Determination of Natural Language Processing Tasks and Tools for Topological Functioning Modelling

Erika Nazaruka and Jānis Osis
Department of Applied Computer Science, Riga Technical University, Sētas iela 1, LV-1048, Riga, Latvia

Keywords: Natural Language Processing Tools, Topological Functioning Model, Computation Independent Model, Domain Modelling.

Abstract: Topological Functioning Modelling (TFM) is based on analysis of exhaustive verbal descriptions of the domain functionality. Manual acquisition of knowledge about the domain from text in natural language requires a lot of resources. Natural Language Processing (NLP) tools provide automatic analysis of text in natural language and may fasten and make cheaper this process. First, the knowledge, its expressing elements of the English language, and processing tasks that are required for construction of the topological functioning model are identified. The overview of the support of these tasks by the main NLP pipelines is based on the available documentation without performing practical experiments. The results showed that among the selected six NLP pipelines the largest support comes from the Stanford CoreNLP toolkit, FreeLing, and NLTK toolkit. They allow analysing not only the words and sentences, but also dependencies in word groups and between sentences. The obtained results can be used for academics and practitioners that perform research on NLP for composition of domain (business, system, software) models.

1 INTRODUCTION

Model Driven Architecture (MDA) (Miller and Mukerji, 2001) proposed by OMG (Object Management Group) gave a ground for new software development principles, where models of the software are at the core of the development process. MDA suggests using three models: a computation independent model (CIM), a platform independent model (PIM) and a platform specific model (PSM). Commonly, a language for MDA models is the UML (Unified Modelling Language), sometimes BPMN (Business Process Model and Notation), and rare SBVR (Semantic Business Vocabulary and Rules). BPMN and SBVR are used for specification of the CIM, while UML for the PIM and PSM.

In our approach, we suggest using a Topological Functioning Model (TFM) as the CIM. The TFM elaborated by Janis Osis at Riga Technical University, Latvia, in 1969, specifies a system from three viewpoints – functional, behavioural and structural. This model can serve as a root model for further system and software domain analysis and transformations to design models and code (Osis and Asnina, 2011b).

There are two approaches for composition of the TFM, namely, TFM4MDA (Topological Functioning Model for Model Driven Architecture) and IDM (Integrated Domain Modelling) presented in (Slihte, Osis and Donins, 2011). Rules of composition and derivation processes from the textual system description within TFM4MDA are provided by examples and described in detail in (Asnina, 2006; Osis, Asnina and Grave, 2007, 2008). Since, TFM4MDA does not have software tool support, results of text processing are kept in tables. Additionally, the TFM can be manually created in the TFM Editor or can also be generated automatically from the business use case descriptions in the IDM toolset (Osis and Slihte, 2010; Šlihte and Osis, 2014). So, TFM4MDA proposes manual processing of the unstructured, but processed text, while IDM – automated processing of use case specifications in the form of semi-structured text. In this case, results of text processing are kept in XMI (XML Metadata Interchange) files using XML (eXtensible Markup Language) structures.

At the present, we decided to use a knowledge base for keeping results of text processing to gain from its inferring mechanism and flexibility. The knowledge frame based approach (Nazaruks and Osis, 2017) is at its very beginning. It assumes that

Nazaruka, E. and Osis, J.
Determination of Natural Language Processing Tasks and Tools for Topological Functioning Modelling.
DOI: 10.5220/0006817205010512
In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*, pages 501-512
ISBN: 978-989-758-300-1

knowledge on domain will be kept in the knowledge frame system. In practice, preparation of the text and manual knowledge acquisition from it is too resource-consuming (Elstermann and Heuser, 2016). Therefore, it is better either to skip the step of preparation of the textual description and start from human analysis of the available information, either to automate or semi-automate this process. We plan to automate the process of knowledge extraction from textual descriptions partially or completely depending on technologies available at the present.

The goal of this research is to understand what possibilities Natural Language Processing (NLP) tools have at the present that could support the automated knowledge acquisition for construction of the TFM from data kept in the knowledge frame system.

The paper is organized as follows. Section 2 presents overview of related work in the field. Section 3 describes the main elements of the TFM, the process of its manual composition and characteristics of its validity. Section 4 presents tasks that are to be supported or assisted by the NLP tools and the overview of the selected NLP tools. Section 5 concludes the paper.

2 RELATED WORK

Knowledge extraction from different types of media is quite important since it may reduce time for analysis of large amount of information. Very interesting approach is presented in (Nakamura *et al.*, 1996), where authors suggest using knowledge extraction from diagrams and its integration with patterns of textual explanations. Nevertheless, the idea has been proposed by Nakamura *et al.* in 1996, it is useful enough also in nowadays (Leopold, Mendling and Polyvyanyy, 2014), since automated creation of explanations for large diagrams would be very helpful in business and system analysis. It can be said that several ways of evolution of this idea relate to knowledge extraction from factual data, diagrams, data warehouses and to data mining (Cannataro, Guzzo and Pugliese, 2002).

Creation of models and UML diagrams from textual documents is presented in several researches. For example, use case diagram creation from textual requirements in Arabic using Stanford Tagger/Parser (Jabbarin and Arman, 2014), and creation of UML Activity Diagram via identification of simple verbal sentences from user requirements in Arabic make a use of Stanford and MADA+TOKAN tagger (Nassar and Khamayseh, 2015). Two research projects suggest creating UML class diagrams from textual requirements in English using the proposed Relative Extraction Methodology (Krishnan and Samuel, 2010), and from use case descriptions (Elbendak, Vickers and Rossiter, 2011). But they do not deal with possible ambiguities of the natural language (NL). 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) mention 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 apply 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 create a semantic network for text model. At the last step, the authors transform this semantic network to one of the mentioned models using patterns. NL analysis can be used for automated composition of conceptual diagrams (Bhala, Vidya Sagar and Abirami, 2014). The authors also noted a need for human participation, as well as several issues of NL 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 model 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 tools used are Stanford Parser and lexical database WordNet 2.1, FrameNet and VerbNet, and NLP libraries that belong to NLTk framework. The only approach that allows complete derivation of the business process model mentioned by the authors is presented by Friedrich, Mendling and Puhlmann in (Friedrich, Mendling and Puhlmann, 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, Laublet and Minel, 2005; Jones *et al.*, 2014).

3 TOPOLOGICAL FUNCTIONING MODELLING

3.1 Topological Functioning Model

The TFM is a formal mathematical model that allows modelling and analysing functionality of the system (Osis and Asnina, 2011b). The system could be a business, software, biological system, mechanical system, etc. The TFM represents modelled functionality as a digraph (X, Θ), where X is a set of inner functional characteristics (called functional features) of the system, and Θ is a topology set on these characteristics in a form of a set of cause-and-effect relations. TFM models can be compared for similarities using the continuous mapping mechanism (Asnina and Osis, 2010). Since 1990s the TFM is being elaborated for software development.

The TFM is characterized by the topological and functioning properties (Osis and Asnina, 2011a). The topological properties are connectedness, neighbourhood, closure and continuous mapping. The functioning properties are cause-and-effect relations, cycle structure, inputs and outputs. The composition of the TFM is presented by Osis and Asnina (2011b).

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).

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

Where (Osis and Asnina, 2011b):

- 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 that gets the result of the action or a set of objects that are used in this action,
- **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 an action with a set 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) which enact a concrete action.
- S is a variable *Subordination* that holds Boolean value of belonging of the functional feature either to the system or to the external environment.

The cause-and-effect relations between functional features define the cause from which the triggering of the effect occurred. The formal definition of the cause-and-effect relations and their combinations (Donins, 2012a; Asnina and Ovchinnikova, 2015) states that a cause-and-effect relation T_{id} is a binary relationship that links a cause functional feature to an effect functional feature as it is stated in (2).

$$T_{id} = <Id, X_c, X_e, \mathbf{L_{out}}, \mathbf{L_{in}}> \quad (2)$$

Where (Donins, 2012a):

- Id is a unique identifier of the cause-and-effect relation;
- X_c is a cause functional feature that may generate the effect functional feature X_e after termination;
- X_e is an effect functional feature;
- $\mathbf{L_{out}}$ is a set of logical relationships between cause-and-effect relations on outgoing arcs of the cause functional feature X_c (optional);
- $\mathbf{L_{in}}$ is a set of logical relationships between cause-and-effect relations on incoming arcs of the effect functional feature X_e (optional).

In fact, this relation indicates control flow transition in the system. The cause-and-effect relations (and their combinations) may use logical *negation (NOT)* and may be joined by the logical operators: *conjunction (AND), disjunction (OR),* or *exclusive disjunction (XOR)*. The logic of the combination of cause-and-effect relations denotes system's behaviour and execution (e.g., decision making, parallel or sequential actions). The formal definition of a logical relationship (3) states that it is put on set **T** of cause-and-effect relations belonging to this logical relationship and Rt specifies the type of it by a logical operator *NOT*, *AND*, *OR*, or *XOR* (Donins, 2012a).

$$L_{id} = <Id, \mathbf{T}, Rt> \quad (3)$$

3.2 Construction Guidelines

Manual construction of the TFM consists of the following steps (Asnina and Osis, 2011):

- Definition of domain functional characteristics:
  - List of domain objects and their properties;
  - List of external systems;

- List of subsystems/actors;
- List of functional features (with the structure equal to (1)).

- Introduction of topology:
  - List of cause-and-effect relations (with the structure equal to (2) and (3));
- Separation of the TFM of the domain:
  - Topological functioning model that must satisfy all topological and functioning properties.

The following running example will be used to illustrate some key points of the discussion. Assume that the following text in the formal style that describes the desired functionality of some book returning machine in the library is presented:

"When the client **appears**, he **can return** the book to the book returning machine. The client **puts** the book into the special tray and **enters** his data of the registration. Then, the machine **checks** the registration and **scans** the image of a book. The machine **searches for** the loaned book in a reader account. *If it is found*, the machine **checks** the end date of a book loan and **evaluates** the condition of a book. *If the end date is not exceeded* and *the condition is good*, then the machine **sends** the book to the storage. Otherwise, *if the end date is exceeded or the condition is not good*, the machine **calculates**, **writes out** and **imposes** the fine to the client. And only then the machine **sends** the book to the book storage. The client **receives** the imposed fine. After sending the book to the storage, the machine **waits for** a new request."

In the text, *domain objects and their properties* are expressed as a noun together with its direct object that is not expressed as a numeral or a pronoun. For example, they can be things, phenomena, products, results of actions, documents, catalogues, human role, services, organizations etc. For each noun, synonyms and homonyms must be analysed as well as their correct sense in each case of a use, e.g., a "train" in one case is a locomotive together with a certain number of carriages, while in other case it is a single locomotive (Asnina and Osis, 2011).

It must be able to determine which of domain objects are *external systems* or *subsystems/actors*. It is necessary to analyse the meaning of nouns, i.e. do they indicate roles, work positions, organizational units such as departments and business centres, organizations, and names of subsystems. A company that uses by-products of functionality of the organization can also be mentioned as an example. "Those objects, which are not subsystems of the system under consideration and whose functionality is not directly subordinated to the system, are external systems" (Asnina and Osis, 2011).

Nouns in the running example (the underlined words) as well as their meaning and purpose in the domain are illustrated in Table 1.

Table 1: Nouns and their purpose in the domain.

Noun	Purpose
Client	Object, actor (executor)
Book	Object
Machine	System, actor (provider, executor; (noun phrase)
Tray	Object, Element of the machine
Data	Properties of the registration (must be refined; noun phrase)
Registration	Object (result of the process)
Image	Property of the book (noun phrase)
Reader	Object (noun phrase)
Account	Object, Property of the reader (noun phrase)
End date	Property of the loan (noun phrase)
Loan	Object (result of the process)
Condition	Property of the Book (noun phrase)
Storage	Object, Element of the machine
Fine	Object
Request	Object

Discovering of *functional features* relates to determination of "a business function" (Asnina and Osis, 2011). A list of functional features must be defined accordingly to the verbs (actions A), their preconditions (**PreCond**) and postconditions (**PostCond**). Preconditions are a set of conditions that allows triggering the action. Postconditions are a set of conditions that are set after a functional feature was executed. A business rule can be either a condition or a functional feature.

A list of domain object identified in the previous step contains objects that are used in the context of this action (O), objects that are the results of this action (**R)**, objects that are marked as external systems or subsystems/actors that either provide (**Pr)** or execute (**Ex)** this concrete action. Together the action, results and the object with preposition form a description (name) of the functional feature.

In the running example, the verbs (in the active voice) are denoted by bold. Adding "-ing" to them and joining objects we can form the part of the specification of functional features (Table 2). Conditions highlighted in text in *italic* are enumerated in Table 3.

Identification of the list of *cause-and-effect* relations among functional features is based on the fact that "a cause-and-effect relation between two

Table 2: Actions, objects and results.

Id	Action	Result	Object	Providers	Executors	*Verb*
1	Appear*ing*	[of]	[a] Client		Client	*appear*
2	Return*ing*		[a] Book	Machine	Client	*return*
3.1	Put*ting*		[a] Book	Machine, Tray	Client	*put*
3.2	Enter*ing*	[the] data [of]	[a] Registration	Machine	Client	*enter*
4	Check*ing*	[the] Registration [of]	[a] Client	Machine	Machine	*check*
5	Scan*ning*	[the] image [of]	[a] Book	Machine	Machine	*scan*
6	Search*ing* for	[the] Book [in]	[a] Reader Account	Machine	Machine	*search*
7	Check*ing*	[the] end date [of]	[a] Loan	Machine	Machine	*check*
8	Evaluat*ing*	[the] condition [of]	[a] Book	Machine	Machine	*evaluate*
9	Calculat*ing*	[the] Fine [to]	[a] Client	Machine	Machine	*calculate*
10	Writ*ing* out	[the] Fine [to]	[a] Client	Machine	Machine	*write out*
11	Impos*ing*	[the] Fine [to]	[a] Client	Machine	Machine	*impose*
12	Send*ing*	[the] Book [to]	[a] Storage	Machine	Machine	*send*
13	Receiv*ing*		[a] Fine		Client	*receive*
14	Wait*ing* for		[a] Request	Machine	Machine	*wait for*

Table 3: Conditions.

Condition	Postcondition of	Precondition for	Explanation
If the book is found	6	7	IF part
If the end date is not exceeded AND *the condition is good*	7, 8	12	IF part
If the end date is exceeded OR the condition is not good	7, 8	9	IF part (that begins the chain of 9, 10, 11)

Table 4: Cause-and-effect relations.

Id	Cause	Effect	Explanation
1-2	1	2	Chronological sequence indicated in the description “When <clause1>, <clause 2>”
2-3.1	2	3.1	Chronological sequence implicitly indicated in the description.
2-3.2	2	3.2	Chronological sequence implicitly indicated in the description.
3.1-4	3.1	4	Chronological sequence indicated in the description “<Sentence1>. Then <sentence 2>”
3.2-4	3.2	4	Chronological sequence indicated in the description “<Sentence1>. Then <sentence 2>”
4-5	4	5	Chronological sequence indicated in the description “<verb phrase 1> and <verb phrase 2>”, where “and” meaning is “if <action 1> is successful, then <action 2>”
5-6	5	6	Chronological sequence implicitly indicated in the description.
6-7	6	7	Chronological sequence indicated in the text as precondition: “If <condition>, <clause>”
6-8	6	8	Chronological sequence indicated in the description as precondition: “If <condition>, “<verb phrase 1> and <verb phrase 2>”
7-9	7	9	Chronological sequence indicated in the text with the precondition: “If <condition>, <clause>”
8-9	8	9	Chronological sequence indicated in the text with the precondition: “If <condition>, <clause>”
9-10	9	10	Chronological sequence indicated in the description as a sequence of actions: “<verb phrase 1>, <verb phrase 2>, and <verb phrase 3>”
9-12	9	12	Chronological sequence indicated in the text with the precondition: “If <condition>, <clause>”
10-11	10	11	Chronological sequence indicated in the description as a sequence of actions: “<verb phrase 1>, <verb phrase 2>, and <verb phrase 3>
11-12	11	12	Chronological sequence indicated in the text with the phrase: “And only then <clause>”
11-13	11	13	Chronological sequence implicitly indicated in the description.
12-14	12	14	Chronological sequence indicated in the description “After <clause1>, <clause 2>”
14-4	14	4	Chronological sequence implicitly indicated in the description.

functional features of the system exists if the appearance of one feature is caused by the appearance of the other feature without participation of any third (intermediary) feature" (Asnina and Osis, 2011). The connection between a cause-and-effect is represented as causal implication, where a logical sequence can serve as a form of expression of it. A cause chronologically *precedes* and *triggers* an effect. Most of causal implications involve multiple factors. In order to identify the cause-and-effect relation the following advices can be mentioned (Asnina and Osis, 2011):

- Words and phrases that can signal relations, e.g., accordingly, because, effect, in order that, since, cause, for, therefore, as a result, if…then, why, consequently, due to, etc.
- Causative verbs, e.g., have, get, let, allow, require and so on.
- Some suffixes that indicate changes, causes and effects: "-ate" can mean to become, to cause (e.g., to update), "-ation" – the result of -ing (e.g., registration is the result of registering process), "-ize" – to make, a cause to be (e.g., finalize), etc.

Table 5: Logical relations: CER denotes cause-and-effect relations, LO denotes a logical operation on the set of cause-and-effect relations.

CER	LO	Explanation
2-3.1, 2-3.2	AND	LO is indicated in the description "<verb phrase 1> and <verb phrase 2>"
3.1-4, 3.2-4, 14-4	AND	LO is implicit and is inferred using logical speculations.
6-7, 6-8	AND	LO is indicated in the description under the same precondition "*If the book is found*"
7-9, 8-9	OR	LO is indicated in the precondition "*If the end date is exceeded OR the condition is not good*".
9-10, 9-12	XOR	LO is indicated as two text blocks under mutually exclusive preconditions: 1) *If the end date is not exceeded AND the condition is good*, and 2) *If the end date is exceeded OR the condition is not good*"
11-12, 9-12	XOR	LO is inferred using logical analysis of chains of cause-and-effect relations: getting the same state from two mutually exclusive paths.
11-13, 11-12	AND	LO is implicit.

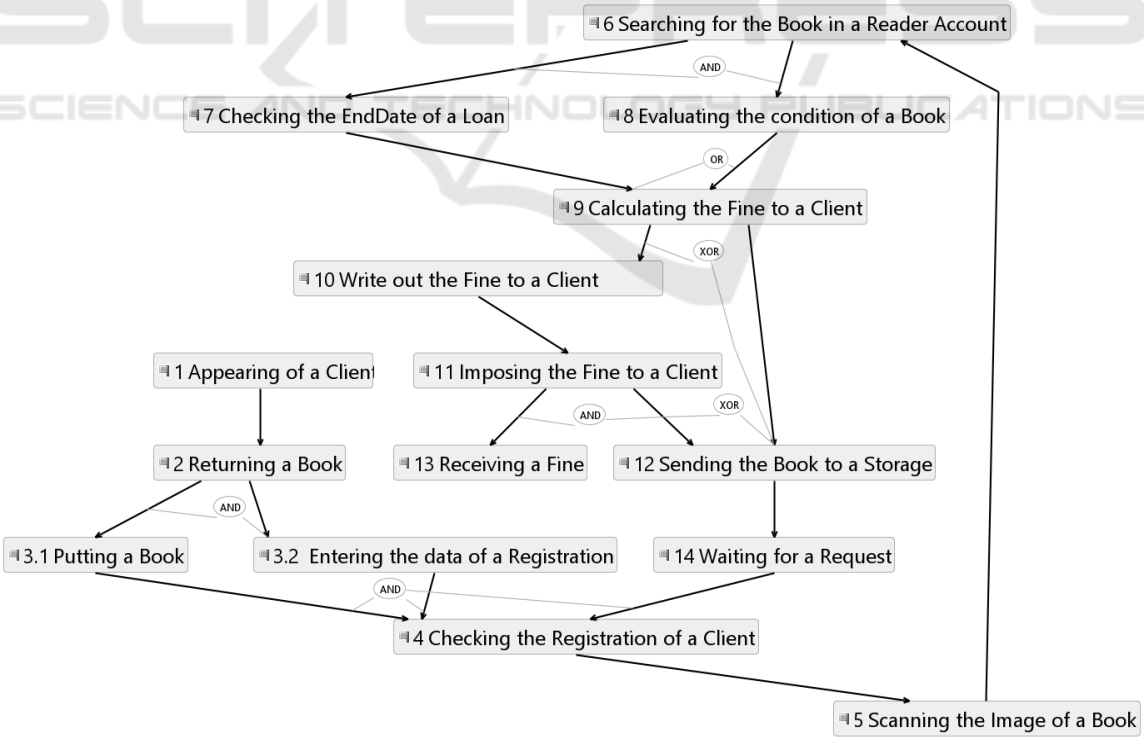

Figure 1: The topological space for the domain with the book returning machine.

In the running example, the chronological and conditional sequences are got using text analysis and inferring based on the experience (Table 5). The composed topological space is illustrated in Figure 1. If we decide to get the TFM of the book returning machine, then we should join neighbourhoods of functional features, where a provider is Machine. In this example, the TFM is equal to the constructed topological space.

The topological cycles are identified as closed paths in the graph, and they are 4-5-6-7-9-10-11-12-14-4, 4-5-6-7-8-10-11-12-14-4, 4-5-6-7-9-12-14-4, and 4-5-6-8-9-12-14-4. The order of cycles must be defined by the expert.

Analysing nouns, we can find that words "registration" and "loan" indicate the processes that are not mentioned in the description, but it must be done before.

4 NEEDED CHARACTERISTICS OF NLP TOOLS

4.1 NLP Application for Constructing the TFM: A Vision

In order to define what characteristics of the NLP tools are needed, let us first consider the scheme of the knowledge frame system and required knowledge. According to the initial scheme of the knowledge frame system (Nazaruks and Osis, 2017), the lists of elements (Section 3.2) should be kept in the instances of the following frame classes:

- manually filled in: FunctionalFeature and Properties,
- manually filled in with some generable values of slots: Object and TopologicalCycle,
- generated: CauseAndEffectRelation and TopologicalOperations, and identifiers of all instances of all frame classes.

The frame class CauseAndEffectRelation has slot values that are generated based on the facts that the cause is specified by its precondition, while the effect is specified by its postcondition (Donins, 2012b). Instances of the frame class TopologicalOperation have the name that must be set "as a union of values of slots 'action' and 'result' of FunctionalFeature, the slot 'owner' gets its value based on the value of slot 'object' in the frame FunctionalFeature, and slot 'returnType' by a type of the value of the slot 'result'" (Nazaruks and Osis, 2017).

In instances of the frame class Object the only generated values are those of slot topologicalOperation, and in instances of the TopologicalCycle – values of slot functionalFeatures that holds a set of references to functional features that are involved in a cycle (closed path).

In this research the focus is put on knowledge that is to be added manually, i.e., values for slots of frame classes Object, Property, and FunctionalFeature. The frame TopologicalCycle is skipped since it requires manual human participation only in determination of the order of cycles.

In the IDM (Slihte, 2015), use case scenario text processing is performed by The Stanford Parser for identifying the executors (**Ex**) and the description of the functional feature that is a union of O, A and **R** in accordance with the following sentence parsing rules:

- Sentences of use case steps must be in simple form to answer a question "Who does what?";
- Identify *coordinating conjunctions* to split a sentence into several clauses, and, thus, several functional features;
- Identify the *verb phrase* (VP) that is considered as a union of action A, object O and result R;
- Identify the *noun phrase* (NP) that can be either object O, result R or an executor Ex;
- Preconditions are taken directly from the corresponding preceding events;
- Postconditions are next functional features in the scenario or sequential events.

In case of formal but unstructured text we cannot use the same principles for discovering pre- and postconditions, while others are suitable. Therefore, NLP tools must be able to perform all four NLP tasks: "tokenization, part-of-speech (POS) tagging, chunking, and Name Entity Recognition (NER)/Classification" (Pinto, Oliveira and Oliveira Alves, 2016) as well as semantic analysis of noun and verb phrases. 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. The general scheme of textual description processing should be as in Figure 2.

4.2 NLP Pipelines for Assistance in the Topological Functioning Modelling

There are a lot of NLP tools (Pinto, Oliveira and Oliveira Alves, 2016) that allow processing text as in a formal as in an informal style (such as used in forums, blogs, and chats). Our research is oriented on finding tools that work with the formal style, since processing is needed for formal documentation.

In our research we want to focus on non-commercial pipeline solutions that do not require

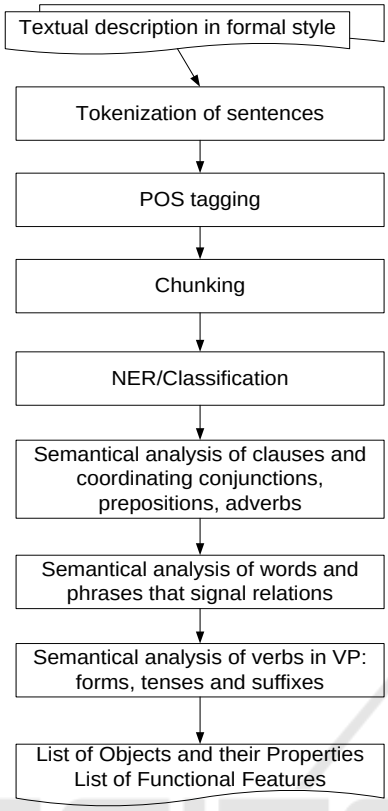

Figure 2: The general scheme of textual description processing for construction of the TFM.

complex installation and that are not complex to understand. The language of processed text is English. The list of pipeline software solutions includes Apache OpenNLP, FreeLing, GATE, LingPipe, Natural Language Tool Kit (NLTK), and StanfordNLP (Rodrigues and Teixeira, 2015). LingPipe has a free licence and a commercial one.

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. It can be accessed from many programming languages, e.g. Java, Python, Ruby, and .NET C#/F#. 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 further identification of actions and corresponding objects, results, modes (that can serve for identification of causal dependencies), executors and providers. Besides that, the Standford CoreNLP implements mention detection and pronominal and nominal coreference resolution that can help in dealing with pronouns and noun phrases that denote concrete phenomena.

Apache OpenNLP (Apache OpenNLP Development Community, 2017) is a machine learning based toolkit. It provides Java library for such tasks as tokenization, sentence segmentation, POS tagging, NER, chunking, parsing and coreference resolution. OpenNLP includes components that allow training and evaluating models. The components are available via their Application Programming Interfaces (APIs). NER can find named entities and numbers using a model of entity types for a language. OpenNLP has several pre-trained models for English, i.e. for dates, locations, money, organizations, percentages, persons, and time. OpenNLP can categorize document contents into predefined categories. The POS tagger marks tokens with their word type and the context using a probability model. The POS tagger may be trained on annotated training sentences and can find base forms of words. Opposite to the Stanford CoreNLP, in chunking OpenNLP provides only separation of syntactically correlated groups of words (noun phrases and verb phrases), but it does not specify their internal and external dependencies. At the present, the parser that is intended for discovering dependencies is developed only for demonstration and testing. Coreference resolution is also limited to noun phrase mentions. The main advantage of OpenNLP is possibility to train models.

The third pipeline is FreeLing (Carreras *et al.*, 2004; Padró *et al.*, 2010; Padró and Stanilovsky, 2012). FreeLing is a C++ library for multi-language NLP tasks that supports customisation. The supported tasks are tokenization, sentence splitting, morphological analysis, NER, POS tagging, Word Sense Disambiguation, Semantic Role Labelling, dependency parsing, shallow parsing, WordNet-based sense annotations, and coreference resolution. NER allows recognizing and classifying dates, numbers, physical magnitudes, currency, ratios. The morphological analysis supports number, quantity and date recognition from word groups, customization of the behaviour of the analysis chain triggered by a regular expression, aggregation of multiple words in a single word object, NER with the precision about 85% for the “basic” module and over 90% for the “bio” module. FreeLing supports analysis of alternatives for words, searching all senses for a word or performing word-sense-disambiguation. The

Named Entity Classification module differs from NER with predefined four classes: Person, Geographical location, Organization, and Others. Besides that, it allows defining the context features that extend the predefined classes. Chunking is performed by the Chart Parser Module. Dependency parsing is either based on rules defined in the file, or uses a statistical dependency parsing module. Coreference resolution allows mention detection and feature extraction. The advanced Semantic Graph Extractor Module supports building a semantic graph that encodes events, relations among events and the actors participating in those events. This module could be useful for discovering causal dependencies of TFM functional features.

GATE is a toolkit that includes "a desktop client for developers, a workflow-based web application, a Java library, an architecture and a process" (The University of Sheffield, 2018). The Information Extraction System supports the following tasks: tokenisation; sentence splitter; morphological analysis (Lemmatiser for determination of base forms of words, Gazetteer for identification of entity names (such as currency, days) in the text based on lists); POS tagger; NER performed by the Semantic Tagger that supports such entity types as Person (and gender), Location and its sub-types, Organization and its sub-types, Money, Percent, Date in the form of date, time and dateTime, Address, Identifier and Unknown; coreference resolution by OrthoMatcher between named entities, and pronominal coreference in noun phrases. In GATE, chunking is supported by several predefined rules and can be customized by a developer.

LingPipe (Carpenter and Breck, 2011; Alias-i, no date) is a Java-based toolkit for NLP processing using computational linguistics. The main tasks are NER and Classification, POS tagging and chunking, and Chinese Word Segmentation. Besides that, it supports sentence splitting, spelling correction, sentiment analysis, singular value decomposition and word sense disambiguation.

The last one, the NLTK framework (Bird, Loper and Klein, 2009; NLTK Project, 2017), is dedicated for programs on Python and provides easy-to-use interfaces for multiple corpora and lexical resources such as WordNet. At the beginning, the NLTK was dedicated to processing textual documents in the formal style, but now it can use also corpora for different genres. The NLTK supports a usage of users' corpora and corpora in other but English languages. A use of different corpora supports semantic analysis of the type "is-a-part-of", discovering synonyms and homonyms. The NLTK supports the tasks of tokenization, POS tagging, chunking, and NER. For example, tagging of verbs can help in analysis of their forms: Are they, for example, in past tense or they have the past participle form? In order to rise the precision of tagging, it is possible to combine the taggers provided.

The NLTK implements Named Entity Classification via decision trees, naive Bayes classifiers, and probabilistic models. Chunking relates as to noun phrases as to verb phrases and supports a use of regular expressions for determination of word groups and their hierarchy as well as for word groups that should be excluded from this process.

The NER includes recognition of the following entity types: ORGANIZATION, PERSON, LOCATION, DATE, TIME, MONEY, PERCENT FACILITY, and GPE (for geo-political entities). The interesting function suggested in the NLTK is Relation Extraction that allows extracting relations among named entities. The NLTK supports also analysis of phrase structure grammars, dependencies and dependency grammars. It could be useful for discovering the structure of functional characteristics by analysing the structure of a sentence and causal dependencies between sentences or clauses.

Table 6: NLP tools for knowledge extraction activities for the topological functioning modelling: T – tokenization, POS – POS tagging, CH – chunking, NER – NER/Classification, and assistance in A-CL – analysis of dependencies between clauses/sentences, A-NP – analysis of dependencies in complex noun phrases, A-PR – analysis of dependencies in predicates, A-VP – analysis of verbs in verb phrases.

NLP tools	Activity							
	T	POS	CH	NER	A-CL	A-NP	A-PR	A-VP
Stanford CoreNLP toolkit	v	v	v	v	v	v	v	v
Apache OpenNLP	v	v	v	v	v			
FreeLing	v	v	v	v	v	v	v	v
GATE	v	v	v	v		v		
LingPipe	v	v	v	v				
NLTK framework	v	v	v	v	v	v	v	v

However, the accuracy of this analysis can be decreased because of syntactic ambiguity of sentences in NL.

The support of the defined tasks of the topological functioning modelling by the considered NLP tools is summarized in Table 6. The largest support of the required tasks comes from the Stanford CoreNLP toolkit, FreeLing and NLTK framework.

5 CONCLUSIONS

The topological functioning modelling is based on knowledge extraction from multiple verbal descriptions of the functions, behaviour, phenomena and structure of the domain. In case of introducing the frame system as a core storage for those knowledge, extraction of it is likely to be automated to be more valuable for software developers or business/system analytics.

The NLP tools support automated knowledge extraction from formal and informal text. The precision of the extraction may differ, but some tools support training a model for NL text processing.

We have defined the tasks for text processing in order to get knowledge necessary for the topological functioning modelling. They are tokenization, POS tagging, chunking, NER/Classification, and assistance in analysis of dependencies between sentences, in complex noun phrases and verb phrases, in predicates, and analysis of verb forms and tenses. The most difficult tasks relate to analysis of dependencies among sentences, verb phrases and predicates. Some dependencies are not explicitly indicated in the text, thus inferring is required.

All the tasks in some degree are supported by the Stanford CoreNLP toolkit, FreeLing and NLTK framework.

We plan to perform research on opportunities provided by these toolkits in more detail. The future research direction relates to practical experiments with the three pipelines in order to evaluate the easiness of implementation of knowledge extraction as well as precision of the discovered knowledge, extension of tool functions and to understand their performance issues.

REFERENCES

Alias-I, no date. *LingPipe Home, LingPipe*. Available at: http://alias-i.com/lingpipe/index.html (Accessed: 10 January 2018).

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. doi: 10.1145/1088622.1088651.

Apache OpenNLP Development Community, 2017. *Apache OpenNLP Developer Documentation, Version 1.8.4.* The Apache Software Foundation. Available at: https://opennlp.apache.org/docs/1.8.4/manual/opennlp.html (Accessed: 10 January 2018).

Asnina, E., 2006. The Computation Independent Viewpoint: a Formal Method of Topological Functioning Model Constructing, *Applied computer systems*, 26, pp. 21–32.

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. doi: 10.5220/0003043200230032.

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. doi: 10.4018/978-1-61692-874-2.ch003.

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 & Software*, 88, pp. 25–41. doi: 10.1016/j.jss.2013.08.036.

Bird, S., Loper, E. and Klein, E., 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Cannataro, M., Guzzo, A. and Pugliese, A., 2002. Knowledge management and XML: derivation of synthetic views over semi-structured data, *ACM SIGAPP Applied Computing Review*. ACM, 10(1), p. 33. doi: 10.1145/568235.568242.

Carpenter, B. and Breck, B., 2011. *NLP with LingPipe*. Draft 0.5. Available at: http://alias-i.com/lingpipe-book/index.html (Accessed: 10 January 2018).

Carreras, X., Chao, I., Padró, L. and Padró, M., 2004. FreeLing: An Open-Source Suite of Language Analyzers, in *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*.

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

Donins, U., 2012b. *Topological Unified Modeling Language: Development and Application*. Riga Technical University.

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. doi: 10.1016/j.jss.2011.02.025.

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. doi: 10.1145/2882879.2882882.

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. doi: 10.1007/978-3-642-21640-4_36.

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. doi: 10.1109/SERA.2006.51.

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. doi: 10.1109/WCCAIS.2014.6916558.

Jones, D. E., Igo, S., Hurdle, J. and Facelli, J. C., 2014. Automatic Extraction of Nanoparticle Properties Using Natural Language Processing: NanoSifter an Application to Acquire PAMAM Dendrimer Properties, *PLoS ONE*. Edited by V. Ceña, 9(1), p. e83932. doi: 10.1371/journal.pone.0083932.

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. doi: 10.1109/ICCCCT.2010.5670730.

Leopold, H., Mendling, J. and Polyvyanyy, A., 2014. Supporting Process Model Validation through Natural Language Generation, *IEEE Transactions on Software Engineering*, 40(8), pp. 818–840. doi: 10.1109/TSE.2014.2327044.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J. and Mcclosky, D., 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), Architecture Board ORMSC*. Available at: http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01.

Nakamura, Y., Takahashi, M., Onda, M. and Ohta, Y., 1996. Knowledge extraction from diagram and text for media integration, in *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*. IEEE Comput. Soc. Press, pp. 488–492. doi: 10.1109/MMCS.1996.535897.

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. doi: 10.1109/IACS.2015.7103200.

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. doi: 10.5220/0006388903790390.

NLTK Project, 2017. *Natural Language Toolkit — NLTK 3.2.5 documentation*. Available at: http://www.nltk.org/ (Accessed: 10 January 2018).

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. doi: 10.4018/978-1-61692-874-2.ch001.

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. doi: 10.4018/978-1-61692-874-2.ch002.

Osis, J., Asnina, E. and Grave, A., 2007. MDA oriented computation independent modeling of the problem domain, in *Proceedings of the 2nd International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE 2007*. Barcelona: INSTICC Press, pp. 66–71.

Osis, J., Asnina, E. and Grave, A., 2008. Formal Problem Domain Modeling within MDA, in Filipe, J., Shishkov, B., Helfert, M., and Maciaszek, L. A. (eds) *Software and Data Technologies: Second International Conference, ICSOFT/ENASE 2007, Barcelona, Spain, July 22-25, 2007, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 387–398. doi: 10.1007/978-3-540-88655-6_29.

Osis, J. and Slihte, A., 2010. Transforming Textual Use Cases to a Computation Independent Model, in Osis, J. and Nikiforova, O. (eds) *Model-Driven Architecture and Modeling-Driven Software Development: ENASE 2010, 2ndMDA&MTDD 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. doi: 10.12948/issn14531305/20.4.2016.01.

Padró, L., Collado, M., Reese, S., Lloberes, M. and Castellón, I., 2010. FreeLing 2.1: Five years of open-source language processing tools, *Proceedings of 7th Language Resources and Evaluation Conference (LREC 2010), ELRA La Valletta, Malta. May, 2010.*

Padró, L. and Stanilovsky, E., 2012. FreeLing 3.0: Towards Wider Multilinguality FreeLing project developer, in

Proceedings of the Language Resources and Evaluation Conference (LREC 2012) ELRA. . Istanbul, Turkey: European Language Resources Association (ELRA).

Pinto, A., Oliveira, H. G. and Oliveira Alves, A., 2016. Comparing the Performance of Different NLP Toolkits in Formal and Social Media Text *, *5th Symposium on Languages, Applications and Technologies (SLATE'16). Open Access Series in Informatics.*, p. 3:1-3:16; Article No.3; doi: 10.4230/OASIcs.SLATE.2016.3.

Rodrigues, M. and Teixeira, A., 2015. *Advanced Applications of Natural Language Processing for Performing Information Extraction*. Cham: Springer International Publishing (SpringerBriefs in Electrical and Computer Engineering). doi: 10.1007/978-3-319-15563-0.

Slihte, A., 2015. *The Integrated Domain Modeling: an Approach & Toolset for Acquiring a Topological Functioning Model*. Riga Technical University.

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

Šlihte, A. and Osis, J., 2014. The Integrated Domain Modeling: A Case Study, in *Databases and Information Systems: Proceedings of the 11th International Baltic Conference (DB&IS 2014)*. Tallinn: Tallinn University of Technology Press, pp. 465–470.

The University of Sheffield, 2018. *GATE: a full-lifecycle open source solution for text processing*. Available at: https://gate.ac.uk/overview.html (Accessed: 10 January 2018).