrid modeling strategies could better balance granularity and completeness. Finally, since the evaluation was conducted on a single dataset, cross-domain validation is required to assess the generalizability and adaptability of the method across different organizational contexts.

These results answer RQ1, indicating that large language models can effectively process informal and unstructured user feedback to generate accurate BPMN representations. The FB2BPMN pipeline maintained consistent precision and recall across all four sentence types: simple, compound, complex, and compound-complex, demonstrating robustness in handling syntactic variability within unstructured text.

Regarding RQ2, the integration of fuzzy string matching notably improved naming consistency and semantic alignment in BPMN generation compared to rule-based baselines. This enhancement reduced synonym-related naming variations and increased overall correspondence between user-expressed actions and BPMN activity labels, confirming the benefit of combining fuzzy matching with LLM-based extraction.

For reference, the rule-based method by [32] provides a suitable baseline for BPMN generation from textual requirements. Their rule and pattern-based framework achieved an average F1 score of about 0.87 on structured requirement documents. In comparison, the proposed FB2BPMN method reached an average F1-score of 0.92 on unstructured user feedback, highlighting the benefit of combining large language models with fuzzy matching to handle informal, domain-specific text.

Beyond quantitative gains, FB2BPMN shows strong potential for real-world deployment across domains such as customer service, healthcare, and industrial workflows, where unstructured feedback can be transformed into actionable process models.

The proposed FB2BPMN framework can also be applied to healthcare systems, where user feedback often highlights operational or service-related issues. For instance, the feedback "*It's difficult to schedule an appointment with the same doctor again after a consultation. Please allow users to rebook easily and add a reminder for follow-up visits.*" is transformed into structured sentences: "*The system allows users to rebook appointments easily*." and "*If a consultation is completed, the system provides a reminder for follow-up visits*." From these, FB2BPMN extracts the fact types (system, allow, rebook appointment) and (consultation, completed),

forming a temporal dependency from consultation to rebook appointment. The resulting BPMN represents a patient follow-up workflow, showing that FB2BPMN effectively captures procedural logic from healthcare narratives with minimal domain adaptation.

A full expert-based evaluation has also been completed, with results openly available in the accompanying Figshare repository.

Despite these promising results, certain threats to validity remain. Although this study used 125 OJS feedback instances, the method demonstrated robust performance across diverse sentence types. Future work will extend the evaluation to larger or multilingual datasets. Conceptually, the approach exhibits robustness in processing noisy and uncertain language, analogous to adaptive fuzzy control mechanisms that stabilize nonlinear systems under uncertainty [47]. These results indicate both the practical value and theoretical soundness of the FB2BPMN framework, laying the groundwork for broader domain and cross-lingual applications.

4 Conclusions

This study presents a method that integrates Natural Language Processing (NLP), Large Language Models (LLM), and fuzzy string matching to automatically translate unstructured user feedback into Business Process Model and Notation (BPMN) diagrams. Addressing a critical gap in business process modeling, the method systematically transforms informal textual input into structured BPMN representations through a four-stage methodology: (1) text structuring, (2) fact extraction, (3) relationship mapping, and (4) BPMN generation. This design supports processing linguistically diverse inputs without requiring prior manual structuring.

Evaluation on 125 annotated feedback instances demonstrated that the proposed method performs robustly across varying levels of feedback complexity. It achieves mean accuracies of 87% for element identification, 77% for naming, and 85% for process flow mapping, with precision scores of 97%, 91%, and 96%, respectively. This high level of precision is particularly valuable in real-world applications where output reliability is critical.

Despite these strengths, challenges remain in achieving consistent element naming and handling semantically diverse expressions, particularly in feedback containing multiple intertwined intents or high lexical variability. Moderate standard deviations observed in naming and flow metrics suggest opportunities for refinement, especially in handling lexical variation and feedback containing multiple intents. Further investigation is needed to improve semantic alignment and address abstraction and compression issues in multi-topic feedback. Additionally, sentence structure was found to influence model performance. Complex and compound complex sentences yielded the highest accuracy and F1 scores across all evaluation aspects, suggesting that syntactic richness enhances interpretability and translation quality. These findings underscore the importance of linguistic features in optimizing user-to-model transformation workflows.

The method establishes a reliable and scalable foundation for translating informal user feedback into BPMN models. It contributes to the development of intelligent, user-responsive business process modeling systems. Theoretically, this study extends BPMN generation research by bridging informal human feedback with formal process modeling through LLM-based structuring. Future work will focus on semantic enhancement, hybrid modeling, and cross-domain validation to improve consistency, balance precision and recall, and ensure generalizability.

Acknowledgement

This work was supported by Institut Teknologi Sepuluh Nopember for the research activities and by Universitas Muhammadiyah Sidoarjo for the publication funding.

References

[1] K. Grolinger, M. A. M. Capretz, A. Cunha, and S. Tazi, “Integration of business process modeling and Web services: A survey,” *Serv. Oriented Comput. Appl.*, vol. 8, no. 2, pp. 105–128, 2014, https://doi.org/10.1007/s11761-013-0138-2.

[2] T. Skersys, P. Danenas, and R. Butleris, “Extracting SBVR business vocabularies and business rules from UML use case diagrams,” 2018, https://doi.org/10.1016/j.jss.2018.03.061.

[3] V. Kale, “Business Process Modeling and Notation,” *Enterp. Process Manag. Syst.*, no. January, pp. 257–275, 2019, https://doi.org/10.1201/9780429453311-14.

[4] M. Javed and Y. Lin, “iMER: Iterative process of entity relationship and business process model extraction from the requirements,” *Inf. Softw. Technol.*, vol. 135, no. January, p. 106558, 2021, https://doi.org/10.1016/j.infsof.2021.106558.

[5] U. Indahyanti, A. Djunaidy, and D. Siahaan, “Auto-Generating Business Process Model From Heterogeneous Documents : A Comprehensive Literature Survey,” in *Proc. 9th Int. Conf. Electrical Engineering, Computer Science and Informatics*, 2022, pp. 239–243. https://doi.org/https://doi.org/10.23919/EECSI56542.2022.9946460.

[6] I. Tangkawarow, R. Sarno, and D. Siahaan, “ID2SBVR: A Method for Extracting Business Vocabulary and Rules from an Informal Document,” *Big Data Cogn. Comput.*, vol. 6, no. 4, 2022, https://doi.org/10.3390/bdcc6040119.

[7] F. Gilson, M. Galster, and F. Georis, "Generating use case scenarios from user stories," in *Proceedings - 2020 IEEE/ACM International Conference on Software and System Processes, ICSSP 2020*, 2020, pp. 31–40. https://doi.org/10.1145/3379177.3388895.

[8] S. Nasiri, A. Adadi, and M. Lahmer, "Automatic generation of business process models from user stories," *Int. J. Electr. Comput. Eng.*, vol. 13, no. 1, pp. 809–822, 2023, https://doi.org/10.11591/ijece.v13i1.pp809-822.

[9] I. Beerepoot *et al.*, "The biggest business process management problems to solve before we die," *Comput. Ind.*, vol. 146, no. December 2022, p. 103837, 2023, https://doi.org/10.1016/j.compind.2022.103837.

[10] T. Grisold *et al.*, "The Five Diamond Method for Explorative Business Process Management," *Bus. Inf. Syst. Eng.*, vol. 64, no. 2, pp. 149–166, 2022, https://doi.org/10.1007/s12599-021-00703-1.

[11] A. Mustansir, K. Shahzad, and M. K. Malik, "Towards automatic business process redesign: an NLP based approach to extract redesign suggestions," *Autom. Softw. Eng.*, vol. 29, no. 1, 2022, https://doi.org/10.1007/s10515-021-00316-8.

[12] J. Wouters, A. Menkveld, S. Brinkkemper, and F. Dalpiaz, "Crowd-based requirements elicitation via pull feedback: method and case studies," *Requir. Eng.*, vol. 27, no. 4, pp. 429–455, 2022, https://doi.org/10.1007/s00766-022-00384-6.

[13] I. Fatawi, M. Asy'ari, H. Hunaepi, T. Samsuri, and M. R. Bilad, "Empowering Language Models Through Advanced Prompt Engineering: A Comprehensive Bibliometric Review," *Indones. J. Sci. Technol.*, vol. 9, no. 2, pp. 441–462, 2024, https://doi.org/10.17509/ijost.v9i2.71481.

[14] A. Mukanova *et al.*, "LLM-Powered Natural Language Text Processing for Ontology Enrichment," *Appl. Sci.*, vol. 14, no. 13, 2024, https://doi.org/10.3390/app14135860.

[15] P. Danenas and T. Skersys, "Exploring Natural Language Processing in Model-To-Model Transformations," *IEEE Access*, vol. 10, no. November, pp. 116942–116958, 2022, https://doi.org/10.1109/ACCESS.2022.3219455.

[16] S. Prasad, H. Gupta, and A. Ghosh, "Leveraging the Potential of Large Language Models," *Inform.*, vol. 48, no. 8, pp. 1–16, 2024, https://doi.org/10.31449/inf.v48i8.5635.

[17] R. Sonbol, G. Rebdawi, and N. Ghneim, "A Machine Translation Like Approach to Generate Business Process Model from Textual Description," *SN Comput. Sci.*, vol. 4, no. 3, pp. 1–16, 2023, https://doi.org/10.1007/s42979-023-01742-z.

[18] J. Köpke and A. Safan, "Introducing the BPMN-Chatbot for Efficient LLM-Based Process Modeling," 2024, https://ceur-ws.org/Vol-3758/paper-15.pdf.

[19] T. Fehrer, D. A. Fischer, S. J. J. Leemans, M. Röglinger, and M. T. Wynn, "An assisted approach to business process redesign," *Decis. Support Syst.*, vol. 156, no. July 2021, 2022, https://doi.org/10.1016/j.dss.2022.113749.

[20] P. Delias and G. T. Nguyen, "Prototyping a business process improvement plan. An evidence-based approach," *Inf. Syst.*, vol. 101, p. 101812, 2021, https://doi.org/10.1016/j.is.2021.101812.

[21] F. Johannsen and H. G. Fill, "Meta Modeling for Business Process Improvement," 2017, doi: 10.1007/s12599-017-0477-1.

[22] Q. Zeng, J. Liu, C. Zhou, C. Liu, and H. Duan, "A Novel Approach for Business Process Similarity Measure Based on Role Relation Network Mining," 2020, https://doi.org/10.1109/ACCESS.2020.2983114.

[23] S. Ayari, Y. B. Hlaoui, and L. Ben Ayed, "A specific language for developing business process by refinement based on BPMN 2.0," *Proc. 16th Int. Conf. Softw. Technol. ICSOFT 2021*, no. July, pp. 489–496, 2021, https://doi.org/10.5220/0010561404890496.

[24] S. Arshad, I. S. Bajwa, and R. Kazmi, "Generating SBVR-XML Representation of a Controlled Natural Language," in *Communications in Computer and Information Science*, 2019, vol. 932, pp. 379–390, https://doi.org/10.1007/978-981-13-6052-7_33.

[25] P. Danenas, T. Skersys, and R. Butleris, "Natural language processing-enhanced extraction of SBVR business vocabularies and business rules from UML use case diagrams," *Data Knowl. Eng.*, vol. 128, no. June 2019, p. 101822, 2020, https://doi.org/10.1016/j.datak.2020.101822.

[26] B. Estrada-Torres, M. Camargo, M. Dumas, L. García-Bañuelos, I. Mahdy, and M. Yerokhin, "Discovering business process simulation models in the presence of multitasking and availability constraints," *Data Knowl. Eng.*, vol. 134, no. December 2020, p. 101897, 2021, https://doi.org/10.1016/j.datak.2021.101897.

[27] T. Skersys, P. Danenas, R. Butleris, A. Ostreika, and J. Ceponis, "Extracting sbvr business vocabularies from uml use case models using m2m transformations based on drag-and-drop actions," *Appl. Sci.*, vol. 11, no. 14, 2021, https://doi.org/10.3390/app11146464.

[28] Q. Zeng, H. Duan, and C. Liu, "Top-Down Process Mining from Multi-Source Running Logs Based on Refinement of Petri Nets," *IEEE Access*, vol. 8, pp. 61355–61369, 2020, https://doi.org/10.1109/ACCESS.2020.2984057.

[29] Q. Mo, F. Dai, D. Liu, J. Qin, Z. Xie, and T. Li, "Development of private processes: A refinement approach," *IEEE Access*, vol. 7, pp. 31517–31534, 2019, https://doi.org/10.1109/ACCESS.2018.2889715.

[30] E. Bazhenova, F. Zerbato, B. Oliboni, and M. Weske, "From BPMN process models to DMN decision models," *Inf. Syst.*, vol. 83, pp. 69–88, 2019, https://doi.org/10.1016/j.is.2019.02.001.

[31] J. T. Licardo, N. Tankovic, and D. Etinger, "A Method for Extracting BPMN Models from Textual Descriptions Using Natural Language Processing," in *Procedia Computer Science*, 2024, vol. 239, no. January, pp. 483–490, https://doi.org/ 10.1016/j.procs.2024.06.196.

[32] S. Sholiq, R. Sarno, and E. S. Astuti, "Generating BPMN diagram from textual requirements," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 10, pp. 10079–10093, 2022, https://doi.org/ 10.1016/j.jksuci.2022.10.007.

[33] S. Sholiq, M. A. Yaqin, A. P. Subriadi, and B. Setiawan, "Generation of business process modeling notation diagrams from textual functional requirements in Indonesian," *Int. J. Electr. Comput. Eng.*, vol. 15, no. 3, pp. 2938–2950, 2025, https://doi.org/10.11591/ijece.v15i3.pp2938-2950.

[34] X. Han *et al.*, "A-BPS: Automatic Business Process Discovery Service using Ordered Neurons LSTM," *Proc. - 2020 IEEE 13th Int. Conf. Web Serv. ICWS 2020*, vol. 1, pp. 428–432, 2020, https://doi.org/10.1109/ICWS49710.2020.00063.

[35] A. Mustansir, K. Shahzad, and M. K. Malik, "Sentiment Analysis of User Feedback on Business Processes," in *Proceedings - 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops, ASEW 2021*, 2021, pp. 204–210, https://doi.org/https://doi.org/10.1109/ASEW52652.2021.00048.

[36] S. Ahmed and K. Shahzad, "Augmenting Business Process Model Elements With End-User Feedback," *IEEE Access*, vol. 10, no. October, pp. 115635–115651, 2022, https://doi.org/10.1109/ACCESS.2022.3216418.

[37] K. Baïna, M. El Hamlaoui, and H. Kabbaj, "Business Process Modelling Augmented: Model Driven transformation of User Stories to Processes," *ACM Int. Conf. Proceeding Ser.*, pp. 215–220, 2020, https://doi.org/10.1145/3419604.3419793.

[38] S. Ahmed and A. Mustansir, "Mapping Textual Feedback to Process Model Elements," *Proc. - 2020 35th IEEE/ACM Int. Conf. Autom. Softw. Eng. Work. ASEW 2020*, pp. 131–137, 2020, https://doi.org/10.1145/3417113.3423376.

[39] S. Nasiri, Y. Rhazali, M. Lahmer, and N. Chenfour, "Towards a Generation of Class Diagram from User Stories in Agile Methods," in *Procedia Computer Science*, 2020, vol. 170, pp. 831–837, https://doi.org/10.1016/j.procs.2020.03.148.

[40] D. C. Wynn and A. M. Maier, "Feedback systems in the design and development process," *Res. Eng. Des.*, vol. 33, no. 3, pp. 273–306, 2022, https://doi.org/10.1007/s00163-022-00386-z.

[41] L. Yan *et al.*, "Practical and ethical challenges of large language models in education: A systematic scoping review," *Br. J. Educ. Technol.*, vol. 55, no. 1, pp. 90–112, 2024, https://doi.org/10.1111/bjet.13370.

[42] H. Waheed, S. U. Hassan, N. R. Aljohani, J. Hardman, S. Alelyani, and R. Nawaz, "Predicting academic performance of students from VLE big data using deep learning models," *Comput. Human Behav.*, vol. 104, 2020, https://doi.org/10.1016/j.chb.2019.106189.

[43] A. Praveen Kumar, A. Nayak, K. Manjula Shenoy, R. J. Manoj, and A. Priyadarshi, "Pattern-Based Syntactic Simplification of Compound and Complex Sentences," *IEEE Access*, vol. 10, pp. 53290–53306, 2022, https://doi.org/10.1109/ACCESS.2022.3174846.

[44] A. Haj, A. Jarrar, Y. Balouki, and T. Gadir, "The Semantic of Business Vocabulary and Business Rules: An Automatic Generation from Textual Statements," *IEEE Access*, vol. 9, pp. 56506–56522, 2021, https://doi.org/10.1109/ACCESS.2021.3071623.

[45] M. S. M. Rudwan and J. V. Fonou-Dombeu, "Hybridizing Fuzzy String Matching and Machine Learning for Improved Ontology Alignment," *Futur. Internet*, vol. 15, no. 7, 2023, https://doi.org/10.3390/fi15070229.

[46] S. Van Woensel, W.; Motie, "NLP4PBM: A systematic review on process extraction using natural language processing with rule based, machine and deep learning methods," *Enterp. Inf. Syst.*, 2024, https://doi.org/10.1080/17517575.2024.2417404

[47] X. Qin, S. Li, and H. Liu, "Adaptive fuzzy synchronization of uncertain fractional-order chaotic systems with different structures and time-delays," *Adv. Differ. Equations*, vol. 2019, no. 1, pp. 1–16, 2019, https://doi.org/10.1186/s13662-019-2117-1.