International Journal of Basic and Applied Sciences, 14 (8) (2025) 212-222

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/2fxxam53
Research paper

Extracting and Minimizing Relations for Enhanced Coverage of User Stories

Amol Sharma [1,2] *, Anil Kumar Tripathi [1]

[1] Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi 221005, Uttar Pradesh, INDIA
[2] Department of Computer Science and Engineering, Meerut Institute of Technology, Meerut 250103, Uttar Pradesh, INDIA
**Corresponding author E-mail: amolsharma.rs.cse21@itbhu.ac.in*

Received: October 27, 2025, Accepted: December 7, 2025, Published: December 12, 2025

Abstract

User stories are the first choice of practitioners for expressing requirements in agile projects. Semi-structured natural language (NL) user stories, though easy to read and write, cannot accurately represent a problem domain holistically. The conceptual model, constituting relations (e.g., Teacher teaches Students) among key concepts in a domain, comes in handy to serve the purpose. Several NLP-based approaches exist that extract these relations automatically from user stories for conceptual modeling. These approaches do not make optimum use of NLP capabilities, and consequently, inefficiency and incompleteness in the extracted models are observable. To deal with these issues, we propose an approach for relation extraction using the Open Information Extraction (OpenIE) NLP technique. The OpenIE facilitates our process by automatically extracting relation triples (subject, relation, object). The primary extraction results in a multitude of relations that we reduce into a minimal set by applying a two-step reduction process. The minimal set of relations helps us achieve efficiency as we plate up necessary minimum relations for further processing in software development. The expanded coverage of the input user story by the minimal relation set indicates a high degree of completeness, as our minimal set of relations represents most of the domain knowledge hemmed in by the user story. We devise two metrics, Relation Set Reduction (RSR) and Relation Set Coverage (RSC), to evaluate our approach. The evaluation of two extensive user story datasets shows promising results as we are able to achieve a 65.91% reduction (RSR) in relations, with 86.45% coverage (RSC) of the user stories.

Keywords: *Conceptual Model; Relation Extraction; Relation Minimization; NLP; OpenIE.*

1. Introduction

Requirements are commonly expressed as user stories in agile development projects [3]. All the subsequent phases of software development rely on user stories as central artifacts [7], [8]. Agile practitioners generally write user stories using Connextra notation [4]: “As a <role>, I want to <feature> so that <benefit>." As per this notation, user stories represent who (role) needs what (feature) and why (benefit) [5, 6]. In the proposed approach, we call for the user stories to be supplied in the same format. Since conceptual/domain models are derived from the requirements, understanding how user stories evolve from natural language specification to structured domain model becomes essential. User stories are refined into low-level specifications in conceptual/ domain models; this is the central idea behind model-driven engineering [9]. Understanding the problem domain is a critical factor in the success of a software development project. The conceptual or domain model construction is a key step in understanding the problem domain and transitioning from natural language (NL) requirements to meticulous specifications worth analysing [1]. A conceptual model consists of semantic relations between structured concepts in the problem domain [10], [32]. Semantic relations form the backbone of domain understanding; the next concern lies in extracting these relations from user stories.

The most valuable semantic information about the problem domain is hidden in the relations between different elements of the graphical models [34]. Relation extraction from the NL documents is a major Information Extraction (IE) task and contributes to the formal model generation [34]. These relationship models provide a holistic view of the requirements and support identifying dependencies, redundancies, and inconsistencies [11], [33]. Manual extraction of relations from a large pool of user stories is tedious and takes significant effort and time. Manual construction and maintenance of conceptual models outweigh the benefits accrued out of it [11]. Thus, the automated extraction of relations will greatly assist domain engineers in constructing conceptual models. It will increase the efficiency compared with creating a domain model from scratch, out of large requirements [2].

Several approaches, tools, and techniques have been proposed in the literature to extract relations from user stories as part of conceptual model extraction [11-20]. Due to user stories being expressed in NL, NLP has been the primary technique applied in most of the proposed extraction processes [21]. However, relation extraction for conceptual modeling still needs to use NLP capabilities optimally [32]. The

lack of efficiency is regarded as one of the difficulties in extraction. [22]. The extraction techniques also suffer from the issue of incompleteness, leading to expensive slip-ups later [18]. Therefore, an efficient approach is still sought after [23]. To address these limitations, an automated method that extracts and refines relations more effectively is the need of the hour.

In light of the abovementioned issues, we propose an automated approach for extracting a set of relations from semi-structured NL user stories and minimizing this set further. We aim to improve the characteristics in terms of enhanced coverage with the reduced number of relations. Thus, we set forth the objective for this work as maximal coverage of domain knowledge (enclosed in the user story) through a minimal set of relations. The minimal relation set is essential to achieve efficiency, as one can infer domain knowledge from such a set by engaging a minimum amount of time and effort. Maximizing coverage is essential to deal with the issue of incompleteness, as high coverage tends to make the resulting conceptual model more complete. To operationalize this objective, Open Information Extraction (OpenIE) may act as the foundational extraction mechanism.

Unlike existing approaches, our approach is based on Open Information Extraction (OpenIE). We leverage sophisticated Stanford Open NLP's 'openie' annotator [25] to extract binary relations (Teacher, teaches, student) from a user story. We further process the extracted relation triples (subject, relation, object) to arrive at a minimal set of relations. Two-step reductions are performed to reduce the original set of relations to the minimal set of relations. To demonstrate the applicability of the idea and the corresponding method, we evaluate the approach on two datasets belonging to two distinct problem domains and purposefully show that our approach attains a 65.91% reduction in relations, with 86.45% coverage of the domain knowledge enclosed in the user stories.

The remainder of the paper is organized as follows. In section 2, we report on the research work related to ours. Section 3 elaborates on the proposed relation extraction and reduction process. In section 4, we present the evaluation of the proposed approach and discuss the results obtained. Section 5 concludes the paper and gives future directions.

2. Related Work

User stories are a well-known notation for expressing software requirements in agile development projects [5], [6], [8]. Out of different templates for writing user stories, the Connextra format is a widely adopted notation among agile practitioners [6]. Connextra template focuses on three important aspects of a requirement [30]: who needs it? What is needed? Why is it needed? These three aspects correspond to role, feature, and benefit, respectively. The benefit part is deemed optional.

As the user stories are expressed in NL, they are easy to read, write, and understand but are not capable of providing a holistic view of the system, focusing on relations among key entities [16, 33]. One has to extract these relations, forming a conceptual model, so as to provide the intended holistic view and support discovering dependencies, redundancies, and inconsistencies in requirements [11, 33].

User stories being expressed in NL, NLP has been the core technology applied by most of the approaches to relation extraction proposed in the literature. In [35], Berry et al. identify model generation from NL requirements as one of the types; the approaches of all NLP-based RE tools are categorized. A number of NLP-based tools and techniques [11-20] have been proposed with diversified approaches and different target models. Tools such as Visual Narrator and ArTu target related but distinct automation goals. Visual Narrator [11] extracts entities and relations using rule-based NLP and generates conceptual models, but it does not perform relation-set reduction or aim to optimize coverage across large user-story corpora. ArTu [16] instead produces goal models: it identifies roles, actions, and intentions from user stories and constructs goal models rather than binary semantic relations. In contrast, our approach focuses on deriving high-coverage OpenIE-based relation triples and minimizing them, producing a compact relation set suitable as input to downstream conceptual modeling.

After a careful analysis of the proposed approaches, we identified certain gaps that require immediate attention:

- These approaches still to make the best possible use of sophisticated NLP techniques readily available at our disposal [32]. We leverage one such technique named Open information extraction (OpenIE) in our work to extract relation triples [25] (Subject, relation, object) from user stories. OpenIE has been put to use for serving several other purposes like inferring access rules [36], question answering, and information retrieval [26]. To the best of our knowledge, this technique is not used for extracting relations from user stories by far.
- The other gap is a lack of efficiency [22]. The relationship models are to be constructed by engaging minimum resources. We are considerate of this issue in our work and involve two relation reduction steps to filter the noisy (unwanted) relations from results produced by OpenIE. The sole purpose of doing so is to minimize the cost and effort to be put into conceptual modeling based on these relations.
- The other challenge is incompleteness in the generated models, which may result in expensive rework later [18] and may derail the entire software project. The performance for extracting relations in the pioneering work of [37] is reported in terms of recall to be in 50-60% and precision to be in the 80–100% range, which is not very impressive [11]. We deal with this issue with an added focus on enhanced coverage of user stories; our minimal relations set is deemed to cover user stories maximally. We believe that as much of the user story is covered by the relation(s), more knowledge is retained in the relationship model, contributing to completeness.

Recent progress in relation extraction has been driven by neural architectures that jointly model entities and relations using contextualized pre-trained encoders. Span-based frameworks leverage contextual span representations and graph propagation to support cross-sentence reasoning and document-level extraction, improving performance on heterogeneous text [45]. Single-stage joint extraction models reformulate relation extraction as token-pair linking or sequence tagging, enabling effective handling of overlapping and nested relations [46]. Comprehensive surveys further highlight how transformer encoders, span propagation, and joint decoding have advanced relation-extraction automation and opened new research directions [47].

Open Information Extraction (OpenIE) has similarly evolved from rule-based and dependency-driven systems to neural and transformer-based architectures capable of abstractive and generative triple extraction. Surveys report improved extraction quality in neural OpenIE while noting the increased data, annotation, and compute requirements of these methods [48]. More recent reviews also describe the progression from rule-based models to large language models and hybrid extraction strategies [49].

Although neural RE and neural OpenIE methods demonstrate strong extraction performance, they typically require large annotated datasets, domain-specific fine-tuning, and significant computational resources. In contrast, our use of a deterministic OpenIE annotator combined with targeted subsumption-based reduction for relation extraction provides a lightweight, training-free solution that generalizes well to heterogeneous user-story text while keeping extraction transparent, explainable, and easy to integrate with reduction pipeline.

Our relation reduction process and evaluation metrics take inspiration from the software testing domain. Researchers have looked into the usage of test suite reduction techniques to address the issue of test suite size [40]. Techniques for lowering the size of the test suite, such as [41 - 43], try to do so by deleting unnecessary test cases from the original test suite in accordance with specified coverage requirements [39]. Regression testing's cost can be reduced by reducing the test suite, according to earlier research [38]. Along the same lines, we reduce

the relation set size by removing unnecessary relations with the objective of lowering the conceptual modeling and, further, the design elaboration cost.

3. The Proposed Approach

This section presents the proposed approach of extracting the set of relations from the user story and reducing it to a minimal set. In subsection 3.1, we discuss the preliminaries that the reader is expected to be equipped with in order to understand the relation extraction and reduction process presented in subsection 3.2.

3.1. Preliminaries

Here, we discuss the two preliminary concepts, Open Information Extraction (OpenIE) and Minimal Set of Relations, that need to be comprehended so as to understand the proposed process of extracting and reducing the set of relations.

3.1.1. Open information extraction (OpenIE)

OpenIE extracts binary relations from plain text. In a binary relation, two arguments are linked by the relation's name, so it creates a triple [25] (Subject; relation; Object).
For example, 'Mahatma Gandhi is regarded as the Father of Nation'would create a relation triple (Mahatma Gandhi→ is regarded as→ Father of Nation), that corresponds to the open domain relation 'is-regarded-as (Mahatma Gandhi, Father of Nation).'
OpenIE is ascertained to be useful in a number of NLP tasks, like relation extraction, question answering, and information retrieval [26].

3.1.2. Minimal set of relations

The minimal set of relations is the output that the proposed process is intended to produce. The minimal set of meaningful relations must contain necessary minimum relations. Here, by necessary minimum relations, we mean the following:

- The relations that cumulatively cover most of the domain knowledge that the user story is written to convey. The knowledge covered by the minimal set must be no less than the knowledge covered by the originally extracted set with a greater number of relations.
- No relation in the set must be such that it subsumes any other relation in the set. Every relation must convey some piece of additional domain knowledge that is not covered by any other relation in the set.

3.2. Relation extraction and reduction process

Here, we elaborate on the steps involved in the process of extracting the set of relations from the user stories and reducing it to the minimal set of relations. Fig. 1 depicts the proposed process. The process starts with the extraction of the initial set of relations from the user story by applying the Stanford OpenIE extractor [25]. The 'role' part of the input user story mentions the primary actor, which has references to the 'feature' and 'benefit' part of the user story. So, we identify the primary actor from the initial set of extracted relation triples. The initial set of extracted relations may contain superfluous relations that need to be eliminated from the set. In relation reduction I, some superfluous relations are removed based on certain keywords. As the open IE extractor is not able to resolve the primary actor references, we replace all these references with the primary actor in the subject and object part of all the relation triples in the reduced set. In relation reduction II, remaining superfluous relations are removed on the basis of subsumption criteria. As the final output, we obtain the minimal set of relation triples. Table 1 shows the output obtained from each of the process steps when performed on an example user story.

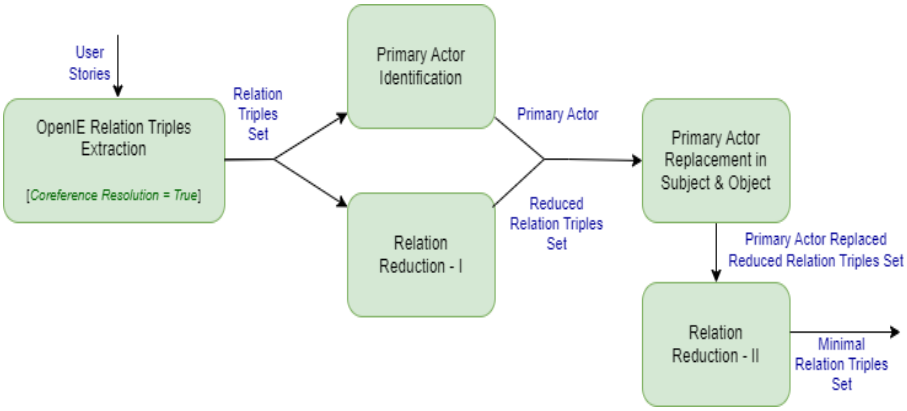

Fig. 1: Relation Extraction and Reduction Process.

3.2.1. OpenIE relation triples extraction

Firstly, we extract relation triples from input user stories using Stanford OpenIE annotator (openie) [25], which is a Java implementation of the open IE system described in [26]. As described in section 3.1.1, a relation triple (Subject; relation; Object) specifies a relation between a subject and an object.
The openie annotator depends on some other annotators to function: tokenize, ssplit, pos, lemma, depparse, natlog [25]. Additionally, it requires ner and coref for resolving coreferences. Table 2 lists and describes the annotators supporting openie.
Out of all the options to configure the behavior of openie, we keep most of them at their default values except resolve_coref (option to resolve coreferences). The default value of the option is false, but we change it to true so as to run the coreference resolver (and named entity recognizer as a dependency). The purpose is to replace pronominal mentions with their canonical mention in the relation triples extracted from the user story. In the example user story, US1, mentioned next:

US1 (User Story 1): As an administrator, I want to access the customer's master record so that I can make changes to it.
The token ‘it’ is resolved to be referring ‘customer’s master record’ in relation triples extraction.

Table 1: Process Steps and Respective Output for an Example User Story.

User Story: As an administrator, I want to access customer's master record so that I can make changes to it.
OpenIE Relation Triples Extraction
Initial Relation Triples Set:
customer→has→master record
I→want As→administrator
I→can make→changes
I→can make changes to→customer's master record
I→access→customer's master record
Primary Actor Identification
Primary Actor:
administrator
Relation Reduction – I
Reduced Relation Triples Set:
customer→has→master record
I→can make→changes
I→can make changes to→customer's master record
I→access→customer's master record
Primary Actor Replacement in Subject & Object
Primary actor replaced Reduced Relation Triples Set:
customer→has→master record
administrator→can make→changes
administrator→can make changes to→customer 's master record
administrator→access→customer's master record
Relation Reduction – II
Minimal Relation Triples Set:
administrator→access→customer's master record
administrator→can make changes to→customer 's master record
customer→has→master record

The openie, with the annotator and configuration options stated above, extracts the relation triples from the input user story. For example, the relations extracted from US1 are as follows:

R1_1: customer→has→master record
R1_2: I→want As→administrator
R1_3: I→can make→changes
R1_4: I→can make changes to→customer 's master record
R1_5: I→access→customer 's master record
For the sake of clarity and ease of understanding, we opt for the following notation to represent a relation:
subject→relation→object

We label the relations as R1_1, R1_2, R1_3, R1_4, and R1_5 respectively so as to refer to them in further sections. These relations form the initial set of relations extracted from the first step in our process.

Table 2: List of Annotators

Name	Task	Description	Example
tokenize	Tokenization	Turns text into tokens.	"John reads book." is turned into tokens: "John", "reads","book","."
ssplit	Sentence Splitting	Divides the text into sentences.	"John reads book. He is very studious." is divided into "John reads book." and "He is very studious."
Pos	Part of Speech Tagging	Applies part of speech tags to tokens.	"John" is assigned the tag "NNP" (proper noun).
lemma	Lemmatization	Maps the token to its lemma, the base dictionary form.	"reads" is mapped to "read".
Ner	Named Entity Recognition	Recognizes named entities in text.	"John" is recognized as "PERSON".
depparse	Dependency Parsing	Analyzes the grammatical structure of text by establishing relationships between its tokens.	"reads" relates to "John" through dependency "nsubj" (nominal subject).
coref	Coreference Resolution	Finds the references of the same entity in the text.	In "John reads book. He is very studious.", "He" refers to "John".
natlog	Natural Logic	Marks quantifier scope and polarity of the token, as per the natural logic semantics.	In "all cats have tails”, all is marked as a quantifier with subject scope [1, 2) and object scope [2, 4). "cats" is marked as a downward-polarity token, and all other tokens as upwards polarity [27].

It can be observed from these relations that despite coreference resolution being in place, the openie cannot resolve the pronoun 'I' (in relations R1_2 to R1_5) to the entity it refers to. The steps 3.2.2 and 3.2.4 deal with this issue. Other than that, there are some extraneous relations (relations R1_2 and R1_3, reasons discussed in upcoming sections) that would not serve any purpose in domain understanding and further in conceptual modeling and, thus, are subjected to be removed from the set to minimize it. The two reduction steps, 3.2.3 and 3.2.5, tackle this minimization problem.

3.2.2. Primary actor identification

As per the standard predefined format [30] of writing a user story, it has three aspects to capture about a requirement: Who requires the functionality, What functionality is required, and Why the functionality is required. According to the widely used Connextra template, the user story takes the following form:
"As a <role> I want to <feature/ functionality> so that <benefit>" [4].
Thus, a user story generally has three constituents [11, 29, 30], which [4] calls as role, means and ends parts [4]. Each of these parts has an indicator as a delimiter.
The role part involves the role indicator ('As a/an') and the role itself. This part talks about a primary actor who interacts with the system to invoke the desired feature/ functionality. We intend to identify this primary actor in the current step. The reason is that the relations extracted in step 3.2.1 do not have a direct mention of the primary actor. We only have the pronominal reference to it, and the coreference resolver is also not able to resolve the same.
Due to the indicator 'As a/an' delimiting the role part, we always find a relationship: I→want As→<role>, so we extract the object of this relation to be identified as the primary actor. For example, in the relation R1_2, the administrator is identified to be the primary actor.
As there may be a pronominal reference to the primary actor in both the subject and object part of the extracted relations, in step 3.2.4, we make use of the primary actor identified in this step to replace such references.

3.2.3. Relation reduction – I

Here, in this step, we minimize our initial set of relations obtained from step 3.2.1. For this minimization to take place, we apply the first level of reduction in relations based on keyword matching. After closely observing the relations extracted from a multitude of user stories, we found that certain relations with the keywords 'want' and 'want As' as the relation names are superfluous as they do not contribute any significant knowledge about the problem domain and thus are candidates of removal from the initial set of relations.
Out of the two keywords, our reduction is based on the 'want As' we already processed in step 3.2.1 for the primary actor identification. Apart from that, as a relation, this does not make any sense, and so is subjected to be removed. For example, we remove relation R1_2 from our initial set of relations extracted from the example user story US1 in step 3.2.1.
The other keyword, 'want', has also been witnessed to be a relation name in several superfluous relations. From the example user story, US2, as mentioned below:
US2: As a Staff, I want to search book by name so that I can quickly search for a book.
We get the following relations extracted by openie:

R2_1: I→want→I can search for book
R2_2: I→want→so I can search
R2_3: I→search→book
R2_4: I→want→so I can search for book
R2_5: I→search→book by name
R2_6: I→can quickly search for→book
R2_7: I→want→I can quickly search
R2_8: I→want→so I can quickly search
R2_9: I→want As→Staff
R2_10: I→can search for→book
R2_11: I→want→I can quickly search for book
R2_12: I→want→so I can quickly search for book
R2_13: I→want→I can search

Out of these extracted relations, 8 relations (R2_1, R2_2, R2_4, R2_7, R2_8, R2_11, R2_12, and R2_13) have the keyword 'want' mentioned as the relation's name. If we carefully analyze, the only fact that we are getting to know about is 'I can quickly search for book' mentioned in the 'object' position. The same knowledge can be obtained through only one relation triple R2_6 in the set. Realizing that all 8 relations convey no substantial conceptual knowledge about the problem domain, we remove them all from the initial set of relations.
It is worth mentioning here that 8 relations (R2_1, R2_2, R2_4, R2_7, R2_8, R2_11, R2_12, and R2_13) are reduced due to the 'want' keyword, and one relation (R2_9) is reduced due to 'want As' keyword. So, 9 out of 13 relations are reduced in this step. The reduced set of relations now contains the following relation triples:

R2_3: I→search→book
R2_5: I→search→book by name
R2_6: I→can quickly search for→book
R2_10: I→can search for→book

It can be easily deduced from the example that this step makes a significant reduction of relations in such cases of user stories.

3.2.4. Primary actor replacement in subject & object

In this step, we find the pronominal references to the primary actor in the subject and object of extracted relation triples and replace such references with the primary actor we identified in step 3.2.2.
In the subject part, we most commonly find a single reference to the primary actor in the form of the pronoun "I". For example, in 4 relation triples extracted from US1 (R1_2 to R1_5) and all the 13 relation triples extracted from US2 refer to the primary actor through the pronoun "I". But we also find certain exceptions to this where we have a phrase in the subject part containing multiple tokens, out of which there may be one token referring to the primary actor. The referring token may exist at any position in the phrase.
For example, From the user story US3 given below:
US3: As a visitor, I want to provide my personal details to purchase a ticket.
Openie extracts the following relationships as a result of step 3.2.1.

R3_1: I details→purchase→ticket
R3_2: I personal details→purchase→ticket
R3_3: I→want As→visitor
R3_4: I→provide→I details
R3_5: I→provide→I personal details

The step 3.2.2 identifies 'visitor' to be the primary actor from the relation R3_3. Step 3.2.3 removes relation R3_3 as a result of the first level of relation reduction. The reduced relation set will contain the following relations:

R3_1: I details→purchase→ticket
R3_2: I personal details→purchase→ticket
R3_4: I→provide→I details
R3_5: I→provide→I personal details

We may observe in the above set of relations that R3_4 & R3_5 have a single reference (in the form of single token "I" in subject) to the primary actor, and R3_1 & R3_2 have multiple token phrases as subject, having reference to the primary actor at the Ist position.
Likewise, we find such references in the object part as well. In relations R3_4 and R3_5, the phrases in the object part have reference to the primary actor at the Ist position.
The current step, 3.2.4, aims at replacing such pronominal references, in both subject and object, with the identified primary actor. In the running example, all the occurrences of 'I' will undergo replacement with the identified primary actor, i.e., 'visitor'. So, we shall get the following set of relations after the intended replacement is done:

R4_1: visitor details→purchase→ticket
R4_2: visitor personal details→purchase→ticket
R4_3: visitor→provide→visitor details
R4_4: visitor→provide→visitor personal details

3.2.5. Relation reduction – II

In this step, we apply the second level of reduction to the primary actor replaced reduced relation set obtained as an output from step 3.2.4. The basis for reduction here in this step is the 'subsumption' criteria.
Subsumption Criteria. A relation, say R1, is said to be subsumed by some other relation, say R2, if all the tokens involved in all the three parts (subject, relation, and object) of R1, irrespective of the order, are cumulatively present in R2. For example, the relation R4_3 is subsumed by R4_4 as all the tokens of R4_3 (visitor, provide, visitor, details) are present in R4_4 (visitor, provide, visitor personal, details). We call R1 to be a subsumed relation and R2 to be a subsuming relation.
The idea behind checking subsumption is that the subsumed relation conveys no additional knowledge than the knowledge conveyed by the subsuming relation as the subsumed relation is fully contained in the subsuming relation.
As per the criteria stated above, we iteratively check for the subsumption of every relation in the reduced relation set obtained after step 3.2.4 with every other relation in the same set. All the subsumed relations are subsequently removed from the set. Once we are done with this step, we get the minimal set of relations (as discussed in section 3.1.2) as the final output.
Example of Subsumption:
Consider the following two relations extracted after primary actor replacement:

R4_3: visitor → provide → visitor details
R4_4: visitor → provide → visitor personal details
Token sets:
Tokens(R4_3) = {visitor, provide, visitor, details}
Tokens(R4_4) = {visitor, provide, visitor, personal, details}

Since all tokens of R4_3 appear in R4_4, R4_3 is fully contained within R4_4.
Therefore, R4_3 is subsumed by R4_4, and we remove R4_3 in the final minimal set.
The reduction performed in this step will result in the minimal set of relations for user story US3 having the following relations:

R4_2: visitor personal details→purchase→ticket
R4_4: visitor→provide→visitor personal details

3.2.6. Algorithmic specification

To complement the descriptive explanation of the Relation Reduction Steps I & II, we present the algorithms (pseudocode form) of the two reduction steps. The pseudocode highlights the operational logic of keyword-based filtering (Reduction I) and subsumption-based minimization (Reduction II), making the process more explicit and reproducible.

Algorithm 1: Relation Reduction I (Keyword Filtering)
This step removes relations that contain auxiliary constructs (e.g., want, want As) which do not contribute meaningful domain semantics.

Input: InitialRelations R
Output: FilteredRelations R1
R1 ← ∅
for each relation r in R do
 if relation_name(r) contains "want" or "want As" then
 continue // discard superfluous relation
 end if

R1 ← R1 ∪ {r}
end for
return R1

Algorithm 2: Relation Reduction II (Subsumption-Based Minimization)
After primary actor replacement, the remaining relations are examined pairwise. A relation is removed if all its constituent tokens (subject, relation phrase, object) are fully contained in another relation, indicating no additional domain knowledge.

Input: ActorReplacedRelations R1
Output: MinimalRelations R2
R2 ← R1
for each relation r1 in R1 do
for each relation r2 in R1 do
if r1 ≠ r2 and isSubsumed(r1, r2) then
R2 ← R2 – {r1} // remove the subsumed relation
break
end if
end for
end for
return R2

The subsumption test operates on the token sets of the two relations:
Function isSubsumed(r1, r2):

Tokens1 ← tokens(subject(r1)) ∪ tokens(relation(r1)) ∪ tokens(object(r1))
Tokens2 ← tokens(subject(r2)) ∪ tokens(relation(r2)) ∪ tokens(object(r2))
if Tokens1 ⊆ Tokens2 then
return true
else
return false
end if

3.2.7. Computational complexity

The computational cost of each step is as follows:
Keyword Filtering (Reduction I):
For n extracted relations, the filtering step performs a single pass over the relation list. Time Complexity: O(n).
Primary Actor Replacement: Each relation is scanned once; therefore, complexity is O(n).
Subsumption Analysis (Reduction II): Each relation is compared pairwise with every other relation, and token-level subset checking is performed.
If t is the maximum number of tokens in a relation, the complexity is:
$O(n^2 \cdot t)$. Since t is small and bounded, this effectively behaves as $O(n^2)$.
Overall Complexity: The end-to-end reduction process is dominated by the subsumption check, giving: $O(n^2)$.

4. Evaluation and Results

In this section, we evaluate our proposed approach in light of the following research questions:
RQ1. How much reduction our approach is able to make in the number of relations obtained originally using openie?
As discussed in section 3.2, we apply two-step reductions on the relations extracted using openie. RQ1 aims at measuring and evaluating percentage reduction in the number of relations. This gives an insight into the usefulness of our relation minimization approach.
RQ2. How much coverage of the user story is achieved by the minimal set of relations obtained by the application of our approach?
The higher the coverage, the more the utility of the minimal relations set. The proposed work aims at maximizing the coverage of domain knowledge through the minimum number of relations. RQ2 aims at measuring and evaluating the percentage coverage of the input user story done by the minimal relations set.

4.1. Datasets

We evaluate the proposed approach on two data sets related to two well-known systems, namely, the Library Management System (LMS) and the Restaurant Management System (RMS). The datasets are available online [31]. The description of the datasets is mentioned in Table 3. The reasons for choosing the datasets are: firstly, both the datasets contain a significantly higher number of user stories, and secondly, the user stories are related to two very recognizable problem domains of Library and Restaurant.
The evaluation datasets have 384 (LMS) and 389 (RMS) user stories, out of which 353 (LMS) and 377 (RMS) are successfully processed, i.e., we are able to extract and minimize relations from these user stories.

Table 3: Description of Datasets

# of User Stories	LMS	RMS
Total	384	389
Processed	353	377
Unprocessed	31	12
Format Incompatible	12	5

Out of the 31 (LMS) and 12 (RMS) user stories left unprocessed, 12 (LMS) and 5 (RMS) user stories are found format incompatible, i.e., the user stories are not written as per the expected format: "As a/an <role>, I want <function>, so that <benefit>." For the other remaining unprocessed user stories, 19 (LMS) and 7 (RMS), the openie itself was not able to extract any meaningful relation. This calls for further investigation to find the reasons for the same, but here, we opt not to do so as it is out of the scope of the current work.

4.2. Evaluation metrics

The objective of the proposed work is to minimize the set of relations and maximize the coverage of the user story. In line with the objective, we devise two metrics, Relation Set Reduction (RSR) and Relation Set Coverage (RSC), to evaluate our approach.
Relation Set Reduction (RSR). RSR measures the amount of reduction done in the set of relations obtained after applying the two steps of reductions discussed in 3.2.3 and 3.2.5, respectively. RSR is calculated as per the following formula:

RSR = ((Original Relation Set Size – Reduced Relation Set Size)/ Original Relation Set Size)*100

RSR is first calculated for each of the user stories in the set and is then calculated cumulatively for the entire dataset.
Relation Set Coverage (RSC). As stated earlier, the aim is not just to minimize the set of relations but also to cover the user story in question maximally. The user story coverage indicates coverage of the domain knowledge being conveyed by the user story. The calculation of the RSC is based on how many different tokens (words) of the user story are involved in the minimal set of relations we managed to obtain. We calculate RSC using the below-mentioned formula:

RSC = ((Tokens in the minimal relation set)/ (Tokens in the user story – Stop words - Punctuation tokens)) * 100

For the user story US3, mentioned in section 3.2.4, Table 4 exemplifies the calculation of RSC:

Table 4: RSC Calculation Example

As	a	visitor	,	I	want	to	provide	my	personal	details	to	purchase	a	ticket	.
Tokens in the user story									=	16					
Tokens in the minimal relation set									=	6					
Stopwords									=	7					
Punctuation tokens									=	2					
RSC (%)									=	85.71					

The calculation of RSC calls for removing stopwords (e.g., 'As', 'a', 'I', 'want', 'to') and punctuation tokens (e.g., ',', '.')

4.3. Results and discussion

RQ1. Table 5 presents the results of applying two-step reductions on the number of relations we originally extracted using openie. Column 1 lists the datasets. Column 2 in the table presents the total number of relations originally obtained from all the user stories in the respective dataset. Columns 3 and 4 show the number of relations in the set after applying relation reduction I and II, respectively.
Through the relation reduction I step, we are able to achieve a 29.14 % relation reduction for LMS and a 25.16 % reduction for RMS. Relation reduction II further makes us able to achieve 50.85 % and 55.44 % reduction for the data sets, LMS and RMS, respectively.

Table 5: Relation Set Reduction (RSR) Results

	Number of Relationships			RSR			
	Original Set	Reduced Set - I	Reduced Set - II	Cumulative	Maximum	Minimum	Average
LMS	1740	1233	606	65.17	93.75	25	61.25
RMS	1856	1389	619	66.65	98.08	25	60.4

The comparison between the steps, relation reduction – I and relation reduction – II, is shown in Fig. 2.

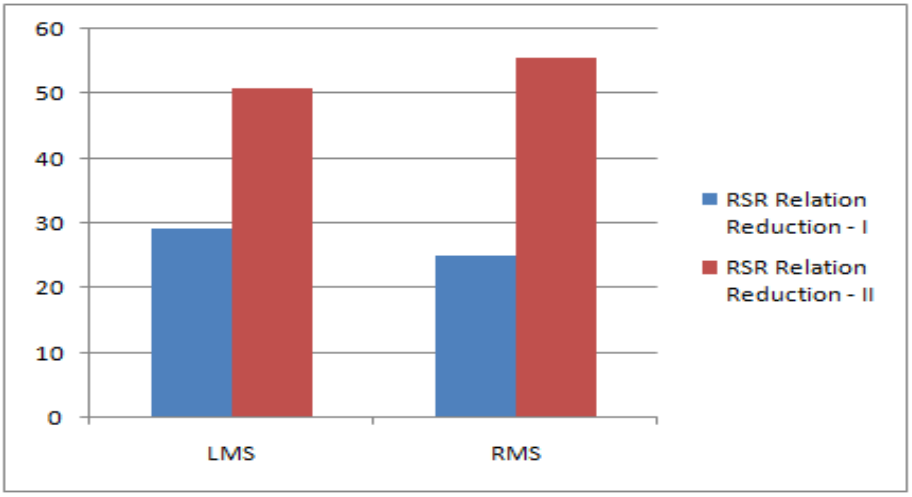

Fig. 2: Comparison between the Two Reduction Steps.

The results clearly indicate that both the steps have marked significant reduction, but in comparison to relation reduction I, relation reduction II achieves higher reduction, i.e., nearly double the former step.

For both the evaluation data sets, columns 5, 6, 7, and 8 show the cumulative, maximum, minimum, and average RSR values, respectively. If we take the average of values obtained for both the data sets, then the performance of the proposed approach is evaluated to be noteworthy as we are able to achieve a cumulative RSR of 65.91 %, the maximum RSR we touched is as high as 95.92%, the RSR at its minimum across the dataset is 25% and the average RSR is 60.83%. The results are encouraging as the proposed approach is able to filter out nearly 66% of relations that are superfluous and noisy. This reduction will result in saving significant amount of cost and effort to be engaged in the development phases that stand on these relations.

RQ2. Table 6 shows the user story coverage results for both evaluation datasets. Column 2 shows the sum of the number of unique tokens per user story for the datasets listed in column 1. Column 3 shows the sum of unique tokens in the minimal set of relations extracted per user story.

Table 6: Relation Set Coverage (RSC) Results

	Number of Unique Tokens			RSC			
	User Story	Relations	Stopwords	Cumulative	Maximum	Minimum	Average
LMS	5959	2558	2937	84.65	100	20	88.74
RMS	6200	2680	3163	88.24	100	27.27	91.52

Here, we consider only those tokens in the relations that are present in the input user story. We would like to mention that sometimes openie constructs relations with additional tokens that are not there in the user story. For example, consider the relation "customer→has→master record" mentioned in Table 1. This relation has been extracted on the basis of the segment "customer's master record" of the user story. The apostrophe ('), being the case of possession, resulted in the relation 'has' between 'customer' and 'master record'. Though the relation 'has' is meaningful and is part of the minimal relation set, we exclude it from the coverage calculation as it is not present in the user story as a token.

Column 4 shows the no. of unique stop words present in the user story. Stop words are commonly used words in user stories, the removal of which allows focus on the important tokens. Our set of stop words involves only the frequently occurring stop words in the user stories: [As, a, an ,, , I, want, to, the, so, that, .]. We keep this set small so as not to project a false sense of coverage. For the sake of simplicity, the punctuation tokens (',' and '.') are also taken to be part of the stop words set.

Column 5 presents the values of cumulative RSC calculated on the basis of values in columns 2, 3, and 4 as per the formula described in section 3.2 and exemplified in Table 4. For LMS, we achieved 84.65% coverage, and for RMS, we attained 88.24% coverage. These results are encouraging as for both the data sets taken together, we are able to cover 86.45 % of the user story tokens through our minimal relations set. On an average, we attain 90.13 % coverage. At the maximum, we achieved 100% complete coverage, and at the minimum, we managed to obtain 23.64 % coverage. To summarize, with the minimum number of relations, we are able to withhold as much as 86.45 % of the domain knowledge imbibed in the user stories. This would make artifacts, such as conceptual models, to be produced on the basis of such knowledge to be more accurate, complete, and relevant.

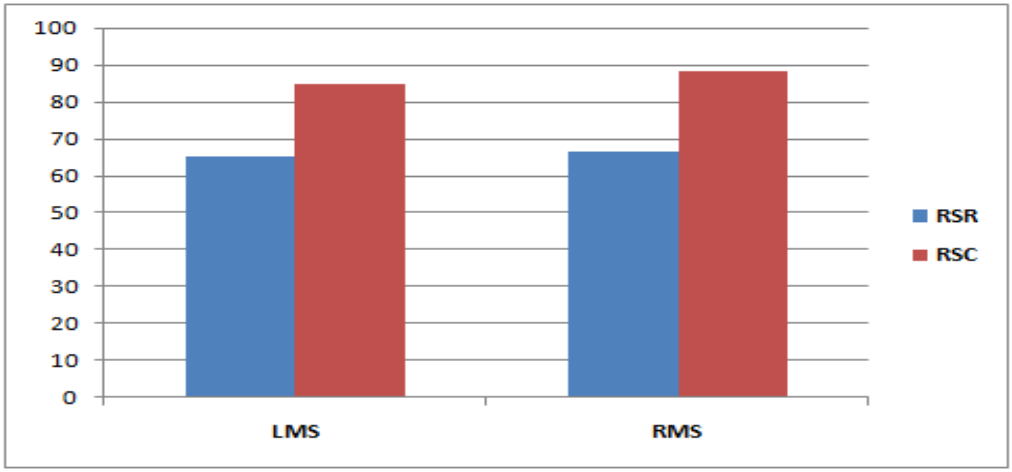

Fig. 3: RSR and RSC for the Evaluation Datasets.

Figure 3 puts together the cumulative RSR and RSC values obtained for the evaluation data sets. It can be observed that for LMS, with the relation set reduced up to nearly 65 %, we are able to achieve 85% coverage, and for RMS, with nearly 67% reduction in the relation set, we are able to attain 88% coverage. These results are cheering as we are able to fulfill the objective set forth: maximal coverage of user story through minimal relation set. Another interesting observation to make in Fig 3 is that the approach performs consistently on both the data sets with a slight difference of 1.5 in reduction (RSR) and 3.6 in coverage (RSC).

4.4. Error analysis of unprocessed user stories

A small number of user stories in both datasets remained unprocessed (12 out of 389 in RMS, and 31 out of 384 in LMS). Examination of these cases shows that they typically contain implicit or weak predicates ("see", "know"), auxiliary-heavy constructions ("be able to edit…", "be able to see…"), or descriptive noun clauses without a clear verb ("what type of books are borrowed the most", "which book has been read the most"). These forms do not provide a stable subject→verb→object pattern, causing OpenIE to skip extraction or produce incomplete triples. Several stories also include modifier-heavy expressions, such as temporal, quantitative, or comparative phrases ("every week and month", "more than one book", "remaining time for returning books"), which OpenIE generally ignores.

These unprocessed stories represent cases where the linguistic form inherently resists predicate-based extraction, rather than limitations of the reduction process itself. Given that they constitute a very small fraction of the data, their impact on the overall results is minimal. Nonetheless, they highlight natural boundaries of OpenIE-based extraction and identify opportunities for future work, such as paraphrasing or predicate normalization, to further reduce omissions.

4.5. Impact on downstream modeling

The minimal relation set produced by our reduction process directly benefits downstream domain modeling tasks such as class diagram generation, entity identification, and dependency analysis. By eliminating redundant or overlapping relations, the final relation set highlights the relationships and the core domain concepts involved in them as subject and object. This reduces ambiguity and noise during model derivation and leads to more stable and accurate conceptual models. In practice, this means fewer conflicting candidate classes, clearer relation identification, and a reduced likelihood of misinterpreting domain knowledge.

4.6. Implementation

For Open Information Extraction, we use Stanford CoreNLP's 'openie' annotator [26]. We implemented reduction logic using Java. We present the output in the form of an Excel file. The output file contains the user stories, relations extracted, values concerning the reduction metric, RSR, and the values related to the coverage metric, RSC. For Excel file generation from the code, we use Apache POI [44]. Our implementation is available at https://github.com/amolsha/ConceptualModelExtractor.

4.7. Threats to validity

Regarding external validity, though the datasets chosen for evaluation are of large size, whether these data sets are typical of user stories in general is not clear. Furthermore, additional case studies may be performed to check the applicability of the proposed approach in industrial setup.
With regard to construct validity, we mention that our evaluation is based on token-based coverage and does not involve coverage of tacit knowledge. The tacit knowledge is implicit in nature and needs to be made explicit through manual efforts in order to be counted upon. Including tacit knowledge in coverage criteria requires further studies.
RSR and RSC measures alone may not guarantee full conceptual correctness. Nevertheless, adequate coverage and reduced redundancy are essential prerequisites, as missing or noisy relations would limit the quality of any resulting conceptual model.
Threats to the internal validity of the experiments centre on their methodology. However, since metric values are calculated automatically, there is no chance of bias or inaccuracy. However, certain aspects of calculation need a more in-depth investigation, such as the stop words list we created using the datasets in question. In order to arrive at a more precise measurement, we need to customize the list as per the more general cases of user stories.

5. Conclusion

Despite being the most widely adopted notation for expressing requirements, user stories are not able to provide a holistic view comprising relations among key concepts of the system. Aligned with ongoing research on the subject, we also believe that relations are to be extracted from the NL user stories to represent the system through a conceptual model.
We have presented an approach to extract relations from user stories using the state-of-the-art OpenIE NLP technique. We further applied two-step reductions to reduce the initial set of relations into a minimal set of relations. As the final set consists of relations that subsume all the filtered relations, the output from the approach is guaranteed to provide high coverage of the input user stories.
We demonstrated, through our evaluation, that the proposed approach significantly reduces the relation and ensures high coverage of the user stories. The results indicate the usefulness of the proposed approach. The minimal set of relations we obtained as the final output can be used for design elaboration. The processing of fewer relations will cause minimum time and effort consumption, and due to high coverage, the resultant design artifact will cover maximum domain knowledge.
We intend to make use of the results obtained in the present work to arrive at more formal, concise, and complete design models focusing on classes of objects along with their attributes, interrelationships, multiplicities, etc. As our coverage criterion is token-based coverage, it does not cover up the tacit knowledge. Through the explication of tacit knowledge by the experts, we further aim to make provisions in our approach to check the coverage of tacit knowledge along with the explicit one.

References

[1] Yue, T., Briand, L. C., &Labiche, Y. (2011). A systematic review of transformation approaches between user requirements and analysis models. Requirements Engineering, 16(2), 75–99. https://doi.org/10.1007/s00766-010-0111-y.

[2] Li, Y., Schulze, S., Scherrebeck, H. H., &Fogdal, T. S. (2020). Automated Extraction of Domain Knowledge in Practice: The Case of Feature Extraction from Requirements at Danfoss. *Proceedings of the 24th ACM Conference on Systems and Software Product Line: Volume A - Volume A*. https://doi.org/10.1145/3382025.3414968.

[3] Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006.

[4] Cohn, M. (2004). User Stories Applied: for Agile Software Development. Addison Wesley.

[5] Kassab, M. (2014). An Empirical Study on the Requirements Engineering Practices for Agile Software Development. *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, 254–261. https://doi.org/10.1109/SEAA.2014.77.

[6] Lucassen, G., Dalpiaz, F., Werf, J. M., &Brinkkemper, S. (2016). The Use and Effectiveness of User Stories in Practice. *Proceedings of the 22nd International Working Conference on Requirements Engineering: Foundation for Software Quality - Volume 9619*, 205–222. https://doi.org/10.1007/978-3-319-30282-9_14.

[7] Müter Laurens and Deoskar, T. and M. M. and B. S. and D. F. (2019). Refinement of User Stories into Backlog Items: Linguistic Structure and Action Verbs. In M. Knauss Eric. https://doi.org/10.1007/978-3-030-15538-4_7.

[8] and Goedicke (Ed.), Requirements Engineering: Foundation for Software Quality (pp. 109–116). Springer International Publishing. https://doi.org/10.1007/978-3-030-15538-4_7.

[9] Berends, J., & Dalpiaz, F. (2021). Refining User Stories via Example Mapping: An Empirical Investigation. 2021 IEEE 29th International Requirements Engineering Conference (RE), 345–355. https://doi.org/10.1109/RE51729.2021.00038.

[10] Brambilla, M., Cabot, J., & Wimmer, M. (2012). Model-Driven Software Engineering in Practice. Springer International Publishing. https://doi.org/10.1007/978-3-031-02546-4.

[11] Ibrahim, M., & Ahmad, R. (2010). Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques. *2010 Second International Conference on Computer Research and Development*, 200–204. https://doi.org/10.1109/ICCRD.2010.71.

[12] Lucassen, G., Robeer, M., Dalpiaz, F., van der Werf, J. M. E. M., &Brinkkemper, S. (2017). Extracting conceptual models from user stories with Visual Narrator. *Requirements Engineering*, *22*(3), 339–358. https://doi.org/10.1007/s00766-017-0270-1.

[13] Elallaoui, M., Nafil, K., &Touahni, R. (2015). Automatic generation of UML sequence diagrams from user stories in Scrum process. *2015 10th International Conference on Intelligent Systems: Theories and Applications, SITA 2015*. https://doi.org/10.1109/SITA.2015.7358415.

[14] Elallaoui, M., Nafil, K., &Touahni, R. (2018). Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques. *Procedia Computer Science*, *130*, 42–49. https://doi.org/10.1016/j.procs.2018.04.010.

[15] Gupta, A., Poels, G., &Bera, P. (2019). Creation of multiple conceptual models from user stories – a natural language processing approach. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11787 LNCS*, 47–57. https://doi.org/10.1007/978-3-030-34146-6_5.

[16] Gunes, T., &Aydemir, F. B. (2020). Automated Goal Model Extraction from User Stories Using NLP. *Proceedings of the IEEE International Conference on Requirements Engineering*, *2020-Augus*, 382–387. https://doi.org/10.1109/RE48521.2020.00052.

[17] Gunes, T., Oz, C. A., &Aydemir, F. B. (2021). ArTu: A Tool for Generating Goal Models from User Stories. *Proceedings of the IEEE International Conference on Requirements Engineering*, 436–437. https://doi.org/10.1109/RE51729.2021.00058.

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

[19] Javed, M., & Lin, Y. (2021). iMER: Iterative process of entity relationship and business process model extraction from the requirements. *Information and Software Technology*, *135*, 106558. https://doi.org/10.1016/j.infsof.2021.106558.

[20] Nasiri, S., Adadi, A., &Lahmer, M. (2023). Automatic generation of business process models from user stories. *International Journal of Electrical and Computer Engineering (IJECE)*, *13*(1), 809. https://doi.org/10.11591/ijece.v13i1.pp809-822.

[21] Bragilovski, M., Dalpiaz, F., & Sturm, A. (2022). Guided Derivation of Conceptual Models from User Stories: A Controlled Experiment. *Requirements Engineering: Foundation for Software Quality: 28th International Working Conference, REFSQ 2022, Birmingham, UK, March 21–24, 2022, Proceedings*, 131–147. https://doi.org/10.1007/978-3-030-98464-9_11.

[22] Raharjana, I. K., Siahaan, D., &Fatichah, C. (2021). User Stories and Natural Language Processing: A Systematic Literature Review. *IEEE Access*, *9*, 53811–53826. https://doi.org/10.1109/ACCESS.2021.3070606.

[23] Yue, T., Briand, L. C., &Labiche, Y. (2015). AToucan: An Automated Framework to Derive UML Analysis Models from Use Case Models. *ACM Trans. Softw. Eng. Methodol.*,*24*(3). https://doi.org/10.1145/2699697.

[24] J. Becker, M. Rosemann, C. Von Uthmann, Guidelines of business process modeling, in: Business process management, Springer, Berlin, Heidelberg, 2000, pp. 30–49. https://doi.org/10.1007/3-540-45594-9_3.

[25] Neill CJ, Laplante PA (2003) Requirements engineering: the state of the practice. IEEE Softw 20(6):40. https://doi.org/10.1109/MS.2003.1241365.

[26] *Stanford Open Information Extraction*. (n.d.). Retrieved September 16, 2023, from https://nlp.stanford.edu/software/openie.html

[27] Angeli, G., Johnson Premkumar, M. J., & Manning, C. D. (2015). Leveraging Linguistic Structure for Open Domain Information Extraction. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 344–354. https://doi.org/10.3115/v1/P15-1034.

[28] Natural Logic. (n.d.). Retrieved September 16, 2023, from https://stanfordnlp.github.io/CoreNLP/natlog.html

[29] Full List of Annotators. (n.d.). Retrieved September 16, 2023, from https://stanfordnlp.github.io/CoreNLP/annotators.html

[30] Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., &Brinkkemper, S. (2016). Improving agile requirements: the Quality User Story framework and tool. *Requirements Engineering*, *21*(3), 383–403. https://doi.org/10.1007/s00766-016-0250-x.

[31] Wautelet Yves and Heng, S. and K. M. and M. I. (2014). Unifying and Extending User Story Models. In J. and Q. C. and R. C. and M. Y. and M. H. and H. J. Jarke Matthias and Mylopoulos (Ed.), Advanced Information Systems Engineering (pp. 211–225). Springer International Publishing. https://doi.org/10.1007/978-3-319-07881-6_15.

[32] Kose, S. G., & Aydemir, F. B. (2023). *A User Story Dataset for Library and Restaurant Management Systems*. Zenodo.

[33] Arora, C., Sabetzadeh, M., Briand, L., & Zimmer, F. (2016). Extracting domain models from natural-language requirements: Approach and industrial evaluation. *Proceedings - 19th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2016*, 250–260. https://doi.org/10.1145/2976767.2976769.

[34] Robeer, M., Lucassen, G., van der Werf, J. M. E. M., Dalpiaz, F., & Brinkkemper, S. (2016). Automated Extraction of Conceptual Models from User Stories via NLP. *Proceedings - 2016 IEEE 24th International Requirements Engineering Conference, RE 2016*, 196–205. https://doi.org/10.1109/RE.2016.40.

[35] Liu, L., Li, T., & Kou, X. (2014). Eliciting Relations from Natural Language Requirements Documents Based on Linguistic and Statistical Analysis. *2014 IEEE 38th Annual Computer Software and Applications Conference*, 191–200. https://doi.org/10.1109/COMPSAC.2014.27.

[36] Berry Daniel and Gacitua, R. and S. P. and T. S. F. (2012). The Case for Dumb Requirements Engineering Tools. In D. Regnell Björn and Damian (Ed.), *Requirements Engineering: Foundation for Software Quality* (pp. 211–217). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-28714-5_18.

[37] Slankas, J., Xiao, X., Williams, L., & Xie, T. (2014). Relation extraction for inferring access control rules from natural language artifacts. *Proceedings of the 30th Annual Computer Security Applications Conference*, 366–375. https://doi.org/10.1145/2664243.2664280.

[38] Vidya Sagar, V. B. R., & Abirami, S. (2014). Conceptual modeling of natural language functional requirements. *Journal of Systems and Software*, *88*(1), 25–41. https://doi.org/10.1016/j.jss.2013.08.036.

[39] Rehman Khan, S. U., Lee, S. P., Parizi, R. M., & Elahi, M. (2014). A code coverage-based test suite reduction and prioritization framework. *2014 4th World Congress on Information and Communication Technologies (WICT 2014)*, 229–234. https://doi.org/10.1109/WICT.2014.7076910.

[40] Yoo, S., & Harman, M. (2012). Regression Testing Minimization, Selection and Prioritization: A Survey. Softw. Test. Verif. Reliab., 22(2), 67–120. https://doi.org/10.1002/stv.430.

[41] ur Rehman Khan, S., ur Rehman, I., & Malik, S. U. R. (2009). The impact of test case reduction and prioritization on software testing effectiveness. *2009 International Conference on Emerging Technologies*, 416–421. https://doi.org/10.1109/ICET.2009.5353136.

[42] Khan, S.-R., Nadeem, A., & Awais, A. (2006). TestFilter: A Statement-Coverage Based Test Case Reduction Technique. *2006 IEEE International Multitopic Conference*, 275–280. https://doi.org/10.1109/INMIC.2006.358177.

[43] Hao, D., Zhang, L., Wu, X., Mei, H., & Rothermel, G. (2012). On-demand test suite reduction. *2012 34th International Conference on Software Engineering (ICSE)*, 738–748. https://doi.org/10.1109/ICSE.2012.6227144.

[44] Harrold, M. J., Gupta, R., & Soffa, M. lou. (1993). A Methodology for Controlling the Size of a Test Suite. ACM Trans. Softw. Eng. Methodol., 2(3), 270–285. https://doi.org/10.1145/152388.152391.

[45] *Apache POI*. (n.d.). Retrieved August 7, 2023, from https://poi.apache.org/index.html.

[46] Wadden D., Wennberg U., Luan Y., and Hajishirzi H. (2019). Entity, Relation, and Event Extraction with Contextualized Span Representations. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics. 5784–5789. https://doi.org/10.18653/v1/D19-1585.

[47] Wang Y., Yu B., Zhang Y., Liu T., Zhu H., and Sun L. (2020). TPLinker: Single-stage Joint Extraction of Entities and Relations Through Token Pair Linking. *Proceedings of the 28th International Conference on Computational Linguistics*, 1572–1582. https://doi.org/10.18653/v1/2020.coling-main.138.

[48] Zhao X. *et al.* (2024). A Comprehensive Survey on Relation Extraction: Recent Advances and New Frontiers. *ACM Comput. Surv.*, vol. 56, no. 11. https://doi.org/10.1145/3674501.

[49] Zhou S. *et al.* (2022). A Survey on Neural Open Information Extraction: Current Status and Future Directions. 5660–5667. https://www.ijcai.org/proceedings/2022/0793.pdf.

[50] L. Pai *et al.* (2024). A Survey on Open Information Extraction from Rule-based Model to Large Language Model. *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics. 9586–9608. https://doi.org/10.18653/v1/2024.findings-emnlp.560.