Extracting Core Elements of TFM Functional Characteristics from Stanford CoreNLP Application Outcomes

Erika Nazaruka[a], Jānis Osis[b] and Viktorija Griberman[c]
Department of Applied Computer Science, Riga Technical University, Sētas Iela 1, Riga, Latvia

Keywords: Knowledge Acquisition, Natural Language Processing, Stanford Corenlp, Functional Feature, Topological Functioning Model, Computation Independent Model.

Abstract: Stanford CoreNLP is the Natural Language Processing (NLP) pipeline that allow analysing text at paragraph, sentence and word levels. Its outcomes can be used for extracting core elements of functional characteristics of the Topological Functioning Model (TFM). The TFM elements form the core of the knowledge model kept in the knowledge base. The knowledge model ought to be the core source for further model transformations up to source code. This paper presents research on main steps of processing Stanford CoreNLP application results to extract actions, objects, results and executors of the functional characteristics. The obtained results illustrate that such processing can be useful, however, requires text with rigour, and even uniform, structure of sentences as well as attention to the possible parsing errors.

1 INTRODUCTION

Software development based on principles of Object Management Group's Model Driven Architecture (Miller and Mukerji 2001) considers models as a core of the development process. Model Driven Architecture (MDA) suggests using a chain of model transformations, namely, from a computation independent model (CIM) to a platform independent model (PIM), then to a platform specific model (PSM) and to source code.

In our vision of implementation of those principles (Figure 1), we suggest using a knowledge model based on the Topological Functioning Model (TFM) as the CIM to generate code via an intermediary model – Topological UML model (Osis and Donins 2017). The TFM elaborated by Janis Osis at Riga Technical University (Osis 1969) specifies a functioning system from three viewpoints – functional, behavioural and structural. This model can serve as a core model for further system and software domain analysis and transformations to design models and code (Osis and Asnina 2011b). Extraction of TFM elements requires textual description of functionality of the system. At the present, we have manual processing of the unstructured, but processed text, and automated processing of use case specifications in the form of semi-structured text (Osis and Slihte 2010; Slihte et al. 2011). In the latter, results are kept in XMI (XML Metadata Interchange) files using XML (eXtensible Markup Language) structures.

The new approach (Figure 1) supposes using a Natural Language Processing (NLP) pipeline for text processing (Nazaruka and Osis 2018) and a knowledge base (Nazaruks and Osis 2017; Nazaruks and Osis 2018) for keeping and managing results of the processing. The aim is to gain from NLP, Natural Language Understanding, and an inferring mechanism and flexibility of the knowledge base. Advantages of using the knowledge base are discovering conflicts in knowledge, managing synonyms, inferring new knowledge from the existing one.

In practice, preparation of text and manual knowledge acquisition are too resource-consuming (Elstermann and Heuser 2016). It is better either to skip the step of preparation of text and start from human analysis of the available information, either to

[a] https://orcid.org/0000-0002-1731-989X
[b] https://orcid.org/0000-0003-3774-4233
[c] https://orcid.org/0000-0002-8368-9362

Nazaruka, E., Osis, J. and Griberman, V.
Extracting Core Elements of TFM Functional Characteristics from Stanford CoreNLP Application Outcomes.
DOI: 10.5220/0007831605910602
In *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2019)*, pages 591-602
ISBN: 978-989-758-375-9

automate or semi-automate this process. We are on the automation way.

The goal of this research is to outline steps for processing Standford CoreNLP outcomes in order to achieve automated knowledge acquisition of the core elements of the TFM functional characteristics.

The paper is organized as follows. Section 2 presents overview of related work in the field. Section 3 describes the core elements of the TFM functional characteristics, how Stanford CoreNLP is used now and what is required to achieve our aims. Section 4 presents steps for processing CoreNLP outcomes, demonstrates them using the example as well as main results and limitations. Section 5 concludes the paper.

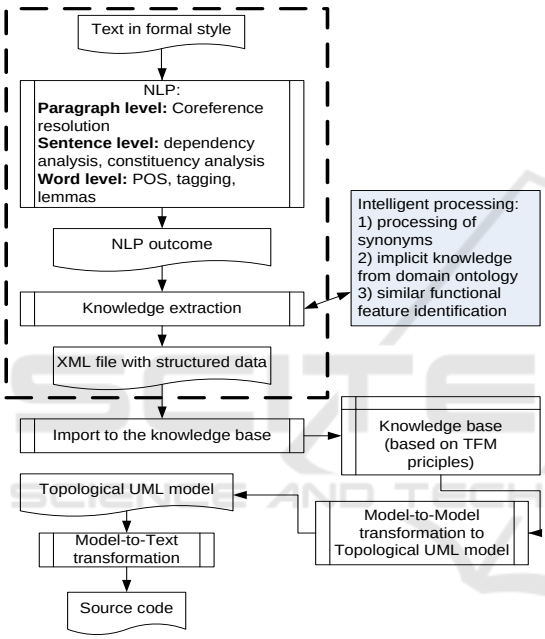

Figure 1: Intelligent Software Development.

2 RELATED WORK

Creation of models using knowledge extraction from different types of media is quite important since it may reduce time for analysis of large amount of information.

Creation of models and UML diagrams from textual documents is presented in several researches. For instance, creation of use case diagrams (Jabbarin and Arman 2014) and UML Activity Diagrams using identification of simple verbal sentences (Nassar and Khamayseh 2015) from textual requirements in Arabic. Creation of UML class diagrams from textual requirements (Krishnan and Samuel 2010), and from use case descriptions (Elbendak et al. 2011) in English.

Analysis of textual user requirements in natural language and requirements engineering diagrams can be used to create the Use Case Path model, the Hybrid Activity Diagrams model and the Domain model (Ilieva and Ormandjieva 2006). As Ilieva and Ormandjieva (2006) mentioned the standard way for automatic model creation from text is transformation of text in natural language to the one in formal natural language then to the intermediate model and then to the target requirements engineering model. For text analysis the authors applied syntax analysis by MBT tagger, semantics analysis to discover roles of words in the sentence (subject, predicate and object) and connections among them and then created a semantic network for text models. At the last step, the authors transformed this semantic network to one of the mentioned models using patterns.

Natural language analysis can be used for automated composition of conceptual diagrams (Bhala et al. 2014). The authors also noted a need for human participation, as well as several issues of natural language itself, i.e., sentence structures may have different forms that are not completely predictable, syntactical correctness of sentences, as well as ambiguity in determining attributes as aggregations and in generalization.

The overview of existing solutions in the field of UML model creation from textual requirements and business process models creation from textual documents (Osman and Zalhan 2016) showed that existing tools allow creating Class diagrams, Object diagrams, Use Case diagrams, and several of them provide composition of Sequence, Collaboration and Activity diagrams. All the solutions have certain limitations: some require user intervention, some cannot perform analysis of irrelevant classes, some require structuring text in a certain form before processing, and some cannot correctly determine several structural relationships between classes. The only approach that allows complete derivation of the business process model mentioned by the authors is presented by Friedrich, Mendling and Puhlmann (Friedrich et al. 2011).

Some approaches use ontologies predefined by experts in the field and self-developed knowledge acquisition rules in order to extract knowledge on necessary properties or elements and their values from text documents (Amardeilh et al. 2005; Jones et al. 2014).

In all cases the source document for creation of different UML diagrams is a specification of software requirements expressed as text in formal style.

3 TOPOLOGICAL FUNCTIONING MODEL

3.1 TFM Functional Characteristics

The TFM is a formal model for representing and analysis of functionality of the system of any kind, e.g., business, software, biological, mechanical, etc. (Osis and Asnina 2011b). The TFM may represent functionality as a directed graph (X, Θ), where X is a closed set of inner functional characteristics (hereinafter called *functional features*) of the system, and Θ is a topology set on them in the form of a set of cause-and-effect relations. TFM models can be compared for similarities and differences using the continuous mapping mechanism of topological spaces (Asnina and Osis 2010). The TFM is characterized by its topological and functioning properties (Osis and Asnina 2011a). The topological properties come from algebraic topology, they are connectedness, neighbourhood, closure and continuous mapping. The functioning properties come from the system theory, they are cause-and-effect relations, cycle structure, inputs and outputs (Osis 1969).

Definition of domain functional characteristics (Asnina and Osis 2011) includes determination of:

- List of domain objects and their properties,
- List of external systems,
- List of subsystems/actors,
- List of functional features.

The main TFM element is a functional feature that represents system's functional characteristic, e.g., a business process, a task, an action, or an activity (Osis and Asnina 2011a). It can be specified by a unique tuple (1). Comparing to the original one (Osis and Asnina 2011b), we have added element *D* for the better understanding of meaning of the feature.

FF = <D, A, **R**, **O**, **PrCond**, **PostCond**, **Pr**, **Ex,** S> (1)

Where:

- D is a description of the functional feature,
- A is object's action,
- **R** is a set of results of the object's action (it is an optional element),
- **O** is an object set which contains domain objects that is used or get the result of the action; for atomic functional feature the size of the set is equal to 1,
- **PrCond** is a set of preconditions or atomic business rules,
- **PostCond** is a set of post-conditions or atomic business rules,
- **Pr** is a set of providers of the feature, i.e. entities (systems or sub-systems) which provide or suggest action A with a set **O** of certain objects,
- **Ex** is a set of executors (direct performers) of the functional feature, i.e. a set of entities (systems or sub-systems) that enact action A.
- S is a variable *Subordination* that holds changeable value of belonging of the functional feature either to the system or to the external environment according to the value of **Pr**. This means, for example, if there are two subsystems of the system, namely, *subsystem1* and *subsystem2*, then in case of separation of the TFM for *subsystem2*, S = *external* for functional feature with **Pr** = {*System1*, *subsystem1*}. Likewise, in case of separation of the TFM for *subsystem1*, the value of S will be equal to *inner*.

A cause-and-effect relation between functional features defines the cause from which the triggering of the effect occurred. The formal definition of the cause-and-effect relations and their combinations is given by Osis, Donins, Asnina and Ovchinnikova (Asnina and Ovchinnikova 2015; Donins 2012; Osis and Donins 2017).

3.2 Natural Language Processing in the IDM Toolset

The starting point of applying NLP of the textual description of system functioning for acquiring knowledge for the TFM is implementation in the IDM (Integrated Domain Modelling) toolset, where processing of use case scenario text is performed using the Stanford Parser Java Library for identifying the executors (**Ex**) and the description of the functional feature *D* that is the *verb phrase* from the text of a step in a use case scenario (Osis and Slihte 2010; Slihte et al. 2011)**.**

The prerequisite for parsing is that sentences of use case steps must be in the simple form to answer the question "Who does what?", e.g., "Librarian checks out the book".

Parsing is done according to these steps:

- Identify *coordinating conjunctions* to split a sentence into several clauses, and, thus, several functional features;
- Identify the *verb phrase* (VP tag) that is considered as a union of action A, object O and

result R (if it is indicated) and forms the so-called *description* of the functional feature;
- Identify the *noun phrase* (NP tag) that is marked as executor Ex if it meets the same noun in the list of actors for the use case;
- Preconditions and postconditions are taken directly from the corresponding preceding step in the use case (if they are specified);
- Topological relations are equal to the sequence of use case steps.

As a result, the following elements of the tuple (1) are obtained: 1) A, **R, O** *implicitly* in the description of functional feature, 2) **Ex** (single element), 3) **PrCond** and **PostCond** if they are specified for the use case, 4) Topology is determined according to the sequence of steps specified in the flow of scenarios. The existing process is limited to use case specification that is manually proceeded and structured text.

3.3 Extracting Elements of the TFM: Formulation of the Research Task

The evolution of the topological functioning modelling leads us to the solution, where knowledge extracted from text must be kept in the knowledge-frame base. Part of knowledge can be generated from the manually entered facts. According to the initial scheme of the knowledge frame system (Nazaruks and Osis 2017), the current research puts the focus on knowledge that is to be entered manually, i.e., values for slots of frame classes Object, Property, and FunctionalFeature (Nazaruka and Osis 2018): Object for domain objects, Property for properties of the domain objects, and FunctionalFeature for the TFM functional features.

In case of unstructured text in formal style (hereinafter, *formal text*) we cannot use the same principles for discovering pre- and postconditions, while others are suitable. Therefore, tokenization, part-of-speech (POS) tagging, chunking, and Name Entity Recognition (NER)/Classification" as well as semantic analysis of noun and verb phrases must be done (Nazaruka and Osis 2018). Besides that, the tagged text and parsed trees must be semantically analysed to identify causal dependencies. In step of NER/Classification noun and verb ontology banks must be used.

Therefore, the existing processing must be improved to proceed formal text and to achieve:
- Clear identification of action A, set of results **R** and a set of domain objects, namely, objects **O** with their properties.
- Identification of the **Pr** and **Ex** directly or from the context, if it is not stated explicitly.
- Identification of **PrCond**, **PostCond** from the text according to the context and logical operators (OR, AND, XOR).
- Initialization of the default value of Subordination as "not defined".

Since, the last two points require discourse analysis in text, it will be omitted in this research. Here, the focus is on the sentence and word level analysis.

4 NATURAL LANGUAGE PROCESSING FOR THE TFM

The Stanford CoreNLP toolkit (Manning et al. 2014) contains components that deal with tokenization, sentence splitting, POS tagging, morphological analysis (identification of base forms), NER, syntactical parsing, coreference resolution and other annotations such as gender and sentiment analysis. The NER component recognizes names (PERSON, LOCATION, ORGANIZATION, MISC – miscellaneous) and numerical (MONEY, NUMBER, DATE, TIME, DURATION, SET) entities. Phrases can be parsed using both constituent and dependency representations based on a probabilistic parser that is more accurate according to the parsers that relate to some predefined structures. Discovering basic dependencies can help in identification of actions and corresponding objects, results, modes (that can serve for identification of causal dependencies), executors and providers. Besides that, the Stanford CoreNLP implements mention detection and pronominal and nominal coreference resolution that can help in dealing with pronouns and noun phrases that denote concrete phenomena.

For the given research we use Stanford CoreNLP version 3.9.2 that for POS tagging uses tags listed in Penn Treebank II (Bies et al. 1995). In this research the following tags are mentioned: S – simple declarative clause, NN – noun, single, NNS – noun, plural, NP – noun phrase, PRP – preposition, VBZ – verb, 3rd person singular present, VBP – verb, non-3rd person singular present, VBD – verb, past tense, VBG – verb, gerund or present participle, VBN – verb, past participle, VB – base form, VP – verb phrase, IN – preposition or subordinating conjunction.

4.1 General Steps of using NLP and Processing Outcomes

Preparational Step "Coreference resolution". In steps of identification executors, objects and results may be cases when a proposition (PRP tag) takes part in the relation. The proposition must be substituted with the corresponding noun (tagged NN or NNS) using results of coreference resolution. A preposition and the corresponding noun are linked using the edge *coref*. For example, in the sentence "When the reader completes the request for a book, he gives it to the librarian" (Figure 4), proposition "he" relates to "the reader" and "it" to "the request for a book".

Step 1. Identification of action A can be done using POS, lemmas, word dependency analysis, and constituency parsing and covers several cases. First, verb phrases VP must be identified in the sentence. At this step, we are interested only in verbs as such, not their modality. Therefore, in the found VP verbs tagged as VBZ, VBP, VBD, VBN, VBG, or VB must be determined. The verb word we need to extract must be linked with a noun (tag NN, NNS, or PRP) by using *nsubj* or *dobj* edges. The value of action A is the infinitive form of the verb that can be found using lemmas analysis, as for example for verb in VBZ "creates" it will be "create" (Figure 2).

If the verb has link *compound:prt* to the particle tagged RP, then it must be extracted together with it, e.g., "check out".

Figure 2: The result of lemmas analysis.

Step 2. Identification of elements of set ***Ex*** can be done using four NLP tasks – POS, NER, word dependency analysis and constituency parsing. An element of **Ex** is such noun phrase NP where a noun (tag NN, NNS, or PRP) is linked with the verb (tag VBZ, VBP, or VBD) by using: a) edge *nsubj* for active voice, or b) edge *nmod:agent* for passive voice. If *basic dependencies* are used, then *nmod:agent* (used in *enhanced ++ dependencies*) is replaced by *nmod* to the noun, and the noun is linked with the preposition "by" (tag IN) using edge *case*. This is illustrated by the results of analysis of two sample sentences: "The authorized librarian creates a new reader account" (Figure 3 and Figure 5) and "The new reader card is created by the authorized librarian" (Figure 6). The value of Ex_i is equal to the whole NP that contains the mentioned noun, in our sentence it is NP "the authorized librarian".

The NER task can be applied to check extracted nouns whether they are tagged as "TITLE". However, here NER tagging works only for NN: NNS are skipped.

```
(ROOT (S
  (NP
    (DT The) (VBN authorized)
    (NN librarian)
  )
  (VP
    (VBZ creates)
    (NP
      (DT a) (JJ new)
      (NN reader) (NN account)
    )
  )
  (. .)
))
```

Figure 3: The result of constituency parsing of the sentence in the active voice (at the sentence and phrase levels).

Step 3. Identification of object O_i and *result* R_i should be done in one step. Object O_i (with or without the compound result R_i) is a direct object of the verb (Asnina and Osis 2011).

Step 3.1. Identification of the direct object of the verb – action A. If a sentence contains a verb (action A) in the active voice, then the VP structure includes sub-structure NP, where the direct object is located, i.e. word n_1 tagged as NN, NNS, or PRP and linked by using edge *dobj*.

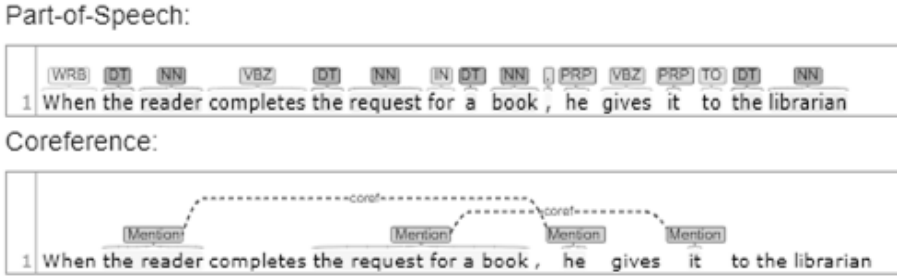

Figure 4: Results of POS identification and coreference resolution.

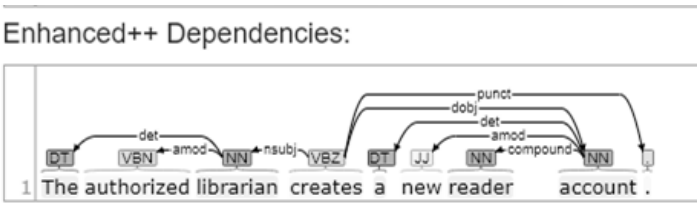

Figure 5: The result of dependency analysis of the sentence in the active voice (the word level).

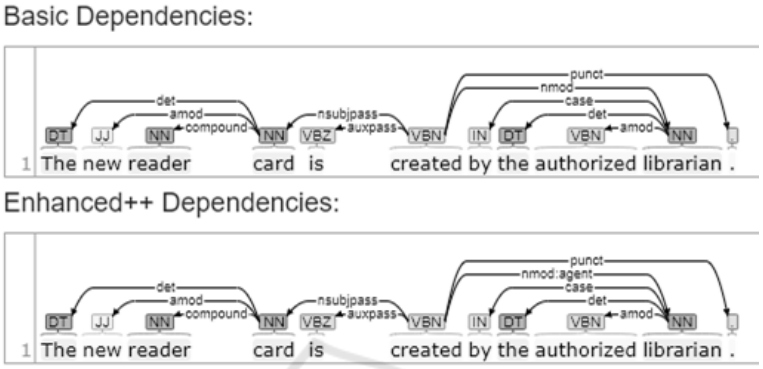

Figure 6: The result of dependency analysis of the sentence in the passive voice (at the word level).

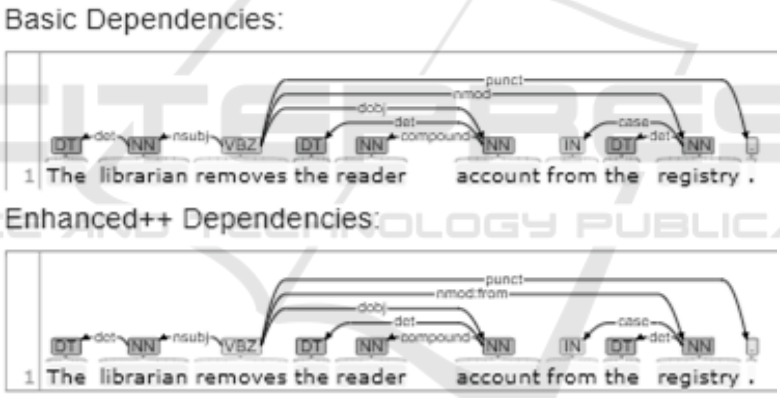

Figure 7: The results of dependency parsing for the sentence with more complex NP.

In case of the passive voice, the VP structure contains sub-structure NP, where the subject is located. Thus, we need to extract word n_1 tagged as NN, NNS, or PRP that is linked with the verb by using edge *nsubjpass*.

Step 3.2. Determination of the object and result of the functional feature.

If the VP of the verb – action A is **not** linked by using any edge *nmod* but *nmod:agent* with another word n_2 tagged as NN, NNS, or PRP, then the following is true:

- If noun n_1 **is not** linked with another noun n_2 in *the same structure NP* by using either edge *compound* or *in the same structure VP* by using one of edges *nmod:poss*, *nmod:of, nmod:to, nmod:into, nmod:from, nmod:for* (in enhanced++ dependencies; otherwise, *nmod* to word n_2 tagged as NN, NNS or PRP and *case* to the preposition "of", "to", "into", "from", or "for" tagged as IN), then the value of O_i is equal to n_1. Otherwise, if such links do exist, the value of O_i is equal to *linked noun* n_2.
- In case if noun n_1 **is** linked with n_2 by using edge *compound*, then leaves of the whole structure NP that contains n_1 are extracted and the preposition "of" is added to the end of the extracted string. The obtained string is the value of element R_i.
- In case if noun n_1 is linked with n_2 by using edge *nmod* (and its variations), then leaves of the whole structure NP that contains n_1 are extracted and the preposition tagged IN linked

with n_2 by using edge *case* is added to the end of the extracted string. The obtained string is the value of element R_i.
- Otherwise, the value of R_i is left empty.

Otherwise, if the VP of the verb – action A *is* linked with another word n_2 (not a direct object or nominal passive subject) tagged as NN, NNS, or PRP using edge *nmod* but *nmod:agent*, too, then the following is true:
- Word n_2 located in the corresponding NP in the prepositional phase PP is a value of the element O_i.
- Leaves of the whole structure NP that has direct child n_1 are extracted. The preposition tagged IN in the sibling prepositional phrase PP is added to the end of the extracted string. The result string is the value of element R_i.

Let us consider the sentence "The librarian removes the reader account from the registry." The verb *removes* is linked with the noun *account* (tag NN) by using edge *dobj* (Figure 7). Leaves of the corresponding NP are extracted as string "the reader account" and supplemented with the preposition "from" (Figure 8). The final string "the reader account from" is written as a value of R_1. The noun *registry* (tag NN) is recorded as a value of O_1.

```
(ROOT (S
  (NP (DT The) (NN librarian))
  (VP
    (VBZ removes)
    (NP
    ((NP
        (DT the)
        (NN reader)
        (NN account)
      )
      (PP
        (IN from)
        (NP (DT the) (NN registry))
      )
     )
    )
   )
    (. .)
))
```

Figure 8: The results of the constituency parsing for the sentence with more complex NP.

In case of conjunctions of NPs, e.g. "creates an account and a card", the head noun or proposition will be linked with the verb by using edge *dobj*, while other nouns or propositions will be linked with the head noun by using edge *conj*. All the linked words must be found and processed according to the abovementioned principles.

Let us consider the sentence in the active voice: "The authorized librarian creates a new reader account". The VP (Figure 3) contains the verb *creates* (tagged VBZ) that is not linked to any noun or proposition by using *nmod* (Figure 5).

The VP contains structure NP, where the direct object (edge *dobj* in Figure 5) is the noun *account*. Let us denote it as n_1. Within the same NP, n_1 = *"account"* is linked to noun n_2 = "*reader*" by the edge *compound*. So, $O_1 = n_2$ = *"reader"*. The NP that contains n_1 = *"account"* is "a new reader account". As n_1 is linked with n_2 by using edge *compound*, then, after adding the proposition "of", R_1 = "*a new reader account of*".

In case of the passive voice, "The new reader card is created by the authorized librarian" (Figure 6), edge *nsubjpass* links the verb *created* with noun *n1* = "*card*". The verb *created* is linked only with NP that contains the agent *librarian* (Figure 9). Thus, following the rules, $O_1 = n_2$ = *"reader"*, and R_1 = *"the new reader card of"*.

```
(ROOT (S
  (NP (DT The) (JJ new) (NN reader) (NN card))
  (VP
    (VBZ is)
    (VP (VBN created)
      (PP
        (IN by)
        (NP
          (DT the)
          (VBN authorized)
          (NN librarian)
        )
      )
    )
  )
  (. .)
))
```

Figure 9: The result of constituency parsing of the sentence with the verb in the passive voice.

Step 4. Identification of description D. The description is a visible part of the functional feature that is needed for its unique identification by a human. The original form is as in expression (2).

action A-ing [[the] *result R*] [prepos.] [a] *object O* (2)

For simplicity form the final form of D we have excluded the ending "ing" and articles (3).

<action A> [<result R_i>]<object O_i> (3)

If one of the elements is empty, then it is replaced by the question mark "?".

4.2 Example and Discussion

The analysed description is the following: "When an unregistered person arrives, the librarian creates a new reader account and a reader card. The librarian gives out the card to the reader. When the reader completes the request for a book, he gives it to the librarian. The librarian checks out the requested book from a book fund to a reader, if the book copy is available in a book fund. When the reader returns the book copy, the librarian takes it back and returns the book to the book fund. He imposes the fine, if the term of the loan is exceeded, the book is lost, or is damaged. When the reader pays the fine, the librarian closes the fine. If the book copy is hardly damaged, the librarian completes the statement of utilization, and sends the book copy to the recycling organization." (Osis et al. 2007).

Going through the steps, from eight full sentences we have obtained 19 functional features (Table 1).

Functional feature 1 lacks a direct object. The 6th and 7th sentences have no results (Table 1, features 11, 13-16).

Functional features 12-14 and 17 have undefined executors (Table 1), they describe some events that happened beyond the system.

Looking at functional features 14 and 17 (Table 1), one can found that the 17th is a refinement of the 14th. Indeed, if we look closer to the initial text, the text "If the book copy is hardly damaged…" concretizes the statement "…if the book…is damaged". So, we may say, that this is one and the same "action" happened outside the system.

Table 2 shows comparison of the 19 functional features with 22 features got after manual text processing.

First, executors are correctly defined for all extracted features.

Second, identification of verbs phrases allowed extracting "outside actions" from adverbial and conditional clauses (features 12-14, 17 on the left side), while in manual processing the "outside actions" have been transformed into "inner actions" that check results of those "outside actions" (features 14, 15 on the right side). Besides that, the obtained feature list is supplemented with implicit "actions" (features 10, 11, 22 on the right side).

Third, identified objects differ, too. For NLP processed text they are *reader (*properties: *reader*

Table 1: Elements of the functional features extracted from text.

Id	Description D	Action A	Result R	Object O	Executors **Ex**
1	Arrive <?> <?>	arrive	?	?	an unregistered person
2	Create a new reader account of reader	create	a new reader account of	reader	the librarian
3	Create a reader card of reader	create	a reader card	reader	the librarian
4	Give out the card to reader	give out	the card to	reader	the librarian
5	Complete the request for book	complete	the request for	book	the reader
6	Give the request for book	give	the request for	book	the reader
7	Check out the requested book from a book fund	check out	the requested book from	book fund	the librarian
8	Return the book copy of book	return	the book copy of	book	the reader
9	Take back the book copy of book	take back	the book copy of	book	the librarian
10	Return the book to book fund	return	the book to	book fund	the librarian
11	Impose <?> fine	impose	?	fine	the librarian
12	Exceed the term of loan	exceed	the term of	loan	?
13	Lose <?> book	lose	?	book	?
14	*Damage <?> book*	*damage*	*?*	*book*	*?*
15	Pay <?> fine	pay	?	fine	the reader
16	Close <?> fine	close	?	fine	the librarian
17	*Damage the book copy of book*	*damage*	*the book copy of*	*book*	*?*
18	Complete the statement of utilization	complete	the statement of	utilization	the librarian
19	Send the book copy to recycling organization	send	the book copy to	recycling organization	the librarian

Table 2: Functional features extracted using NLP outcomes and manual processing.

Functional features extraction (using NLP)			Functional features extraction (manual processing)		
Id	Description *D = <A> <R> <O>*	Executor Ex	Id	Description *<A>-ing [the <R>] [<PRP>] [a] <O>*	Executor Ex
1	Arrive <?> <?>	an unregistered person	1	Arriving [of] a person	person
2	Create a new reader account of reader	the librarian	2	Creating a reader account	Librarian
3	Create a reader card of reader	the librarian	3	Creating a reader card	Librarian
4	Give out the card to reader	the librarian	4	Giving out the card to a reader	Librarian
			5	Confirming the status of a reader	Reader
5	Complete the request for book	the reader	6	Completing a request_for_book	Reader
6	Give the request for book	the reader	7	Sending a request_for_book	Reader
			8	Taking out the book copy from a book fund	Librarian
7	Check out the requested book from a book fund	the librarian	9	Checking out a book copy	Librarian
			10	Giving out a book copy	Librarian
			11	Getting a book copy [by a registered reader]	Reader
8	Return the book copy of book	the reader	12	Returning a book copy [by a registered person]	Reader
9	Take back the book copy of book	the librarian	13	Taking back a book copy	Librarian
10	Return the book to book fund	the librarian	17	Returning the book copy to a book fund	Librarian
11	Impose <?> fine	the librarian	16	Imposing a fine	Librarian
12	Exceed the term of loan	?			
			14	Checking the term of loan of a book copy	Librarian
13	Lose <?> book	?			
14	Damage <?> book	?			
			15	Evaluating the condition of a book copy	Librarian
15	Pay <?> fine	the reader	18	Paying a fine	Reader
16	Close <?> fine	the librarian	19	Closing a fine	Librarian
17	Damage the book copy of book	?			
18	Complete the statement of utilization	the librarian	20	Completing a statement_of_utilization	Librarian
19	Send the book copy to recycling organization	the librarian	21	Sending the book copy to a recycling organization	Librarian
			22	Recycling a book copy	Recycling organization

*account, reader card / card), book (*properties: *request, book copy), book fund* (properties: *book*), *fine, loan (*properties: *term), utilization (*properties: *statement), recycling organization (*properties: *book copy*), while in the manual approach they are *person, reader account, reader card, reader (*properties: *card, status*), *request_for_book, book fund (*properties: *book copy*), *book copy (*properties: *term_of_loan, condition*), *fine, statement_of_-utilization, recycling organization (*properties: *book copy*). During manual processing, the expert has used his knowledge to abstract and unify several concepts.

4.3 Parsing Issues

The result of parsing and POS tagging may be affected by errors in lexical analysis. There are several parser models used by CoreNLP. Sometimes they can provide outputs with incorrect lexical analysis. So, analysing the clause "The librarian checks out the requested book from a book fund to a reader…" an issue was found that depends on the parser used. Until v.3.6.0, the default parser was englishPCFG.ser.gz (Stanford 2018).

```
(ROOT (S
 (NP (DT The) (NN librarian))
 (VP (VBZ checks) (PRT (RP out))
 (NP
  (NP
   (DT the)
   (JJ requested)
   (NN book)
  )
  (PP (IN from)
      (NP
       (NP DT a) (NN book) (NN fund))
       (PP (TO to)
        (NP (DT a) (NN reader))
        )))
 ) (, ,)
```

Figure 10: The result of parsing the sentence by using englishPCFG.ser.gz by CoreNLP GUI.

Using this parser alone in CoreNLP GUI, the POS stage was performed correctly (Figure 10) – the word "checks" was recognized as VBZ (verb, 3rd person singular present), while using the newer one English model in CoreNLP from the command line as well as in online web application *coreNLP.run* it was mistakenly recognizes as a plural noun – NNS (Figure 11). The reason is that the form of the verb is identical to the form of the plural noun, and the result depends on the language model used by the parser.

This means, that some actions can be not defined and not included into the TFM further, at the same time some additional domain objects O will be defined.

It is recommended to exclude situations when an adjective or a cardinal number describes the noun. For example, it is better to write rather "Substruct the first digit from the second digit" than "Substruct the first digit from the second" or "… from the second one". Otherwise, the processing will show incorrect results.

```
<parse>(ROOT
  (S
  (NP (DT The) (NN librarian) (NNS checks))
  (ADVP (IN out) (DT the))
  (VP (VBN requested)
  (NP (NN book))
  (PP (IN from)
  (NP (DT a) (NN book) (NN fund)))
  (PP (TO to)
  (NP (DT a) (NN reader)))
  (, ,)
  (SBAR (IN if)
  (S
  (NP (DT the) (NN book) (NN copy))
  (VP (VBZ is)
  (ADJP (JJ available)
  (PP (IN in)
  (NP (DT a) (NN book) (NN fund))))))))
  (. .)))
```

Figure 11: The result of parsing the sentence by using a newer English POS model by CoreNLP Command Line.

Stanford CoreNLP allows open information extraction (open IE) from text, i.e., the extraction of relation tuples, typically binary relations (Angeli et al. 2015), such as (nominal-subject; verb-relation; nominal-object).

At first sight, the format of extraction seems very suitable for our purpose. However, it has several disadvantages (Figure 12). First, it does not extract the tuple if one of the elements is absent, as in case of "…an unregistered person arrives…". Second, it extracts tuples, where nominal-object can be an adjective, as in case of "The book copy is blue". Thus, using open IE some tuples may be missed, and some unnecessary tuples may be extracted. Therefore, we need analysis in more detail.

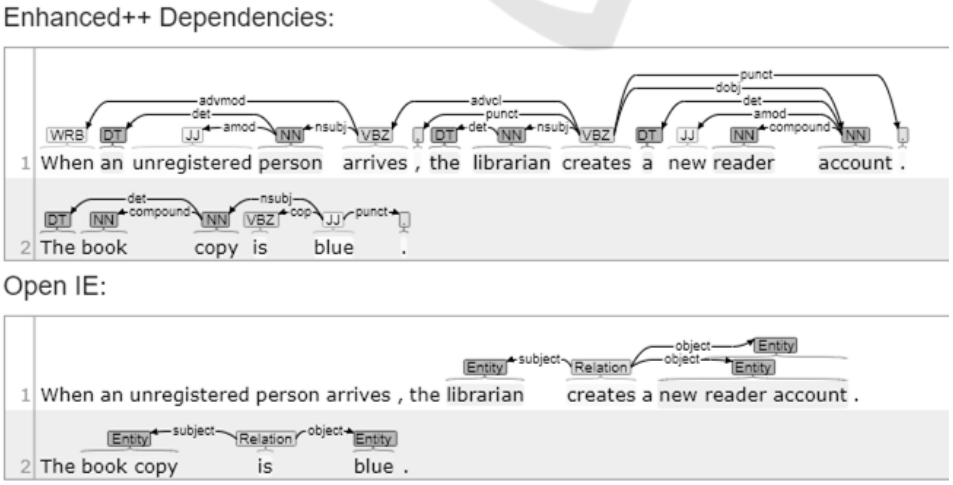

Figure 12: Results of application of the Open IE that limit its usage for TFM element extraction.

5 CONCLUSIONS

Summarizing all the results, we can conclude the following:

- In order to exclude errors in NLP outcomes caused by CoreNLP parser, the source text must avoid verb forms syntactically equals to noun forms.
- Results of manual processing can be more complete, because of expert's ability to add implicit knowledge, but differ from the written text due to expert's ability to modify actions and events ad hoc as well as to find synonyms in the text.
- Structural relations between extracted domain objects differ in two approaches. In case of NLP it depends on completeness of data in the sentence. In case of manual processing it depends on expert's knowledge about the domain, thus, on implicit knowledge.
- Incomplete knowledge can be extracted, because sentences lack information on who/what is the subject when verbs are in the active voice; however, if after processing this knowledge is absent, then text can be supplemented with necessary information.
- Incomplete information may lead to identification of more abstract or more specific functional features for one and the same functional characteristic.
- A sentence can contain information on both executor and recipient. The recipient may be another actor. Now, this actor is specified as an object of action A. However, there are cases when it is worth to specify them separately.

Therefore, processing of textual descriptions even in formal style have issues related to the technical side (i.e., parsing models and outcome representation formats) and to particularities of the natural language (textual description may not have all needed knowledge, structures of sentences differ, implicit synonyms are used). The latter can be partially solved either by using machine learning or manual pre-processing of knowledge, e.g., specification of use case scenarios or user stories and exhaustive software requirements.

Future research expects refinement of the proposed steps to decrease ambiguity in the processing results: to find patterns of sentences that minimize arbitrary interpretations of structural relations between domain objects, to elaborate more specific separation of domain objects involved in actions, as well as identify cause-and-effect relations between functional characteristics of the system.

REFERENCES

Amardeilh, F., Laublet, P. and Minel, J.-L., 2005. Document annotation and ontology population from linguistic extractions. In *Proceedings of the 3rd international conference on Knowledge capture - K-CAP '05*. New York, New York, USA: ACM Press, pp. 161–168.

Angeli, G., Johnson Premkumar, M. and Manning, C.D., 2015. Leveraging Linguistic Structure For Open Domain Information Extraction. In *Proceedings of the Association of Computational Linguistics (ACL)*. Beijing, China: Association for Computational Linguistics, pp. 344–354.

Asnina, E. and Osis, J., 2010. Computation Independent Models: Bridging Problem and Solution Domains. In *Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development*. Lisbon: SciTePress - Science and and Technology Publications, pp. 23–32.

Asnina, E. and Osis, J., 2011. Topological Functioning Model as a CIM-Business Model. In *Model-Driven Domain Analysis and Software Development*. Hershey, PA: IGI Global, pp. 40–64.

Asnina, E. and Ovchinnikova, V., 2015. Specification of decision-making and control flow branching in Topological Functioning Models of systems. In *ENASE 2015 - Proceedings of the 10th International Conference on Evaluation of Novel Approaches to Software Engineering*.

Bhala, V., Vidya Sagar, R. and Abirami, S., 2014. Conceptual modeling of natural language functional requirements. *The Journal of Systems and Software*, 88, pp.25–41.

Bies, A. et al., 1995. *Bracketing Guidelines for Treebank II Style*, Available at: https://repository.upenn.edu/cis_reports/index.4.html.

Donins, U., 2012. Semantics of Logical Relations in Topological Functioning Model. In *Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering, Wrocław, Poland, 29-30 June, 2012*. SciTePress, pp. 217–223.

Elbendak, M., Vickers, P. and Rossiter, N., 2011. Parsed use case descriptions as a basis for object-oriented class model generation. *Journal of Systems and Software*, 84(7), pp.1209–1223.

Elstermann, M. and Heuser, T., 2016. Automatic Tool Support Possibilities for the Text-Based S-BPM Process Modelling Methodology. In *Proceedings of the 8th International Conference on Subject-oriented Business Process Management - S-BPM '16*. New York, New York, USA: ACM Press, pp. 1–8.

Friedrich, F., Mendling, J. and Puhlmann, F., 2011. Process Model Generation from Natural Language Text. In *Proceedings of the 23rd International Conference on*

Advanced Information Systems Engineering (CAiSE 2011). pp. 482–496.

Ilieva, M.G. and Ormandjieva, O., 2006. Models Derived from Automatically Analyzed Textual User Requirements. In *Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06)*. IEEE, pp. 13–21.

Jabbarin, S. and Arman, N., 2014. Constructing use case models from Arabic user requirements in a semi-automated approach. In *2014 World Congress on Computer Applications and Information Systems, WCCAIS 2014*. Hammamet: IEEE, pp. 1–4.

Jones, D.E. et al., 2014. Automatic Extraction of Nanoparticle Properties Using Natural Language Processing: NanoSifter an Application to Acquire PAMAM Dendrimer Properties V. Ceña, ed. *PLoS ONE*, 9(1), p.e83932.

Krishnan, H. and Samuel, P., 2010. Relative Extraction Methodology for class diagram generation using dependency graph. In *2010 INTERNATIONAL CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING TECHNOLOGIES*. IEEE, pp. 815–820.

Manning, C.D. et al., 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. pp. 55–60.

Miller, J. and Mukerji, J., 2001. *Model Driven Architecture (MDA)*, Available at: http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01.

Nassar, I.N. and Khamayseh, F.T., 2015. Constructing Activity Diagrams from Arabic User Requirements using Natural Language Processing Tool. In *2015 6th International Conference on Information and Communication Systems (ICICS)*. Amman: IEEE, pp. 50–54.

Nazaruka, E. and Osis, J., 2018. Determination of Natural Language Processing Tasks and Tools for Topological Functioning Modelling. In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering*. Funchal, Madeira, Portugal: SCITEPRESS – Science and Technology Publications, Lda., pp. 501–512.

Nazaruks, V. and Osis, J., 2017. Joint Usage of Frames and the Topological Functioning Model for Domain Knowledge Presentation and Analysis. In *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: MDI4SE*. Porto, Portugal: SCITEPRESS - Science and Technology Publications, pp. 379–390.

Nazaruks, V. and Osis, J., 2018. Verification of Causality in the Frame System based on the Topological Functioning Modelling. In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, Portugal, Funchal, Madeira, 23-24 March, 2018*. Portugal: SCITEPRESS – Science and Technology Publications, Lda., pp. 513–521.

Osis, J., 1969. Topological Model of System Functioning (in Russian). *Automatics and Computer Science, J. of Academia of Siences*, (6), pp.44–50.

Osis, J. and Asnina, E., 2011a. Is Modeling a Treatment for the Weakness of Software Engineering? In *Model-Driven Domain Analysis and Software Development*. Hershey, PA: IGI Global, pp. 1–14.

Osis, J. and Asnina, E., 2011b. Topological Modeling for Model-Driven Domain Analysis and Software Development : Functions and Architectures. In *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. Hershey, PA: IGI Global, pp. 15–39.

Osis, J., Asnina, E. and Grave, A., 2007. Computation Independent Modeling within the MDA. In *IEEE International Conference on Software-Science, Technology and Engineering (SwSTE'07)*. Herzlia: IEEE, pp. 22–34.

Osis, J. and Donins, U., 2017. *Topological UML modeling : an improved approach for domain modeling and software development*, Elsevier.

Osis, J. and Slihte, A., 2010. Transforming Textual Use Cases to a Computation Independent Model. In J. Osis and O. Nikiforova, eds. *Model-Driven Architecture and Modeling-Driven Software Development: ENASE 2010, 2ndMDAandMTDD Whs.* SciTePress, pp. 33–42.

Osman, C.-C. and Zalhan, P.-G., 2016. From Natural Language Text to Visual Models: A survey of Issues and Approaches. *Informatica Economica*, 20(4), pp.44–61.

Slihte, A., Osis, J. and Donins, U., 2011. Knowledge Integration for Domain Modeling. In J. Osis and O. Nikiforova, eds. *Model-Driven Architecture and Modeling-Driven Software Development: ENASE 2011, 3rd Whs. MDAandMDSD*. SciTePress, pp. 46–56.

Stanford, 2018. CoreNLP version 3.9.2. *Understanding Memory and Time Usage*. Available at: https://stanfordnlp.github.io/CoreNLP/memory-time.html.