Code Comment Classification Taxonomies

Marija Kostić, Aleksa Srbljanović, Vuk Batanović and Boško Nikolić, *Member, IEEE*

***Abstract*—Code comments have become an increasingly important kind of software development metadata, due to the possibilities of automated code comment analysis and generation. Different downstream tasks inherently prioritize certain kinds of code comments over others, making it necessary to properly define and identify different comment classes. In this paper, we analyze, compare, and systematize previously proposed code comment classification taxonomies according to their comment classes and applicability. We also present a new taxonomy designed for the tasks of semantic code search and semantic text similarity, and we contrast it to the existing approaches.**

***Index Terms*—code comments; code comment taxonomy; comparison of classification taxonomies.**

I. Introduction

Code comments represent an invaluable source of metadata regarding software implementation. They describe code functionalities and algorithmic specifics, provide usage instructions, point towards additional resources, denote potential or observed programming bugs and issues, etc. In short, code comments play a vital role in helping developers comprehend source code [1]. In this manner, code comments greatly increase code maintainability, particularly when dealing with large software projects and development teams.

Depending on the downstream task in focus, not all code comments are of equal importance. For instance, if one wishes to compare the functionality of two methods, comments which provide authorship information are of little consequence, whereas those describing program implementation are of much greater significance. However, distinct kinds of code comments can be difficult to distinguish, particularly when no clear keywords for each comment type exists. A further complication in identifying relevant comments is the fact that a standardized code comment taxonomy does not exist. Instead, multiple different code comment categorization solutions have been proposed so far, most often designed with a specific programming language and downstream task in mind.

In this paper, we first present a survey of the existing code comment taxonomies and their applications in downstream tasks. Afterward, we present a new comment classification schema suitable for the tasks of semantic code search and semantic textual similarity and applicable to various programming languages. We then compare our approach with previous code comment taxonomies and conclude with some pointers regarding the future use of our comment schema.

Marija Kostić is with the School of Electrical Engineering and the Innovation center of School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: marija.kostic@ic.etf.bg.ac.rs), (https://orcid.org/0000-0003-4923-3748).

Aleksa Srbljanović is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia, (e-mail: aleksa.srbljanovic@etf.bg.ac.rs).

Vuk Batanović is with the Innovation center of School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade Serbia, (e-mail: vuk.batanovic@ic.etf.bg.ac.rs), (https://orcid.org/ 0000-0003-2639-9091).

Boško Nikolić is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia, (e-mail: bosko.nikolic@etf.bg.ac.rs), (https://orcid.org/ 0000-0003-1142-9243).

II. A Survey of Existing Taxonomies

In this section, we review existing code comment classification taxonomies and describe how those systems were applied to specific tasks. A similar, but much shorter survey of this kind was previously presented within [2].

Zhai et al. [3] wanted to leverage program analysis to systematically propagate comments, so that they can be passed on to uncommented code entities and help detect bugs. A bi-directional analysis was designed where: (1) program analysis propagates and updates code comments, and (2) comments provide additional semantic hints to enrich program analysis. For effective propagation, it was vital to understand what kind of information comments convey and to which code elements they refer to, as comments of different categories require different propagation rules. The authors introduced a code comment taxonomy in which there are two dimensions of interest: code entity and content perspective. Code entities commonly commented on are *class*, *method*, *statement*, and *variable*. As for the content perspective, five of them were identified: 1) *what* – a definition or a summary of the code entity's functionality; 2) *why* – the reason why the code entity is provided or its design rationale; 3) *how-it-is-done* – description of the implementation details; 4) *property* – properties of the entities such as pre-condition and post-conditions; 5) *how-to-use* – description of the usage, expected set-up, or the environment of the entity. A set of 5,000 comments from four popular Java libraries were classified at the sentence level. The agreement between two coders, measured using Cohen's Kappa metric [4], was 0.826.

Chen et al. [1] investigated the use of the relationship between code blocks and the categories of the corresponding comments to improve code summarization, where the aim is to automatically generate a code comment based on the given block of source code. They showed that a composite approach, where the most suitable summarization model is selected based on the comment category, outperforms other approaches. Comments were classified into six categories – *what*, *why*, *how-to-use*, *how-it-is-done*, *property* and *others*. Five of them are the same as categories of content perspective in [3], while the sixth one, named *others*, covers unspecified or ambiguous comments. For this task, 20,000 Java methods and their corresponding comments were manually classified.

 ISBN 978-86-7466-930-3

The overall agreement of the three annotators, expressed in the terms of Fleiss' Kappa score [5], is 0.79.

Padioleau et al. [6] studied code comments to understand developers' needs regarding the creation of new tools and languages or the improvement of the existing ones. Comments were classified along four dimensions according to the question of interest: 1) *What?* – content: What is in the comment? Does it contain useful information? Its categories *type*, *interface*, *code relationship* and *pastFuture* and subcategories are closely related to the specific usage of the C programming language for operating systems; 2) *Who?* – people involved: Who or which tool can benefit from a comment? Who is the comment author?; 3) *Where?* – code entity: Where in a file is a comment located?; 4) *When?* – time: When was a comment written? How did the comment evolve over time? The authors considered 1050 comments randomly sampled from the code of three operating systems written in the C programming language: Linux, FreeBSD and OpenSolaris. The *What?* and *Who?* dimensions were manually annotated for each comment, while the other two dimensions were labeled automatically.

Haouari et al. [7] investigated developers' commenting habits via an empirical study. For the quantitative aspect of the study, the authors determined the distribution of the comments with respect to the program construct type that follows it. This allowed them to see what program construct types are documented more often than others. Some of the observed constructs are *package declaration*, *import declaration*, *class declaration*, *method*, *constructor*, *for*, *while*, etc. For the qualitative aspect of the study a new comment taxonomy was designed. This taxonomy has four high-level dimensions: 1) *Object of the comment* which can be a single subsequent instruction (*follow*); the following block of instructions (*block*), no code in the vicinity (*nocode*), or any other situation (*other*); 2) *Comment type* which can be the description of the code functionality (*explanation*), future task to be completed like TODO items (*working*), old code that is commented out instead of being removed (*code*), or any other comments (*other*); 3) *Style* dimension is only observed in the case of explanatory comments (*type=explanation*) and can be either *explicit* or *implicit*; 4) *Quality* dimension is also specific only to explanatory comments. It involves three categories: *fair+* where comments describe functionalities of related code and give other information; *fair* where code functionality is adequately described; and *poor* where some or none of the functionality is described. Analysis for the quantitative aspect was fully automated and applied to all comments within three open-source Java projects. For the qualitative aspect, the authors had 49 developers manually classify 407 comments.

Steidl et al. [8] developed a model for comment quality analysis with four criteria (*coherence*, *usefulness*, *consistency*, and *completeness*). Their comment taxonomy consists of seven high-level categories: 1) *copyright* – copyright or license; 2) *header* – overview of the class functionality; 3) *member* – functionality of a method/field; 4) *inline* – implementation decisions within a method body; 5) *section* – group of methods/fields that belong to the same functional aspect; 6) *code* – commented out code; 7) *task* – developer notes with a remaining todo, a bug, or an implementation hack. Authors created a training set by classifying 830 Java and 500 C++ comments from twelve open-source projects.

Pascarella et al. [9]–[11] focused on increasing the empirical understanding of the types of comments that developers write in source code. After an iterative process of analyzing code files, the authors defined a fine-grained hierarchical taxonomy with two layers: the outer one consisting of six top-level categories (*purpose*, *notice*, *under development*, *style & IDE*, *metadata*, and *discarded*) and the inner one consisting of 16 subcategories. The *purpose* category contains comments that describe the functionality of the related source code. Its three subcategories *summary*, *expand*, and *rationale* respond to the question words what, how, and why, similarly to the categories in [1], [3]. The *notice* category covers comments about warnings, alerts, messages, or functionalities that should be used with care. Its subcategories are *deprecation*, *usage*, and *exception*. Subcategories *todo* (explicit actions to be done), *incomplete* (partial or empty comment bodies), and *commented code* belong to the *under development* top-level category. The *style & IDE* comments are used for communication with the IDE (*directive*) and logical separation of the code (*formatter*). The *metadata* comments define meta information about the code such as license, terms of use, authors, links to external resources (subcategories *license*, *ownership*, and *pointer*). All other comments that do not fit in the previous categories belong to the *discarded* category (subcategories *automatically generated* and *unknown*). Authors decided to classify comments at the character level. That means that annotators had to specify the starting and the ending character of each comment block and its category. After this process, it was found that in only 4% of cases one line had to be classified into more than one category. The study was conducted on more than 6,000 source code files with more than 40,000 lines of Java comments in open source and industrial software projects. To validate the proposed taxonomy, three developers were asked to manually classify 138 lines of comments in three Java source code files. They achieved a Fleiss' Kappa value of 0.9.

Unlike previously mentioned efforts, Shinyama et al. [12] worked only on comments inside functions or methods that explain code at the microscopic level i.e., on *local comments*. These comments are not visible in the documentation and often give insight into developers' minds. They are often crucial for understanding nontrivial parts of the code. As there usually is a relationship between a comment and the code it describes, the authors represented that relationship as an arc with three elements: (1) source – the comment itself; (2) destination – target code; and (3) type of relationship – comment category. They independently made a list of categories suitable for local comments: 1) *postcondition* – conditions that hold after the code is executed, typically used to explain what the code does; 2) *precondition* – conditions that hold before the code is executed, typically used for explaining why the code is needed; 3) *value description* – a

 ISBN 978-86-7466-930-3

phrase that can be equated with a variable, constant or expression; 4) *instruction* – instructions for code maintainers (todo comments); 5) *guide* – guides and examples for code users; 6) *interface* – description of a function, type, class, or interface; 7) *meta information* – author, date, or copyright; 8) *comment out* – commented out code; 9) *directive* – compiler directives that are not directed to human readers; 10) *visual cues* – comments inserted just for the ease of reading; 11) *uncategorized* – all other comments. For each arc element, a statistical classifier was built and trained on 1,000 manually classified Java comments. Classifiers were applied on large corpora of Java and Python comments. Annotation agreement was measured on a separate set of 100 Java comment-code pairs and reached Fleiss' Kappa value of 0.491.

Zhang et al. [13] used supervised learning to automatically classify Python code comments. Since most of the related work is based on Java and C/C++ programming languages, the authors conducted an iterative content analysis session to devise a Python-specific classification taxonomy. Their taxonomy contains 11 categories: 1) *metadata* – license and copyright; 2) *summary* – description of the functionality of the related code; 3) *usage* – explanations on how to use the code that can contain examples; 4) *parameters* – explanations of function parameters; 5) *expand* – detailed explanations of the purpose of a small block of code, usually inline comments; 6) *version* – library version information; 7) *development notes* – comments for developers concerning ongoing work, temporary tips, explanations of functions etc.; 8) *todo* – explicit actions to be done in the future like bug fixing or feature improving; 9) *exception* – indications that a function throws exceptions or suggestions how to prevent unwanted behaviors; 10) *links* – links to external resources; 11) *noise* – meaningless symbols which may be used for separation. The training set consisted of 330 annotated comments from seven popular Python open-source projects on GitHub.

III. A New Code Comment Taxonomy

We explored code comment classification taxonomies in order to differentiate between different kinds of comments for the tasks of semantic code search and cross-level sematic textual similarity. In semantic code search (SCS), the goal is to construct a system which returns the most relevant code block(s) from a software repository for a given query in a natural language. To do so, most models rely on the accompanying code comments which describe the functionality of their respective code blocks [14]. Cross-level semantic similarity (CLSS) is the task in which a computational model ought to return a numerical semantic similarity score for a given pair of texts written in a natural language, where the length of the texts can be dissimilar (e.g., one text is a paragraph, the other is a sentence) [15]–[16]. Solving CLSS is of great use for semantic code search, since SCS implies finding semantic links between texts of different lengths – queries are usually limited to a couple of words or a sentence, whereas the length of code comments can range from a phrase to a paragraph. Obviously, these tasks are not limited to a particular programming or natural language.

We found that no previous code comment taxonomy was designed with these two downstream tasks in mind, so it was necessary to consider the previous classification systems and devise a suitable set of comment categories. Furthermore, we wanted to create a classification taxonomy that would be applicable to various programming languages, including C, C++, C#, Java, JavaScript/TypeScript, PHP, Python, and SQL. Upon reviewing the papers presented in the previous section, we decided to develop our own taxonomy using the approaches of Pascarella et al. [9]–[11] and Steidl et al. [8] as a starting point. This choice was based on the emphasis these works placed on the comments that describe code functionality, since such comments are the most relevant ones for SCS. We therefore distinguish between functional and non-functional comments via two top-level categories. These two categories are then subdivided into eight subcategories. In the remainder of this section, we will present the definition and scope of each of them.

A. Functional

The *Functional* category contains comments that describe the functionality of the corresponding source code. Descriptions can be short, or they can extend over multiple lines. These comments are usually written in a natural language and are used to describe the purpose, behavior, or the reason why something is implemented in a particular way. They can respond to questions *What?*, *Why?*, and *How?*. We do not distinguish between these aspects of functionality because all of them are relevant for the SCS task. However, we do differentiate between three subcategories based on the type of the corresponding source code: 1) *Functional-Module* comments describe the functionality of a particular module like a class or an interface. If a programming language does not use the object-oriented paradigm, these comments pertain to entire files or scripts; 2) *Functional-Method* comments describe the functionality of a function or a method. They are usually located above or at the beginning of a function/method definition or declaration; 3) *Functional-Inline* are all the other comments that describe some functionality. They can describe the functionality of a variable or an expression and can be located inside a method body.

B. Non-Functional

The *Non-Functional* category covers all comments that do not describe code functionality. Subcategories in this top-level category are not relevant for the SCS and CLSS tasks, but we still decided to include them because we wanted to make the annotated datasets usable for other downstream tasks as well. We distinguish between the following five subcategories: 1) The *Notice* category encompasses warnings, alerts, and messages intended for other developers or users of the source code. It also covers information about deprecated artifacts and instructions about alternative methods or classes that should be used. Comments that explain something is implemented in a certain way because of a bug, or a known issue, also belong to this category. Finally, examples or explicit suggestions how

 ISBN 978-86-7466-930-3

to use a functionality are classified as *Notice* comments as well; 2) *General* comments usually define meta-information about the code such as license, copyright, authorship, module/class version, timestamps, the name or path of the file, information about the libraries used in the source code, etc. These comments are usually located at the top of the file; 3) The *Code* category is composed of comments that contain source code that is commented out by developers. This is usually done during testing or debugging. This code may represent new or hidden features, work in progress, features being tested, temporarily removed code or older variants of the code; 4) *IDE* comments are used for communication with the IDE or the compiler to change their default behavior. Comment content is usually of limited value to human

TABLE 1 CLASSIFICATION OF EXAMPLE CODE COMMENTS ACCORDING TO DIFFERENT TAXONOMIES

Class	Class	Example	Haouari [7]	Steidl [8]	Shinyama [12]	Zhang [13]	Pascarella [11]	Zhai [3]	Chen [1]	Our proposal
Functionality by type	What	`Pushes an item onto the top of this stack.` [3]	Type-Explanation	Header|Inline|Member	Postcondition	Summary	Summary	What	What	Functional
	Why	`It eliminates the need for explicit range operations.` [3]			Precondition	-	Rationale	Why	Why	
	How	`Shifts any subsequent elements to the left.` [3]			Uncategorized	Expand	Expand	How-it-is-done	How-it-is-done	
	Property	`The index must be a value greater than or equal to 0.` [3]				-	-	Property	Property	
Functionality by position	Module	`This class is a member of the Java Collections Framework.` [3]		Header	Interface	Summary	Purpose	Class	What|Why How-it-is-done|Property	Functional-module
	Method	`Check for symmetry, then construct the eigenvalue decomposition @param A square matrix` [8]		Member		Summary|parameters		Method		Functional-method
	Inline	*`Increment this when there's a change requiring caches to be invalidated.`*		Inline	(Post|pre) condition	Expand		Statement		Functional-inline
	Variable/Field	`The number of characters to skip.` [3]		Member	Value description			Variable		
Section		`--- Getter and Setter Methods ---` [8]	Type-Other	Section	Visual cue	-	Formatter	-	Others	Functional-inline
Automatically Generated		*`COLORCORRECTION_HPP`*		-	Uncategorized	Noise	Automatically generated	-		
Notes		`Caution: setting a new service manager stub won't replace the existing one.` [7]		Task		Development notes	Todo	-		Notice
Usage		`Example: renderText(100, 100, FONT, 12, "Hello");` [12]	Type-Working	-		Usage	Usage	How-to-use	How-to-use	
Deprecation		*`DEPRECATED: the following property is no longer in use, but defined until 2.0 to prevent conflicts`*		-	Guide	Development notes	Deprecation	-	Others	
Exception		*`@throws TransportExceptionInterface When an unsupported option is passed`*		-		Exception	Exception	-		Functional-method
Link		*`See https://github.com/symfony/symfony/pull/5582`*		-		Links	Pointer	-		Notice
Bug		*`Skip due to crash bug: https://support.microsoft.com/en-us/help/2908087`*		Task		Development notes	Todo	-		Todo
Todo		`TODO: Check synchronization.` [7]			Instruction	Todo		-		
Code		`_mainFrame.hourglassOff();` [7]	Type-Code	Code	Commented out	Noise	Commented code	-		Code
Ownership		*`@author Ben Ramsey <ben@benramsey.com>`*	Type-other	Header	Meta information		Ownership	-		General
Meta		*`Copyright(c) 2019 Intel Corporation.`*		Copyright		Metadata	License	-		
Version		*`@version $Revision: 1.0`*		Header	Uncategorized	Version	Unknown	-		
IDE		`CHECKSTYLE:OFF` [12]		-	Directive	-	Directive	-	IDE	
Other		`The implementation is awesome.` [1]		-	Uncategorized	Noise	Unknown	-		Notice

 ISBN 978-86-7466-930-3

readers; 5) The *Todo* category covers explicit tasks to be done and notes about bugs that need to be fixed.

IV. Comparison With Previous Code Comment Classification Taxonomies

In this section, we present a comparison between the previously described taxonomies. Table 1 shows examples of code comments and their classes according to different taxonomies. In places where we were not sure what category the authors would choose for a specific comment, we put a dash symbol. Comments in italic are new examples, while others are taken from the cited papers. We have omitted the taxonomy presented in [6] from the table since it is extremely specific regarding the type of code it is applied to (operating systems code). Additionally, its authors have not disclosed all the categories they have devised.

All the comments which describe code functionality belong to one of the following classes - *What*, *Why*, *How*, *Property*, *Module*, *Method*, *Inline* and *Variable/Field*. However, these classes can be classified based on two perspectives: (1) according to the type of the commented functionality – categories *What*, *Why*, *How*, and *Property*, or (2) the comment's structural position within the code – categories *Module*, *Method*, *Inline*, *Variable/Field*. Depending on the downstream task, some taxonomies use the functionality type classification [1], [13], others use the comment place classification [8], some use both [3], [12], or neither [7]. The new taxonomy we propose takes into account the placement of a comment within the code. It should be emphasized we do not differentiate between *Inline* and *Variable/Field* functional comments like in some taxonomies [3], [8], [12], but rather group them together under the *Functional-Inline* class.

In some papers there is a separate class for comments which visually divide the code into sections [8], [11]–[12]. We classify such comments as *Functional-Inline* as well.

Some comments do not describe code functionality, but rather contain notes to developers and code users. Several different classes for these kinds of comments have been previously proposed. There are authors [1], [3], [7]–[8] who recognize only some of these comments because not all of them are relevant for their downstream task. In other papers [11]–[13] most of these comments are recognized, but every paper uses a different approach concerning their classification. Some authors [11], [13] use a higher level of granularity while others [12] perceive some or all such comments as one class. In [12] the authors also differentiate between the comments meant for developers and those meant for users. Regarding TODO comments, some taxonomies [12]–[13] clearly separate them from the other comments, while in others [8], [11] there is an overlap between TODO and other comment categories. Our taxonomy distinguishes between *Notice* and *Todo* classes. All the notes for developers/users, use cases, warnings about deprecation, exceptions and links are classified as *Notice*. Messages about missing/unfinished parts of code and bugs are classified as *TODO*.

Several taxonomies [7]–[8], [11]–[12], treat comments which represent parts of code as a separate class, while others do not mention them. In [11], code which is commented out is classified as a *Todo* comment, along with comments for bugs and unfinished code. In our taxonomy parts of code which are commented out are placed in a separate class – *Code*.

Most previous approaches use a separate class for meta-information comments. Some [8], [11], [13] utilize a more granular classification according to license information, authorship, version information etc. In our approach all the meta-information is classified into the *General* category.

A few authors [11]–[12] have proposed a separate class for comments which represent some instructions for the compiler or the development environment. We also include an *IDE* category in our taxonomy.

Some papers [1], [7], [11]–[13] use a separate category for all other comments which are not of interest. However, our taxonomy does not employ such a category, because we do not want to allow annotators to easily dismiss ambiguous comments which are difficult to categorize.

Additional information about the presented comment taxonomies is shown in Table 2. It contains the number of comments that are/will be annotated for each taxonomy, the used comment granularity, the natural and programing languages each taxonomy is applied to, the annotation agreement (if reported), and the downstream task. Data from the table shows that annotations are typically done on small sets of code comments written in English, and that comments are taken from one or two programming languages at most. It is hard to compare annotation agreements because agreement measures differ from paper to paper and relate to different numbers of annotators and different comment set sizes.

Some of the earliest taxonomies were specific for the task they were solving [6]–[7] and observed more than two perspectives (e.g., object, style, beneficiary etc.). Because of their complexity, they are not useful for tasks other than the ones they were designed for. But, over time, two perspectives became prominent: (1) what is the entity of a comment, and (2) how a comment describes the functionality of the entity.

All papers in this survey worked with comments in English and in one of the following programming languages: C/C++, Java, or Python. As mentioned in [11] many object-oriented languages have very similar functionalities, and it is reasonable to expect that their comments will behave the same. We can see that in more recent works [1], [3], [8], [11]–[13] taxonomies designed for Java, Python, and C++ are similar. For other programming paradigms (e.g., functional), further research must be done.

Although some authors wanted only to empirically study and understand the types of code comments [11]–[13], most of the times classification was done as a first step in solving a particular downstream task. In a couple of papers, it is shown that that kind of approach is fruitful. For example, Chen et al. [1] have found that different summarization models work best for different categories of comments. By including comment

 ISBN 978-86-7466-930-3

TABLE 2 GENERAL INFORMATION ABOUT CODE COMMENT CLASSIFICATION TAXONOMIES

Paper	Year	Number of comments	Granularity	Language	Programming Languages	Annotation Agreement	Downstream task applicability
Padioleau [6]	2009	1050	Comment	English	C	-	Understanding developers' needs regarding the creation or improvement of tools and languages.
Haouari [7]	2011	407	Comment	English	Java	-	Investigation of developers commenting habits.
Steidl [3]	2013	1330	Comment	English	Java, C++	-	Quality analysis of source code comments.
Shinyama [12]	2018	1000	Comment	English	Java, Python	Fleiss' Kappa = 0.491	Analysis of comments inside functions or methods that often give insight into the developers' minds.
Zhang [13]	2018	330	Comment	English	Python	-	Classification of Python code comments.
Pascarella [11]	2019	40000	Character	English	Java	Fleiss' Kappa = 0.9	Increasing the empirical understanding of the types of comments that developers write.
Zhai [3]	2020	5000	Sentence	English	Java	Cohen's Kappa = 0.826	Code-comment propagation.
Chen [1]	2021	20000	Comment	English	Java	Fleiss' Kappa = 0.79	Code summarization.
Our proposal	2022	~10000	Character	English	C/C++, C#, Java, JavaScript/ TypeScript, PHP, Python, SQL	To be determined	Semantic code search and cross-level semantic textual similarity.

classification, they were able to design a composite summarization model that outperforms a standard approach where one model is applied to all comments. Another example is the work of Zhai et al. [3] focused on code comment propagation. Here, comment classification was necessary because comments with different content related to different programming entities cannot be propagated in the same way.

V. CONCLUSION

In this paper, we have analyzed and compared previously proposed code comment classification taxonomies. We have systematized them according to the comment classes they use as well as according to their applicability to different programming languages and downstream tasks. We have also presented a new comment taxonomy, designed for the tasks of semantic code search and semantic textual similarity, applicable to various programming languages.

In order to validate the usefulness of our taxonomy, we are currently engaged in the creation of a code comment corpus which will encompass around 10,000 comments written in English or Serbian, and taken from a spectrum of programming languages(C/C++, C#, Java, PHP, Python, SQL, and JavaScript/TypeScript). We aim to manually annotate this corpus using the proposed taxonomy and use it to enable automated comment classification, both as a stand-alone task and as a first step within the mentioned downstream tasks.

ACKNOWLEDGMENT

This work was supported by the Science Fund of the Republic of Serbia, grant no. 6526093, AI-AVANTES.

REFERENCES

[1] Q. Chen, X. Xia, H. Hu, D. Lo, S. Li, "Why My Code Summarization Model Does Not Work: Code Comment Improvement with Category Prediction," *ACM Trans. Softw. Eng. Methodol.*, vol. *30*, no. *2*, pp. *1-29*, Feb 2021.

[2] B. Yang, Z. Liping, Z. Fengrong, "A Survey on Research of Code Comment," Proc. 2019 3rd International Conference on Management Engineering, Software Engineering, and Service Sciences-ICMSS 2019, Wuhan, China, pp. 45-51, Jan. 12, 2019.

[3] J. Zhai, X. Xu, Y. Shi, G. Tao, M. Pan, S. Ma, L. Xu, W. Zhang, L. Tan, X. Zhang, "CPC: automatically classifying and propagating natural language comments via program analysis," Proc. ACM/IEEE 42nd International Conference on Software Engineering, Seoul, South Korea, pp. 1359-1371, Oct. 1, 2020.

[4] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. *20*, no. *1*, pp. *37-46*, 1960.

[5] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological Bulletin*, vol. *76*, no. *5*, pp. *378-382*, 1971.

[6] Y. Padioleau, L. Tan, Y. Zhou, "Listening to programmers — Taxonomies and characteristics of comments in operating system code," Proc. 2009 IEEE 31st International Conference on Software Engineering, Vancouver, Canada, pp. 331-341, May 16-24, 2019.

[7] D. Haouari, H. Sahraoui, P. Langlais, "How Good is Your Comment? A Study of Comments in Java Programs," Proc. 2011 International Symposium on Empirical Software Engineering and Measurement, Banff, Canada, pp. 137-146, Sept. 22-23, 2011.

[8] D. Steidl, B. Hummer, E. Juergens, "Quality analysis of source code comments," Proc. 2013 21st International Conference on Program Comprehension, San Francisco, USA, pp. 83-92, May 20-21, 2013.

[9] L. Pascarella, "Classifying code comments in Java mobile applications," Proc. 2018 IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems, Gothenburg, Sweden, pp. 39-40, May 27-June 3, 2018.

[10] L. Pascarella, A. Bacchelli, "Classifying Code Comments in Java Open-Source Software Systems," Proc. 2017 IEEE/ACM 14th International Conference on Mining Software Repositories, Buenos Aires, Argentina, pp. 227-237, May 20-21, 2017.

[11] L. Pascarella, M. Bruntink, A. Bacchelli, "Classifying code comments in Java software systems," *Empirical Software Engineering*, vol. *24*, no. *3*, pp. *1499-1537*, Jun 2019.

[12] Y. Shinyama, Y. Arahori, K. Gondow, "Analyzing Code Comments to Boost Program Comprehension," Proc. 2018 25th Asia-Pacific Software Engineering Conference , Nara, Japan, pp. 325-334, Dec. 4-7, 2018.

[13] J. Zhang, L. Xu, Y. Li, "Classifying Python Code Comments Based on Supervised Learning," Proc. 2018 15th International Conference on Web Information Systems and Applications (WISA), Taiyuan, China, pp. 39-47, Sept. 14-15, 2018.

[14] H. Husain, H.-H. Wu, T. Gazit, G. Miltiadis, A. M. Brockschmidt, "CodeSearchNet Challenge Evaluating the State of Semantic Code Search," Accessed: Dec. 29, 2021. [Online]. Available: https://github.com/github/CodeSearchNet.

[15] D. Jurgens, M. T. Pilehvar, R. Navigli, "SemEval-2014 Task 3: Cross-Level Semantic Similarity," Accessed: Dec. 29, 2021. [Online]. Available: http://alt.qcri.

[16] D. Jurgens, M. T. Pilehvar, R. Navigli, "Cross level semantic similarity: an evaluation framework for universal measures of similarity," *Language Resources and Evaluation*, vol. *50*, no. *1*, pp. *5-33*, Mar 2016.

 ISBN 978-86-7466-930-3

Primena ConvLSTM modela za predikciju optičke debljine aerosola

Uzahir R. Ramadani, Dušan P. Nikezić, Dušan S. Radivojević, Nikola Mirkov i Ivan Lazović

***Apstrakt*—Napravljen je ConvLSTM model sa ConvLSTM2D slojevima za predvđanje optičke debljine aerosola. Ulazni skup podataka čine satelitski snimci optičke debljine aerosola na 8 dana od 2000. godine. Kada je model obučen dato je predviđanje optičke debljine aerosola kao i evaluacija modela. Razvijen je i primenjen nad istim podacima i CNN LSTM model i dato je poređenje oba modela. Za evaluaciju je korišćena Srednja kvadratna greška (MSE) kao i Srednja apsolutna greška (MAE). ConvLSTM model je pokazao manju grešku i rezultati su pokazali da može da se koristi za predviđanje optičke debljine aerosola.**

***Ključne reči* — Optička debljina aerosola; ConvLSTM; satelitski snimci; CNN LSTM.**

I. Uvod

MAŠINSKO učenje kao deo veštačke inteligencije omogućuje napredovanje nauke u raznim oblastima. Savremeni tehnološki razvoj preko Interneta i velikih količina podataka omogućuje jednostavan pristup i korišćenje tih velikih baza podataka. Satelitski snimci kao skup podataka preko daljinske detekcije postali su dostupni naučnicima što omogućuje njihovo istraživanje. NASA pruža preko 50 različitih globalnih skupova podataka satelitskih snimaka. Jedan od tih skupova podataka je MODAL2_E_AER_OD koji predstavlja optičku debljinu aerosola snimanu svaki 8 dan od 2000. godine.

Odranije je poznato da aerosoli igraju važnu ulogu u oblikovanju vremena i klime, a novo istraživanje pokazalo je da i najsitnije čestice imaju nesrazmerno veliko dejstvo. Aerosoli iako mali su proizvođači snažnih kiša. Računarskim simulacijama naučnici su pokazali dejstvo tih čestica na olujne oblake. Iako su te čestice male, ima ih mnogo, i mogu da formiraju male kapi, na koje se kondenzuje isparavanje vode. To rezultuje oslobađanjem veće količine toplote koja osnažuje vazdušne mase. Više toplog vazduha ulazi u oblake i time stvara više leda, snega, munja i kiše.

Uzahir R. Ramadani - Institut za nuklearne nauke „Vinča“ - Institut od nacionalnog značaja za Republiku Srbiju, Univerzitet u Beogradu, Mike Petrovića Alasa bb., 1100 Beograd, Srbija (e-mail: uzahir@vin.bg.ac.rs).

Dušan P. Nikezić - Institut za nuklearne nauke „Vinča“ - Institut od nacionalnog značaja za Republiku Srbiju, Univerzitet u Beogradu, Mike Petrovića Alasa bb., 1100 Beograd, Srbija (e-mail: dusan@vin.bg.ac.rs).

Dušan S. Radivojević - Institut za nuklearne nauke „Vinča“ - Institut od nacionalnog značaja za Republiku Srbiju, Univerzitet u Beogradu, Mike Petrovića Alasa bb., 1100 Beograd, Srbija (e-mail: dusanr@vin.bg.ac.rs).

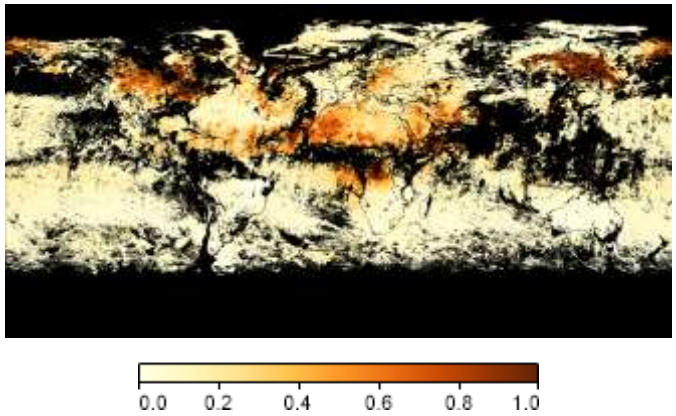
Nikola Mirkov - Institut za nuklearne nauke „Vinča“ - Institut od nacionalnog značaja za Republiku Srbiju, Univerzitet u Beogradu, Mike Petrovića Alasa bb., 1100 Beograd, Srbija (e-mail: nmirkov@vin.bg.ac.rs).

Ivan Lazović - Institut za nuklearne nauke „Vinča“ - Institut od nacionalnog značaja za Republiku Srbiju, Univerzitet u Beogradu, Mike Petrovića Alasa bb., 1100 Beograd, Srbija (e-mail: ivan.lazovic@vin.bg.ac.rs).

II. Metod

Spektroradiometar umerene rezolucije (MODIS - Moderate Resolution Imaging Spectroradiometer) je satelitski senzor koji se koristi za merenja zemlje i klime. MODIS instrumenti hvataju podatke u 36 spektralnih opsega u talasnim dužinama od 0.4 mm do 14.4 mm i u različitim prostornim rezolucijama (2 opsega na 250 m, 5 opsega na 500 m i 29 opsega na 1 km). Sa svojom visokom prostornom rezolucijom, ali niskom vremenskom rezolucijom, MODIS podaci su korisni za praćenje promena u globalnoj koncentraciji aerosola tokom vremena.

NASA skupovi podataka dostupni su kao RGB i kao PNG fajlovi i u rezoluciji 3600 x 1800 piksela. Sl. 1 prikazuje jedan snimak iz skupa podataka MODAL2_E_AER_OD koji predstavlja globalni AOT (aerosol optical thickness) na svakih 8 dana [12, 13] što predstavlja 993 slike do 14.09.2021.



Sl. 1. Snimak od 2021-07-12 globalnog AOT.

Optička debljina manja od 0.1 (bledožuta) ukazuje na kristalno čisto nebo sa maksimalnom vidljivošću, dok vrednost od 1 (crvenkasto smeđa) ukazuje na veoma maglovite uslove.

Preko MODIS-a na NASA-inim satelitima Terra i Aqua prati se AOT kao i distribucija aerosola, u većem delu sveta (okeani i vlažni delovi kontinenata) na dnevnom nivou, svakih 8 dana i mesečno. MODIS se koristi za praćenje masene koncentracije aerosola, optičkih svojstava i zračenja. MODIS-ove informacije o aerosolu se koriste za proučavanje klimatologije aerosola, za praćenje izvora i ponora specifičnih tipova aerosola (kao što su sulfati i drugi industrijski/urbani

 ISBN 978-86-7466-930-3

aerosol i aerosol sagorevanja biomase), i služe kao ulazni podaci za klimatsko modeliranje [7]. MODIS-ove informacije o aerosolu mogu se koristiti za predviđanje PM [11].

Satelitski snimci MODAL2_E_AER_OD predstavljaju ulazne podatke u modelu mašinskog učenja ConvLSTM. Tradicionalne metode predviđanja vremenskih serija fokusiraju se na univarijantne podatke sa linearnim odnosima i fiksnom i ručno dijagnostikovanom vremenskom zavisnošću. Neuronske mreže dodaju mogućnost učenja potencijalno nelinearnih odnosa sa proizvoljno definisanim, ali fiksnim brojem ulaza i izlaza koji podržavaju multivarijantno i višestepeno predviđanje.

Duboko učenje (DL – Deep Learning) je deo ML zasnovan na veštačkim neuronskim mrežama (ANN - Artificial Neural Networks). U DL konvoluciona neuronska mreža (CNN - Convolutional Neural Network) je klasa ANN, koja se najčešće primenjuje za analizu vizuelnih slika. U CNN-u ulaz je tenzor sa oblikom (broj ulaza) x (visina) x (širina) x (kanali). [17 , 18]. Jedna od primena CNN-a je izdvajanje prostornih informacija iz slika.

LSTM - Long Short-Term Memory je tip Rekurentne neuronske mreže (RNN - Recurrent Neural Networks) dizajniran tako da nema problem sa nestajućim gradijentom. LSTM mreže imaju povratne veze koje daju neuronima mogućnost odluke na temelju ne samo trenutne već i prethodnih vrednosti [19].

Prostorno-vremensko predviđanje može da se uradi CNN i LSTM, gde CNN (Convolution2D) služi za ekstrahovanje prostornih karakteristika, dok se LSTM koristi za otkrivanje korelacija tokom vremena [20]. Međutim, slaganjem ovih vrsta slojeva, korelacija između prostornih i vremenskih karakteristika možda neće biti pravilno ekstrahovana. Rešenje je mrežna struktura sposobna da uhvati prostorno-vremenske korelacije ConvLSTM [21]. U Kerasu je napravljena klasa (sloj) ConvLSTM2D, koja izračunava konvolucione operacije i u ulaznoj i u rekurentnoj transformaciji da bi istovremeno uhvatila prostorno-vremenske podatke. ConvLSTM2D je rekurentni sloj, baš kao i LSTM, ali interna množenja matrice se razmenjuju sa operacijama konvolucije. Kao rezultat toga, podaci koji teku kroz ConvLSTM ćelije zadržavaju ulaznu dimenziju umesto da budu samo 1D vektor [21 - 23]. Glavna razlika između ConvLSTM i LSTM je broj ulaznih dimenzija. Pošto su LSTM ulazni podaci jednodimenzionalni, nisu pogodni za podatke o prostornoj sekvenci kao što je video, satelitske slike itd. ConvLSTM je dizajniran za 3D podatke na ulazu.

III. CONVLSTM MODEL

Ulaz kod LSTM je 3D tenzor sa oblikom: broj uzoraka, vremenski korak, karakteristike (samples, time steps, features). Ulaz konvolucionog sloja je 4D tenzor sa oblikom: uzorci, redovi, kolone, kanali (samples, rows, cols, channels). Ulaz za ConvLSTM model je 5D tenzor sa oblikom: uzorci, vremenski koraci, redovi, kolone, kanali (samples, time steps, rows, cols, channels).

Izlaz ćelije LSTM zavisi od atributa return_sekuence. Kada je postavljeno na True, izlaz je sekvenca tokom vremena (jedan izlaz za svaki ulaz). U ovom slučaju, izlaz je 3D tenzor sa oblikom: uzorci, vremenski koraci, karakteristike. Kada je return_sekuences podešen na False, izlaz je poslednja vrednost niza, odnosno 2D tenzor sa oblikom (uzorci, karakteristike). Izlaz ConvLSTM sloja je kombinacija Convolution i LSTM izlaza. Baš kao i LSTM, ako je return_sekuences = True, onda vraća niz kao 5D tenzor sa oblikom: uzorci, vremenski koraci, redovi, kolone, filteri. Kada je return_sekuences = False, onda vraća samo poslednju vrednost niza kao 4D tenzor sa oblikom (uzorci, redovi, kolone, filteri).

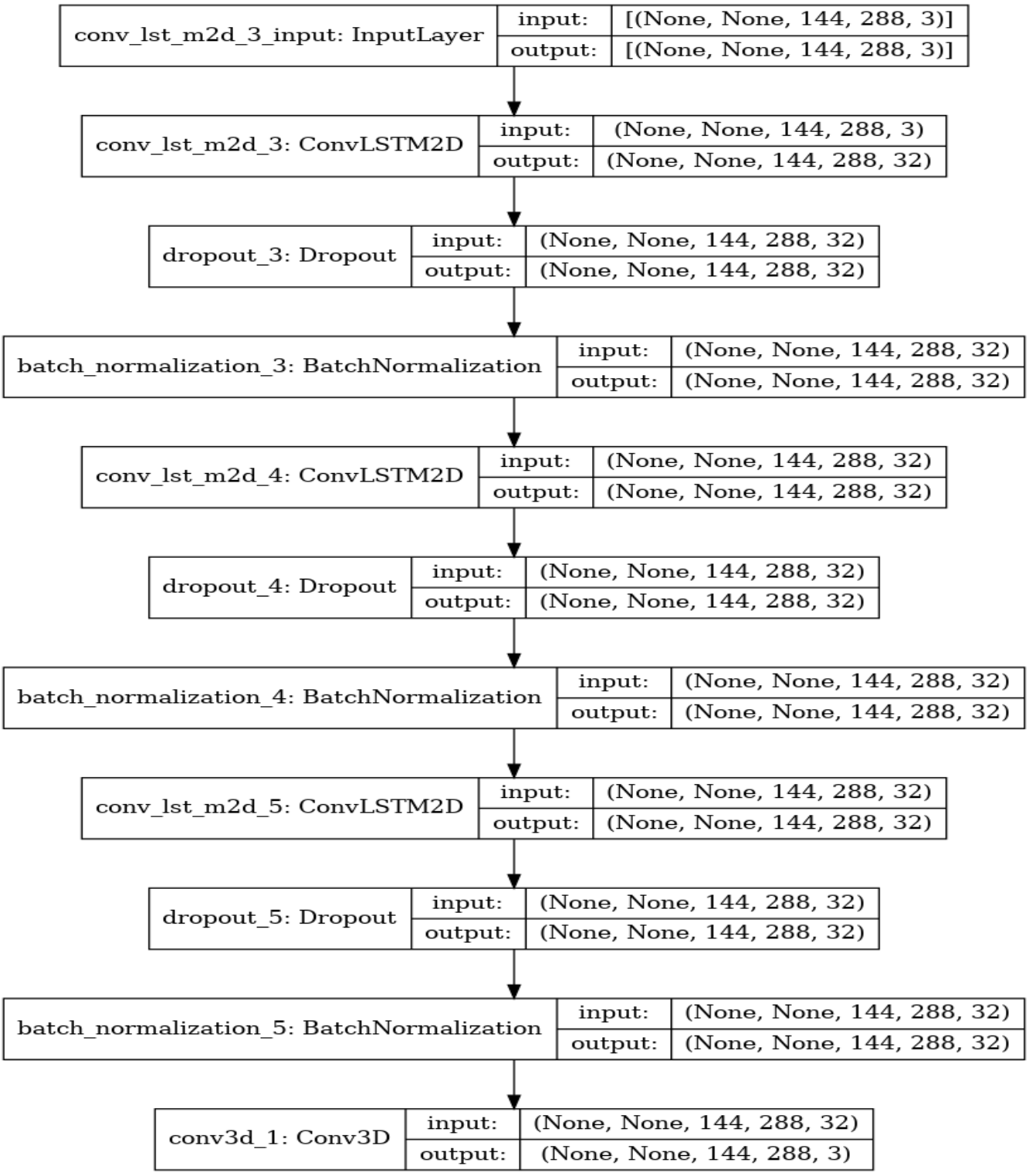
ML model koji se koristi u ovoj studiji sastoji se od 3 sloja ConvLSTM2D i krajnjeg sloja Conv3D kao izlaza. Conv3D sloj stvara konvoluciono jezgro koje je konvoluirano sa ulazom sloja da bi se proizveo tenzor izlaza, tj. visina, širina, kanal slike. ML model je napravljen u Kerasu.

CNN LSTM model ima prvi ulazni sloj koji prima niz od 10 slika rezolucije 144x288 piksela sa tri RGB kanala. Obzirom da Conv2D sloj može da funcioniše samo sa jedom slikom, njemu i u narednih 5 slojeva dodat je sloj TimeDistributed koji omogućava rad sa nizom ulaznih podataka. Drugi sloj Conv2D ima 32 filtera radi boljih performansi modela. BatchNormalization u trećem i petom sloju se koristi radi prevecije takozvane eksplozije loss funkcije i ravnomernijeg napredovanja odnosno normalizacije tokom procesa učenja, sprečavanjem prevelikog napredovanja pojedinih težina u odnosu na ostale u neuralnoj mreži odnosno svakom sloju zasebno. Četvrti sloj Conv2D sa tri filtera vraća format na tri RBG kanala. Šesti sloj Reshape vrši poravnanje 2D formata slike u 1D format zbog pripreme za sledeći sedmi LSTM memorijski sloj. Osmi Dropout sloj vrši prevenciju preteranog učenja kod koga se javlja veoma dobro procesiranje poznatih, ali loše procesiranje novih podataka. Deveti Dense sloj formira konačan broj piksela izlazne predikovane slike koji se u desetom Reshape sloju transformniše u odgovarajući format.

Skup ulaznih podataka bio je MODAL2_E_AER_OD, skup satelitskih slika od 18.02.2000. do 14.09.2021. na svakih 8 dana. Odnos train/test bio je 70/30, 80/20 i 90/10. Optimum je postignut podelom 80/20. Najbolje rezultate dala je funkcija aktivacije 'hard_sigmoid' za Conv3D sloj i optimizator 'adam' (learning_rate=0,001). Problem prediktivnog modeliranja regresije uključuje predviđanje veličine realne vrednosti, tako da je za funkciju gubitka korišćen 'mse', a za metriku RootMeanSkuaredError (RMSE). Sl. 2 prikazuje korišćeni ConvLSTM model (slika je dobijena komandom: plot_model(model, to_file='model.png', show_shapes=True, show_layer_names=True)).

https://www.kaggle.com/code/dusan75/convlstm-notebook09a46b0eb3

 ISBN 978-86-7466-930-3



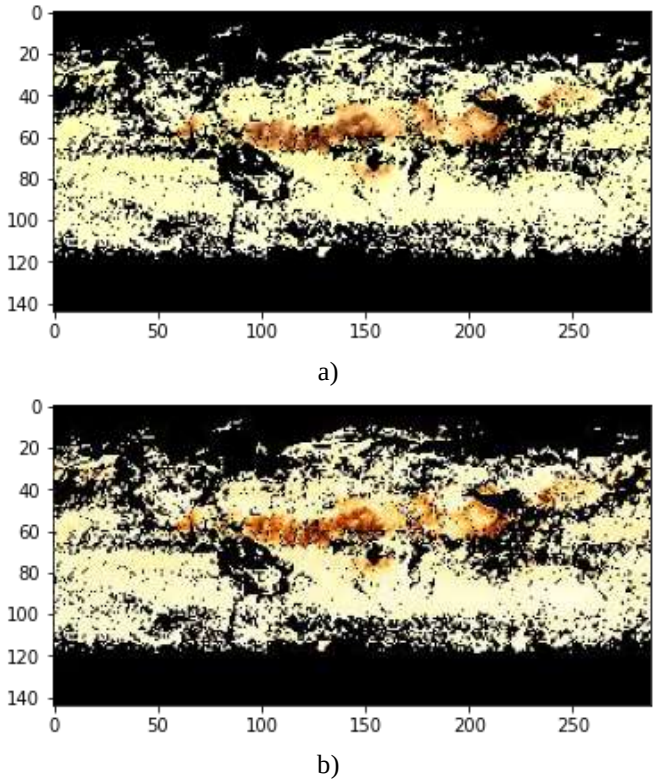
Sl. 2. Prikaz ConvLSTM modela.

Nakon obuke ML modela, urađeno je predviđanje prve slike iz skupa test podataka i upoređeno je sa originalnom prvom slikom iz skupa podataka testa.

IV. Rezultati i diskusija

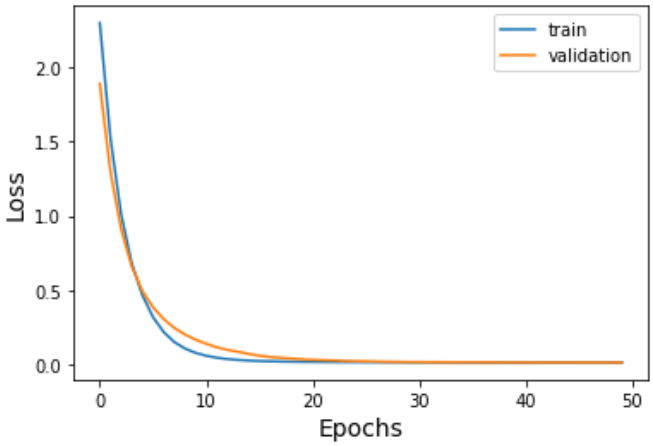
Sa ConvLSTM modelom je predikovana prva slika iz test skupa podataka, Sl. 3a. Sl. 3b prikazuje originalnu prvu sliku iz test skupa podataka.

 ISBN 978-86-7466-930-3



Sl. 3. Globalni AOT a) predikovani; b) realni.

Validacija ML modela je zasnovana na metrici evaluacije, tj. mse 0,0116 (mean squared error). Ovi rezultati dokazuju da se predloženi ML model može koristiti za AOT prognoze. Sl. 4 prikazuje grafikon gubitka tokom 50 epoha obuke.

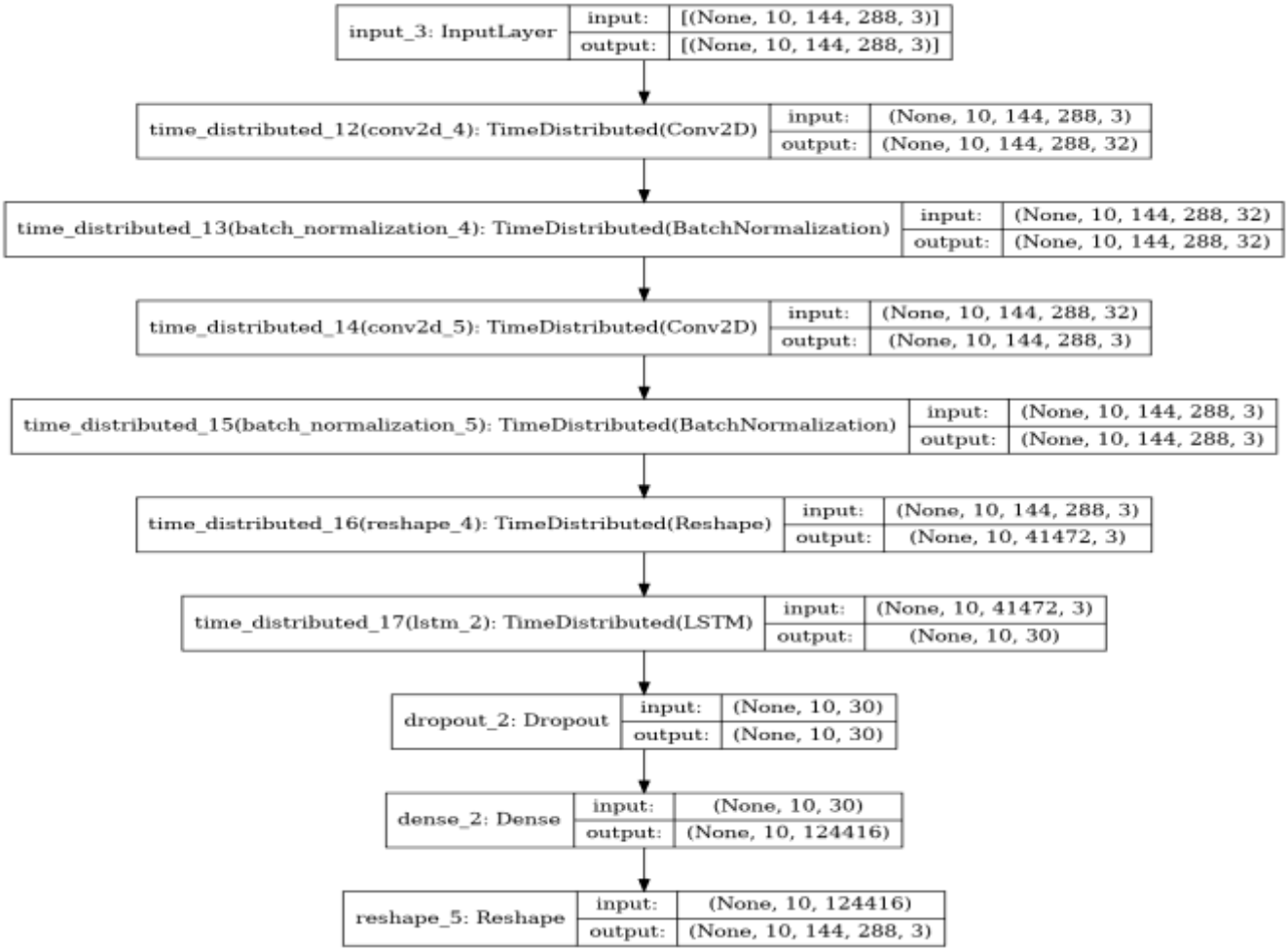


Sl. 4. Grafikon gubitka u zavisnosti broja epoha.

Pre ConvLSTM2D za prostorno-vremensku zavisnost i predikciju se koristio CNN+LSTM model [24]. Za poređenje ConvLSTM i CNN LSTM razvijen je novi CNN LSTM model. Sl. 5 prikazuje novi model CNN LSTM.

Metrika evaluacije bila je za mse 0,1117 i u poređenju sa ConvLSTM modelom (mse 0,0116) je veća vrednost. Niže vrednosti mse ukazuju na manju grešku.

 ISBN 978-86-7466-930-3



Sl. 5. Prikaz CNN LSTM modela.

Primena Mean Absolute Percent Error metrike na konkretnom primeru nije primenljiva, obzirom na činjenicu da se među y-true vrednotima nalazi i vrednost 0 zbog koje metoda ne daje vrednosti u predviđenom opsegu od 0 do 1. Testiranje modela nalazi se na adresi:

https://www.kaggle.com/code/dusan75/test-of-convlstm-notebook09a46b0eb3?scriptVersionId=94940211

Za bolje poređenje dva modela urađena je sledeća statistika. Srednja apsolutna greška (MAE - Mean Absolute Error) koristi se za predviđanja u segmentu od 9 frejmova po slici, Tabela 1.

TABELA I
PROSEČNA GREŠKA PREDVIĐANJA U SEGMENTU OD 9 FREJMOVA PO SLICI

ConvLSTM	CNN LSTM
0.02200718	0.24474642
0.02213584	0.2462145
0.02251104	0.24545886
0.02263835	0.2449535
0.0227564	0.24526556
0.02259156	0.24504275
0.02232519	0.24720876
0.02259231	0.2456902
0.02278301	0.24560421

Iz dobijenih rezultata može se zaključiti da je ConvLSTM model zahtevao dobro strukturirane ulazne podatke, pravi izbor i optimalno podešene hiperparametre modela pre nego što bi mogao da se koristi za pouzdana AOT predviđanja.

V. ZAKLJUČAK

Aerosoli su jedan od najvećih izvora nesigurnosti u modeliranju klime. Zračenje aerosolima može objasniti razliku između posmatranih i modelovanih trendova prosečne globalne temperature. U stvari, interakcija sa sunčevim i zemaljskim zračenjem pomoću aerosola remeti radijativni budžet rasejanjem i apsorpcijom sunčeve svetlosti. Mnoge nedavne studije pokazuju važnost uključivanja aerosola u klimatske modele za posmatranje i merenje ljudskog uticaja na atmosfersku hemiju i klimatske promene.

Ova studija je istraživala mogućnost ConvLSTM modela za predviđanje globalnog AOT-a sa MODIS satelitskih snimaka. Sloj ConvLSTM2D u Kerasu spaja prostornu i vremensku komponentu i omogućava predikciju. Satelitski snimci pomažu u sagledavanju globalnog transporta zagađivača.

Modeli mašinskog učenja kreirani su od strane autora, tj. originalne i inovativne su strukture, pa se i rezultati treniranja

 ISBN 978-86-7466-930-3

i predviđanja modela, pored originalne strukture, mogu uzeti kao doprinos zajednici radi poređenja postignutih rezultata.

ZAHVALNICA

Ovaj rad je realizovan u okviru istraživačke teme 1002205 koga finansira Ministarstvo prosvete, nauke i tehnološkog razvoja Republike Srbije.

LITERATURA

[1] C.Q. Lin, G. Liu, A.K.H. Lau, Y. Li, C.C. Li, J.C.H. Fung, X.Q. Lao, High-resolution satellite remote sensing of provincial PM2.5 trends in China from 2001 to 2015, Atmospheric Environment, Volume 180, 2018, Pages 110-116, ISSN 1352-2310, https://doi.org/10.1016/j.atmosenv.2018.02.045.

[2] Zhaoxi Wang, Yang Liu, Mu Hu, Xiaochuan Pan, Jing Shi, Feng Chen, Kebin He, Petros Koutrakis, David C. Christiani, Acute health impacts of airborne particles estimated from satellite remote sensing, Environment International, Volume 51, 2013, Pages 150-159, ISSN 0160-4120, https://doi.org/10.1016/j.envint.2012.10.011.

[3] Shuaiyi Shi, Tianhai Cheng, Xingfa Gu, Hong Guo, Yu Wu, Ying Wang, Fangwen Bao, Xin Zuo, Probing the dynamic characteristics of aerosol originated from South Asia biomass burning using POLDER/GRASP satellite data with relevant accessory technique design, Environment International, Volume 145, 2020, 106097, ISSN 0160-4120, https://doi.org/10.1016/j.envint.2020.106097.

[4] Xiaoli Wei, Ni-Bin Chang, Kaixu Bai & Wei Gao (2020) Satellite remote sensing of aerosol optical depth: advances, challenges, and perspectives, Critical Reviews in Environmental Science and Technology, 50:16, 1640-1725, DOI: 10.1080/10643389.2019.1665944.

[5] Filonchyk, M., Yan, H., Zhang, Z. et al. Combined use of satellite and surface observations to study aerosol optical depth in different regions of China. Sci Rep 9, 6174 (2019). https://doi.org/10.1038/s41598-019-42466-6.

[6] Ian Colbeck, Mihalis Lazaridis, Aerosol Science: Technology and Applications, ISBN: 978-1-119-97792-6 December 2013, Wiley, https://doi.org/10.1002/9781118682555.ch1.

[7] https://neo.gsfc.nasa.gov/

[8] Dušan P. Nikezić, Zoran J. Gršić, Dragan M. Dramlić, Stefan D. Dramlić, Boris B. Lončar, Slavko D. Dimović, Modeling air concentration of fly ash in Belgrade, emitted from thermal power plants TNTA and TNTB, Process Safety and Environmental Protection, Volume 106, 2017, Pages 274-283, ISSN 0957-5820, https://doi.org/10.1016/j.psep.2016.06.009.

[9] You, W.; Zang, Z.; Zhang, L.; Li, Y.; Pan, X.; Wang, W., National-Scale Estimates of Ground-Level PM2.5 Concentration in China Using Geographically Weighted Regression Based on 3 km Resolution MODIS AOD. Remote Sens. 2016, 8, 184. https://doi.org/10.3390/rs8030184.

[10] Tov Elperin, Andrew Fominykh, Itzhak Katra, Boris Krasovitov, Modeling of gas adsorption by aerosol plumes emitted from industrial sources, Process Safety and Environmental Protection, Volume 111, 2017, Pages 375-387, ISSN 0957-5820, https://doi.org/10.1016/j.psep.2017.06.022.

[11] Naresh Kumar, Allen Chu, Andrew Foster, An empirical relationship between PM2.5 and aerosol optical depth in Delhi Metropolitan, Atmospheric Environment, Volume 41, Issue 21, 2007, Pages 4492-4503, ISSN 1352-2310, https://doi.org/10.1016/j.atmosenv.2007.01.046.

[12] https://neo.gsfc.nasa.gov/archive/rgb/

[13] https://neo.gsfc.nasa.gov/archive/rgb/MODAL2_E_AER_OD/

[14] Song Tang, Yixin Mao, Rachael M. Jones, Qiyue Tan, John S. Ji, Na Li, Jin Shen, Yuebin Lv, Lijun Pan, Pei Ding, Xiaochen Wang, Youbin Wang, C. Raina MacIntyre, Xiaoming Shi, Aerosol transmission of SARS-CoV-2? Evidence, prevention and control, Environment International, Volume 144, 2020, 106039, ISSN 0160-4120, https://doi.org/10.1016/j.envint.2020.106039.

[15] Maria A. Zoran, Roxana S. Savastru, Dan M. Savastru, Marina N. Tautan, Laurentiu A. Baschir, Daniel V. Tenciu, Exploring the linkage between seasonality of environmental factors and COVID-19 waves in Madrid, Spain, Process Safety and Environmental Protection, Volume 152, 2021, Pages 583-600, ISSN 0957-5820, https://doi.org/10.1016/j.psep.2021.06.043.

[16] Eleftheriadis, K., Gini, M.I., Diapouli, E. et al,. Aerosol microphysics and chemistry reveal the COVID19 lockdown impact on urban air quality., Sci Rep 11, 14477 (2021). https://doi.org/10.1038/s41598-021-93650-6.

[17] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, N.I. Chervyakov, Application of the residue number system to reduce hardware costs of the convolutional neural network implementation, Mathematics and Computers in Simulation, Volume 177, 2020, Pages 232-243, ISSN 0378-4754, https://doi.org/10.1016/j.matcom.2020.04.031.

[18] Radhesyam Vaddi, Prabukumar Manoharan, Hyperspectral image classification using CNN with spectral and spatial features integration, Infrared Physics & Technology, Volume 107, 2020, 103296, ISSN 1350-4495, https://doi.org/10.1016/j.infrared.2020.103296.

[19] Sepp Hochreiter, Jürgen Schmidhuber; Long Short-Term Memory. Neural Comput 1997; 9 (8): 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735.

[20] Weyn, J. A., Durran, D. R., & Caruana, R. (2020). Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. Journal of Advances in Modeling Earth Systems, 12, e2020MS002109. https://doi.org/10.1029/2020MS002109.

[21] X. Shi, Z. Chen, H. Wang and D. Yeung, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting", Proc. Neural Inf. Process. Syst. (NIPS), pp. 802-810, 2015.

[22] J. Donahue et al., "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp. 677-691, 1 April 2017, doi: 10.1109/TPAMI.2016.2599174.

[23] W. Hu, H. Li, L. Pan, W. Li, R. Tao and Q. Du, "Spatial–Spectral Feature Extraction via Deep ConvLSTM Neural Networks for Hyperspectral Image Classification," in IEEE Transactions on Geoscience and Remote Sensing, vol. 58, no. 6, pp. 4237-4250, June 2020, doi: 10.1109/TGRS.2019.2961947.

[24] X. Ding, L. Feng, Y. Zou and G. Zhang, "Deep Learning Aided Spectrum Prediction for Satellite Communication Systems," in IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 16314-16319, Dec. 2020, doi: 10.1109/TVT.2020.3043837.

[25] Christof G. Beer, Johannes Hendricks, Mattia Righi, Bernd Heinold, Ina Tegen, Silke Groß, Daniel Sauer, Adrian Walser, Bernadett Weinzierl, Modelling mineral dust emissions and atmospheric dispersion with MADE3 in EMAC v2.54, Geosci. Model Dev., 13, 4287–4303, 2020., doi.org/10.5194/gmd-13-4287-2020.

[26] https://aeronet.gsfc.nasa.gov/

[27] Weyn, J. A., Durran, D. R., Caruana, R., & Cresswell-Clay, N. (2021). Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. Journal of Advances in Modeling Earth Systems, 13, e2021MS002502. https://doi.org/10.1029/2021MS002502.

ABSTRACT

ConvLSTM model was developed with ConvLSTM2D layers for predicting aerosol optical thickness. The input dataset was satellite images of aerosol optical thickness on 8-day beginning from 2000. When the model was trained, aerosol optical thickness prediction as well as model evaluation was given. The CNN LSTM model was developed and applied over the same dataset and a comparison of both models is given. Mean square error (MSE) as well as Mean absolute error (MAE) were used for evaluation. The ConvLSTM model showed less error and the results showed that it could be used to predict the aerosol optical thickness.

ConvLSTM Application for Prediction Aerosol Optical Thickness

Uzahir R. Ramadani, Dušan P. Nikezić, Dušan S. Radivojević, Nikola Mirkov i Ivan Lazović

Primena veštačke inteligencije na terminal za daljinsko upravljanje stanice za punjenje električnih vozila koja se napaja iz obnovljivih izvora električne energije

Jovan Vujasinović i Goran Savić

***Apstrakt*— U ovom radu je opisana primena veštačke inteligencije na terminal za daljinsko upravljanje stanice za punjenje električnih vozila, koja se napaja iz obnovljivih izvora električne energije. Kako terminal omogućava daljinsko upravljanje punjačima električnih vozila, pametnim baterijama, pametnim brojilima, fiskalnim kasama i eventualno daljinsko upravljanje obnovljivim izvorom električne energije i drugim uređajima u okviru objekta, neophodno je definisati i razraditi odgovarajući algoritam upravljanja radom terminala. U ovom radu je razmatrana realizacija tog upravljanja primenom veštačke inteligencije. Na ovaj način ovakve stanice za punjenje električnih vozila postaju potpuno autonomne u svom radu, i daju optimalne rezultate, što podiže njihovu dostupnost korisnicima električnih vozila. To potencijalno podstiče povećanje obima korišćenja električnih vozila za koje se energija obezbeđuje iz obnovljivih izvora, čime se smanjuje stepen zagađenja vazduha kao i negativni efekti koje ono sa sobom donosi.**

***Ključne reči*— Veštačka inteligencija; punjači električnih vozila; obnovljivi izvori energije.**

I. Uvod

Povećanje broja proizvedenih i korišćenih električnih vozila, ima za cilj smanjenje zagađenja životne sredine i štetnih efekata koje ono sa sobom nosi, a koji su posledica emisije štetnih gasova vozila na dizel i benzinski pogon. Međutim, preduslov da bi upotreba vozila na električni pogon zaista doprinela smanjenju zagađenja životne sredine, je da električna energija koja se koristi za punjenje električnih vozila bude proizvedena iz izvora koji ne zagađuju životnu sredinu, tj. iz obnovljivih izvora električne energije. To dovodi i do neminovnog razvoja infrastrukture za punjenje električnih vozila, što se ogleda i u stalnom povećanju broja stanica za punjenje električnih vozila koje se napajaju iz obnovljivih izvora električne energije [1].

Povećanje dostupnosti stanica za punjenje električnih vozila, kako korisnicima električnih vozila, tako i operaterima elektrodistribucije, snabdevača i poreske uprave, kao i vlasnicima i korisnicima same stanice, ostvaruje se integracijom u jedan veći sistem, zahvaljujući kojem se postiže ušteda vremena i novca, i efikasnija upotreba elektrodistributivne mreže. Da bi se ostvarile funkcionalnosti tog sistema, koje su od značaja svim korisnicima, realizovano je daljinsko upravljanje stanice za punjenje električnih vozila. Ključni uređaj koji omogućava pomenuto daljinsko upravljanje je terminal za daljinsko upravljanje stanice za punjenje električnih vozila koja se napaja iz obnovljivih izvora električne energije. Proces upravljanja je zasnovan na primeni naprednih algoritama veštačke inteligencije, čija je primena na terminal za daljinsko upravljanje stanice za punjenje električnih vozila prezentovana u ovom radu.

U Sekciji II ovog rada je opisana arhitektura sistema za daljinsko upravljanje stanice za punjenje električnih vozila koja se napaja iz obnovljivih izvora električne energije, u Sekciji III je data kategorizacija stanica i terminala, u Sekciji IV je predstavljena veštačka inteligencija, u Sekciji V je opisano mašinsko učenje, u Sekciji VI je razmatrana primena veštačke inteligencije na terminal, dok Sekcija VII predstavlja kratak zaključak.

II. Arhitektura sistema

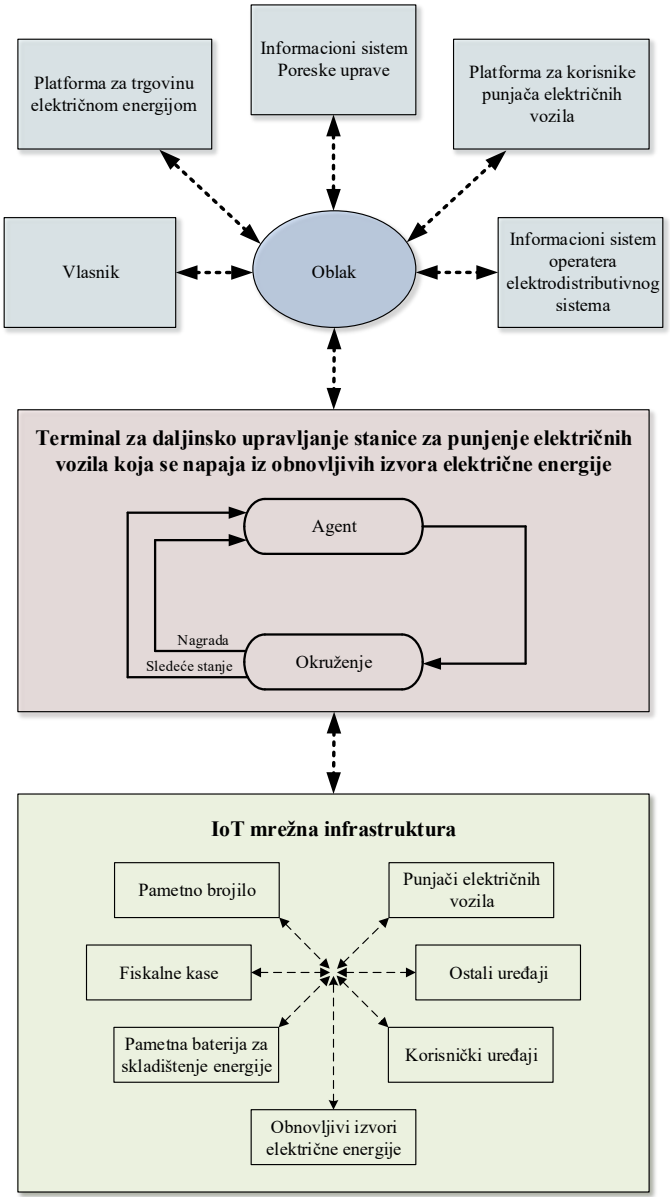
Blok šema arhitekture sistema za daljinsko upravljanje stanice za punjenje električnih vozila koja se napaja iz obnovljivih izvora električne energije [2] je prikazana na Sl. 1. Terminal za daljinsko upravljanje stanice za punjenje električnih vozila, koji predstavlja osnovnu komponentu sistema, je povezan sa punjačima električnih vozila, pametnom baterijom za skladištenje energije, obnovljivim izvorima električne energije, pametnim brojilom, fiskalnim kasama, korisničkim uređajima i ostalim uređajima, pomoću IoT mrežne infrastrukture. Pored toga, terminal za daljinsko upravljanje stanice za punjenje električnih vozila je povezan i sa oblakom preko internet veze, čime je omogućeno da se podaci dobijeni od punjača električnih vozila, pametne baterije za skladištenje energije, obnovljivih izvora električne energije, pametnog brojila i fiskalnih kasa, prate, skladište i obrađuju. Pristup tim podacima je omogućen sledećim platformama: platformi za korisnike punjača električnih vozila, platformi za trgovinu električnom energijom, informacionom sistemu operatera elektrodistributivnog sistema i informacionom sistemu Poreske uprave. Platforma za korisnike punjača električnih vozila omogućava vlasnicima električnih vozila da dobiju sve potrebne informacije o punjačima električnih vozila. Platforma za trgovinu

Jovan Vujasinović – Elektrotehnički fakultet, Univerzitet u Beogradu, Bulevar Kralja Aleksandra 73, 11120 Beograd , Srbija (e-mail: jovan.vujasinovic@vfholding.rs).

Goran Savić – Elektrotehnički fakultet, Univerzitet u Beogradu, Bulevar Kralja Aleksandra 73, 11120 Beograd, Srbija (e-mail: gsavic@etf.rs).

 ISBN 978-86-7466-930-3

električnom energijom omogućava obavljanje trgovine električnom energijom koja je raspoloživa u sistemu. Takođe, pristup podacima u oblaku imaju i vlasnici stanica za punjenje električnih vozila. Ti podaci se procesiraju primenom algoritama veštačke inteligencije, čime se ostvaruje efikasna upotreba distributivne mreže uz korišćenje inovativnih pametnih energetskih usluga, kao i značajne uštede u celom sistemu.



Sl. 1. Blok šema arhitekture sistema za daljinsko upravljanje stanice za punjenje električnih vozila koja se napaja iz obnovljivih izvora energije.

III. KATEGORIJE STANICA I TERMINALA

Stanice za punjenje električnih vozila koje se napajaju iz obnovljivih izvora električne energije možemo kategorisati na rezidencijalne, komercijalne i industrijske stanice [3]. Isto tako, prema tipu stanice kod koje se primenjuje, terminale možemo podeliti na light, standard i extended terminale [4].

Rezidencijalne stanice su namenjene za kućnu upotrebu, samo za potrebe domaćinstva. U okviru ove stanice očekuje se instalacija 1-2 punjača za električna vozila. Potrebe ovakvih stanica mogu se pokriti samo korišćenjem solarnih obnovljivih izvora. Kod ovih stanica koristi se light terminal koji ima najjednostavniji hardver i softver, a ceo sistem za upravljanje ovakvom stanicom je vrlo sličan sa sistemom za upravljanje energijom u domaćinstvima.

Komercijalne stanice su male i srednje stanice za punjenje električnih vozila koje imaju visoku frekvenciju punjenja vozila. Obično su to stanice namenjene prodaji električne energije vlasnicima električnih vozila, kao što su srednje stanice koje će eventualno zameniti postojeće benzinske pumpe sa 10-20 punjača, manje stanice u domaćinstvima sa 1-2 punjača za ovu namenu, kao i kao stanice na manjim parkiralištima sa do 20 punjača. Potrebe ovakvih stanica ne mogu se pokriti korišćenjem samo solarnih obnovljivih izvora. Gde geografsko područje dozvoljava, ima smisla koristiti vetrogenerator. Moguća je upotreba generatora na biomasu, ako prostorni kapaciteti to dozvoljavaju, što bi prvo mogao biti slučaj u ruralnim područjima. U urbanim sredinama, u mnogim mestima jedini način da se nadoknadi nedostatak električne energije je preuzimanje energije iz elektro distributivne mreže. Kod ovih stanica koristi se standard terminal koji ima složeniji hardver i softver, a ceo sistem za upravljanje ovakvom stanicom je složeniji od sistema za upravljanje energijom u domaćinstvima..

Industrijske stanice su velike stanice za punjenje električnih vozila namenjene za opsluživanje flote električnih vozila od nekoliko stotina komada. U ovu grupu stanica spadaju stanice kod proizvođača električnih vozila, stanice kod velikih drumskih prevoznika i stanice na masovnim parkiralištima kao što su garaže, tržni centri itd. Slično komercijalnim stanicama, potrebe ovakvih stanica ne mogu se pokriti korišćenjem samo obnovljivih izvora električne energije, a često jedini način da se nadoknadi nedostajuća električna energija je preuzimanje iste iz elektro distributivne mreže. Kod ovih stanica koristi se extended terminal, koji ima najsloženiji hardver i softver.

IV. VEŠTAČKA INTELIGENCIJA

Veštačka inteligencija se može definisati kao sposobnost digitalnog računara ili kompjuterski kontrolisanog robota da obavlja zadatke koji se obično povezuju sa inteligentnim bićima [5]. Izraz se često primenjuje na projekat razvoja sistema obdarenih intelektualnim procesima karakterističnim za ljude, kao što je sposobnost rasuđivanja, otkrivanja značenja, generalizacije ili učenja iz prethodnog iskustva. Psiholozi generalno ne karakterišu ljudsku inteligenciju samo jednom osobinom, već kombinacijom mnogih različitih sposobnosti. Istraživanja u oblasti veštačke inteligencije su se uglavnom fokusirala na sledeće komponente inteligencije: učenje, rasuđivanje, rešavanje problema, percepciju i korišćenje jezika. Pored ovih osnovnih komponenti, mogu se naći istraživanja koja se bave i razvojem mašina koje imaju sledeće sposobnosti: planiranje, predstavljanje znanja, kretanje, socijalna inteligencija i opšta inteligencija.

Rasuđivanje znači izvođenje zaključaka koji odgovaraju situaciji. Zaključci se klasifikuju kao deduktivni ili induktivni.

 ISBN 978-86-7466-930-3

Najznačajnija razlika između ovih oblika rezonovanja je u tome što u deduktivnom slučaju istinitost premisa garantuje istinitost zaključka, dok u induktivnom slučaju istinitost premise daje podršku zaključku bez da daje apsolutnu sigurnost. Postignut je značajan uspeh u programiranju računara za izvođenje zaključaka, posebno deduktivnih zaključaka. Međutim, pravo rezonovanje uključuje više od samog izvođenja zaključaka; podrazumeva izvođenje zaključaka relevantnih za rešenje konkretnog zadatka ili situacije. Ovo je jedan od najtežih problema sa kojima se veštačka inteligencija suočava.

Rešavanje problema, posebno u veštačkoj inteligenciji, može se okarakterisati kao sistematsko traženje niza mogućih radnji u cilju dostizanja nekog unapred definisanog cilja ili rešenja. Metode rešavanja problema dele se na metode posebne namene i metode opšte namene. Metoda posebne namene je skrojena za određeni problem i često koristi veoma specifične karakteristike situacije u kojoj je problem ugrađen. Nasuprot tome, metoda opšte namene je primenljiva na širok spektar problema. Mnogi različiti problemi rešeni su programima veštačke inteligencije. Neki primeri su pronalaženje pobedničkog poteza (ili niza poteza) u igri na ploči, osmišljavanje matematičkih dokaza i manipulacija „virtuelnim objektima“ u kompjuterski generisanom svetu.

Percepciju možemo opisati kao skeniranje okoline pomoću različitih čulnih organa, stvarnih ili veštačkih, i razlaganje scene na zasebne objekte u različitim prostornim odnosima. Analiza je komplikovana činjenicom da objekat može izgledati drugačije u zavisnosti od ugla iz kojeg se posmatra, pravca i intenziteta osvetljenja u sceni i koliko je objekat u kontrastu sa okolnim poljem. Mašinsku percepciju [6] možemo definisati kao sposobnost da se koriste ulazni signali sa senzora (kao što su kamere, mikrofoni, bežični signali, sonar, radar i taktilni senzori) da bi se zaključili aspekti sveta, odnosno okruženja. Aplikacije uključuju i prepoznavanje govora, prepoznavanje lica i prepoznavanje objekata. Trenutno je veštačka percepcija dovoljno napredna da omogući optičkim senzorima da identifikuju pojedince, autonomnim vozilima da voze umerenom brzinom na otvorenom putu i robotima da lutaju kroz zgrade skupljajući prazne limenke soda.

Korišćenje jezika omogućava mašinama da čitaju i razumeju ljudski jezik. Dovoljno moćan sistem za obradu ljudskog jezika bi omogućio korisničke interfejse na ljudskom jeziku i sticanje znanja direktno iz izvora pisanih od strane ljudi, kao što su tekstovi i vesti. Neke jednostavne primene korišćenja jezika uključuju pronalaženje informacija, odgovaranje na pitanja i mašinsko prevođenje [7].

Planiranje omogućava računaru da predviđa kako će ga njegove akcije promeniti i da donosi odluke koji maksimiziraju korist iz dostupnih izbora [8]. U klasičnim problemima planiranja, računar može pretpostaviti da je on jedini sistem koji deluje na svetu, dozvoljavajući sebi da bude siguran u posledice svojih akcija. Međutim, ako računar nije jedini akter, onda to zahteva da agent razmišlja u neizvesnosti, i kontinuirano ponovo procenjuje svoje okruženje i prilagođava se. Planiranje sa više računara koristi saradnju i konkurenciju mnogih računara za postizanje zadatog cilja.

Predstavljanje znanja i inženjering znanja [6] omogućavaju programima veštačke inteligencije da inteligentno odgovaraju na pitanja i donose zaključke o činjenicama iz stvarnog sveta. Reprezentacija „onog što postoji“ je ontologija: skup objekata, relacija, koncepata i svojstava formalno opisanih tako da softverski programi mogu da ih tumače. Najopštije ontologije se nazivaju gornje ontologije, koje pokušavaju da obezbede osnovu za sva druga znanja i deluju kao posrednici između ontologija domena koje pokrivaju specifično znanje o određenom domenu znanja.

Kretanje se u velikoj meri koristi u robotici [6]. Lokalizacija je način na koji robot zna svoju lokaciju i mapira svoje okruženje. Kada se dobije malo, statičko i vidljivo okruženje, ovo je lako; međutim, dinamična okruženja, kao što je (u endoskopiji) unutrašnjost tela koje diše, predstavljaju veći izazov.

Razvoj sposobnosti socijalne inteligencije je interdisciplinarna oblast koji obuhvata sisteme koji prepoznaju, tumače, obrađuju ili simuliraju ljudska osećanja, emocije i raspoloženje [9]. Na primer, neki virtuelni asistenti su programirani da pričaju razgovorno ili čak da se šale na duhovit način; to čini da izgledaju osetljiviji na emocionalnu dinamiku ljudske interakcije ili da na drugi način olakšaju interakciju između čoveka i računara.

Mašina sa opštom inteligencijom može rešiti širok spektar problema sa širinom i svestranošću sličnom ljudskoj inteligenciji. Postoji nekoliko konkurentnih ideja o tome kako razviti veštačku opštu inteligenciju.

V. MAŠINSKO UČENJE

Mašinsko učenje je disciplina veštačke inteligencije koja se bavi implementacijom kompjuterskih softvera koji su u stanju da samostalno uče [10]. Kako bi mogli da predviđaju ili donose odluke, algoritmi mašinskog učenja grade model zasnovan na uzorku podataka, poznatih kao podaci o obuci. Podskup mašinskog učenja je usko povezan sa statistikom, ali nije svako mašinsko učenje statističko učenje. Pretraga podataka, odnosno praksa analize velikih baza podataka u cilju generisanja novih informacija, je srodna oblast proučavanja, koja se fokusira na istraživačku analizu podataka.

Generalno, algoritmi mašinskog učenja se koriste za predviđanje ili klasifikaciju [11]. Na osnovu nekih ulaznih podataka, koji mogu biti označeni ili neoznačeni, algoritam će proizvesti procenu o obrascu u podacima. Zatim se vrši procena takvog predviđanja. Ako postoje poznati primeri, funkcija greške može da napravi poređenje da proceni tačnost predviđanja. Ako predviđanje može bolje da se uklopi u tačke podataka u skupu za obuku, onda se odgovarajući koeficijenti prilagođavaju da bi se smanjila neslaganja između poznatog primera i procene. Algoritam će ponoviti ovaj proces evaluacije i optimizacije, samostalno ažurirajući koeficijente dok se ne dostigne prag tačnosti.

Mašinsko učenje se može klasifikovati na: nadgledano, nenadgledano i polunadgledano učenje, kao i pojačano učenje. Učenje pod nadzorom, takođe poznato kao nadgledano mašinsko učenje, definisano je upotrebom označenih skupova podataka za obuku algoritama koji klasifikuju podatke ili tačno predviđaju ishode. Kako se ulazni podaci unose u

 ISBN 978-86-7466-930-3

model, on prilagođava svoje koeficijente dok se model ne uklopi na najbolji mogući način. Učenje bez nadzora, poznato i kao nenadgledano mašinsko učenje, koristi algoritme mašinskog učenja za analizu i grupisanje neoznačenih skupova podataka. Ovi algoritmi otkrivaju skrivene obrasce ili grupisanje podataka bez potrebe za ljudskom intervencijom. Njegova sposobnost da otkrije sličnosti i razlike u informacijama čini ga idealnim rešenjem za istraživačku analizu podataka, strategije unakrsne prodaje, segmentaciju kupaca, prepoznavanje slika i obrazaca. Polu-nadgledano učenje nudi optimalan odnos između učenja pod nadzorom i učenja bez nadzora. Tokom obuke, koristi manji skup označenih podataka da vodi klasifikaciju i izdvajanje karakteristika iz većeg, neoznačenog skupa podataka. Polu-nadgledano učenje može da reši problem nedostatka označenih podataka (ili nemogućnosti da priuštite da označite dovoljno podataka) za obuku algoritma za učenje pod nadzorom. Pojačano učenje je model mašinskog učenja ponašanja koji je sličan nadgledanom mašinskom učenju, ali algoritam nije obučen korišćenjem uzoraka podataka. Ovaj model uči koristeći pokušaje i greške. Niz uspešnih ishoda će biti pojačan kako bi se razvila najbolja preporuka ili politika za dati problem.

Neke metode koje se koriste u mašinskom učenju uključuju neuronske mreže, Bajesove mreže, linearnu regresiju, logističku regresiju, slučajnu šumu (drvo odlučivanja), mašinu vektora podrške (SVM), grupisanje k-srednjih vrednosti, metode verovatnoće klasterisanja i još mnogo toga. Metoda neuronske mreže imitira rad biološkog mozga.

VI. Primena veštačke inteligencije na terminal

U radu koji daje pregled stanja u oblasti sistema upravljanja energijom u domaćinstvu [12] može se uočiti da između ostalih postoje i grupe radova koji se bave primenom veštačke inteligencije u toj oblasti. Uvidom u to, možemo uočiti da se najviše primenjuje pojačano učenje. Ovo saznanje se takođe odnosi i na sistem za daljinsko upravljanje stanicom za punjenje električnih vozila, iz razloga što je *light* varijanta terminala namenjena za korišćenje kod rezidencijalne stanice za punjenje električnih vozila. Rezidencijalna stanica za punjenje električnih vozila predstavlja sistem upravljanja energijom u domaćinstvu. Pored rezidencijalne stanice, postoje i komercijalne i industrijske stanice, kod kojih se koriste dosta složenije *standard* i *extended* varijanta terminala. Ipak, zbog njihove sličnosti sa *light* varijantom terminala, možemo reći da se i na njih odnosi gore pomenuto saznanje. Dakle, generalno za rad terminala za daljinsko upravljanje najzgodnije je primeniti pojačano učenje. Iz svega prethodno rečenog sledi da možemo u daljem tekstu dati pregled glavnih zaključaka iz primene tehnika mašinskog učenja, odnosno tehnika pojačanog učenja na sisteme upravljanja energijom u domaćinstvu, jer su oni potpuno primenljivi i dobra polazna osnova za primenu istih tehnika na terminal za daljinsko upravljanje stanicom za punjenje električnih vozila, koja se napaja iz obnovljivih izvora električne energije.

Kod upravljanja energijom u domaćinstvu, glavni cilj je optimizacija, odnosno minimiziranje ukupne potrošnje električne energije i smanjenje računa za struju u pametnoj kući [13], kao i optimalan plasman električne energije dobijene iz obnovljivih izvora, ako su oni primenjeni. Tipičan sistem upravljanja se ne može uspešno prilagoditi raznim uređajima sa različitom složenošću rasporeda, niti je prikladan za primenu u realnom vremenu. Algoritmi pojačanog učenja (RL) su u poslednje vreme predloženi kao potencijalni kandidati za rešavanje ovih problema zbog njihove prilagodljivosti i sposobnosti da nauče prioritete i navike kupaca i optimizuju upravljanje ovakvim energetskim sistemima koji su često podložni različitim ulazima kao što su dinamičke cene električne energije, podaci meteorološke prognoze (sunce, vetar,...) i obrasci potrošnje energije (ponašanje korisnika). RL se smatra tipom algoritma sa mašinskim učenjem za donošenje odluka u stohastičkom okruženju. Ne zahteva matematički model i pogodan je za složene aplikacije u realnom vremenu. RL algoritam ima šest parametara, a to su agent, okruženje, skup stanja S, skup akcija A, nagrade R i vrednost akcije Q(s,a). Generalno, RL agent je u interakciji sa okruženjem kao što je prikazano na Sl. 1 u samom bloku terminala.

Prva upotreba pojačanog učenja za upravljanje energijom kod kuće opisana je u [14], gde se neuronska mreža koristi za kontrolu grejanja, ventilacije, klimatizacije (HVAC) i osvetljenja kako bi se smanjila nelagodnost korisnika i smanjili troškovi energije. Pored toga, postoje i radovi u kojima se koristi pojačano učenje za planiranje uključivanja/isključivanja uređaja kao odgovor na signale o cenama. To praktično znači pomeranje određenih fleksibilnih opterećenja, što omogućava minimiziranje troškova za potrošnju električne energije, tako što ne se prelazi određeni prag snage a bez izazivanja nezadovoljstva korisnika zbog odlaganja rada uređaja. Pojačano učenje primenjuje se takođe i na realizaciju različitih funkcija koje mere nezadovoljstvo korisnika kada uređaji ne uspeju da obave traženi zadatak u potrebnom vremenu. Od početka dosta je primenjivana metoda neuronske mreže [15,16], a u poslednje vreme prilično se primenjuje i metoda nazvana Q-učenje.

Algoritmi Q-učenja su RL tehnike koje su usvojene da bi se stekla optimalna politika određivanja akcija koje agent preuzima. Na osnovu primljene nagrade , agent je u stanju da optimizuje svoju politiku određivanja akcija koje treba preuzeti, i time maksimizira ukupne nagrade koje će dobiti u budućnosti. To se postiže ažuriranjem odgovarajućih koeficijenata prilikom svake iteracije, kako bi se optimizovale performanse agenta. U zavisnosti od složenosti arhitekture konkretnog sistema, moguće je i optimizaciju upravljanja energijom rešiti direktno primenom Q-učenja [13], ili primeniti Q-učenje za razbijanje glavnog problema na podzadatke koji se zatim rešavaju nezavisno korišćenjem RL [17].

Pored navedenih mogućih primena tehnika veštačke inteligencije na rešavanje glavnog problema optimizacije upravljanja energijom u ovakvim sistemima, postoje i primene u kojima se glavni problem optimizacije upravljanja energijom rešava primenom neke druge tehnike za upravljanje energije, ali se veštačka inteligencija koristi kao pomoćna metoda za predviđanje ulaznih parametara [18,19].

VII. Zaključak

U ovom radu je razmatrana primena veštačke inteligencije

 ISBN 978-86-7466-930-3

na terminal za daljinsko upravljanje stanice za punjenje električnih vozila, koja se napaja iz obnovljivih izvora energije. Prvo je prikazana arhitektura kompletnog sistema za daljinsko upravljanje stanice. Zatim je dat i pregled komponenti inteligencije koje su dosada istraživači pokušali da implementiraju u mašinama. Posebno je prikazana komponenta mašinsko učenje sa klasifikacijom i metodama koje se koriste. Potom je dat pregled primene veštačke inteligencije u oblasti sistema za upravljanje energijom u domaćinstvima. Uzevši u obzir određene sličnosti ovakvih sistema sa sistemima za daljinsko upravljanje stanicom za punjenje električnih vozila, koja se napaja iz obnovljivih izvora električne energije, zaključeno je da u ovom trenutku na terminal za daljinsko upravljanje ovakvom stanicom najviše smisla ima primeniti tehniku pojačanog učenja, i to tzv. metodu Q-učenja. Konkretna primena ove metode može biti predmet daljeg rada.

LITERATURA

[1] P. Arunkumar, K. Vijith, "IOT Enabled Smart Charging Stations for Electric Vehicle," International Journal of Pure and Applied Mathematics, vol. 119, no. 7, pp. 247-252, 2018.

[2] J. Vujasinović, G. Savić, Ž. Đurišić, "Arhitektura sistema za daljinsko upravljanje stanice za punjenje električnih vozila koja se napaja iz obnovljivih izvora energije," 64. konferencija za elektroniku, telekomunikacije, računarstvo, automatiku i nuklearnu tehniku (ETRAN), pp. 302-306, Novi Sad, Srbija, Septembar 2020.

[3] Jovan Vujasinovic, Goran Savic, Zeljko Despotovic; „Arhitecture and Sizing of System for Remote Control of Renewable Energy Sources Powered Station for Electric Vehicles Charging", IEEE International Energy Conference, May 2022, Riga, Litvania

[4] Jovan Vujasinovic, Goran Savic, Milan Prokin; „Terminal for Remote Control of Renewable Energy Sources Powered Station for Electric Vehicles Charging", 10th Mediterranean Conference on Emedded Computing, June 2021, Budva, Montenegro

[5] https://www.britannica.com/technology/artificial-intelligence#ref219078

[6] Russell Stuart J.; Norvig Peter , "Artificial Intelligence: A Modern Approach (2nd ed.)", Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2, 2003

[7] Luger George; Stubblefield William,"Artificial Intelligence: Structures and Strategies for Complex Problem Solving (5th ed.)", Benjamin/Cummings, ISBN 978-0-8053-4780-7, 2004.

[8] Poole David; Mackworth Alan; Goebel Randy, "Computational Intelligence: A Logical Approach", New York: Oxford University Press. ISBN 978-0-19-510270-3, 1998.

[9] Tao Jianhua, Tan Tieniu, "Affective Computing and Intelligent Interaction" Affective Computing: A Review Vol. LNCS 3784. Springer. pp. 981–995. doi:10.1007/11573548, 2005.

[10] https://www.britannica.com/technology/machine-learning

[11] https://www.ibm.com/cloud/learn/machine-learning

[12] Usman Zafar, Sertac Bayhan, Antonio Sanfilippo, „Home energy management system concepts, configurations, and technologies for the smart grid", IEEE Access, 10.1109/ACCESS.2020.3005244

[13] F. Alfaverh, M. Denai, and Y. Sun, "Demand Response Strategy Based on Reinforcement Learning and Fuzzy Reasoning for Home Energy Management ", IEEE Access, 10.1109/ACCESS.2020.2974286, February 2020.

[14] M. C. Mozer, ''The neural network house: An environment that adapts to its inhabitants,'' Assoc. Advancement Artif. Intell., Menlo Park, CA, USA, Tech. Rep. SS-98-02, 1998. [Online]. Available: https://www.aaai.org/Papers/Symposia/Spring/1998/SS-98-02/SS98-02-017.pdf

[15] E.Matallanas,M.Castillo-Cagigal,A.Gutiérrez,F.Monasterio-Huelin, E. Caamaño-Martín, D. Masa, and J. Jiménez-Leube, ''Neural network controller for active demand-side management with PV energy in the residential sector,'' *Appl. Energy*, vol. 91, no. 1, pp. 90–97, Mar. 2012.

[16] B. Yuce, Y. Rezgui, and M. Mourshed, ''ANN–GA smart appliance scheduling for optimised energy management in the domestic sector,'' *Energy Buildings*, vol. 111, pp. 311–325, Jan. 2016.

[17] A. Sheikhi, M. Rayati, and A. M. Ranjbar, ''Dynamic load management for a residential customer; reinforcement learning approach,'' *Sustain. Cities Soc.*, vol. 24, pp. 42–51, Jul. 2016.

[18] Luis Galván, Juan M. Navarro, Eduardo Galván, Juan M. Carrasco, Andrés Alcántara, "Optimal Scheduling of Energy Storage Using A New Priority-Based Smart Grid Control Method", *Energies* **2019**, *12*, 579, doi:10.3390/en12040579 www.mdpi.com/journal/energies.

[19] Jovan Vujasinovic, Goran Savic; „Demand Side Management and Integration of a Renewable Sources Powered Station for Electric Vehicle Charging into a Smart Grid", 15. International Conference on Applied and Theoretical Electricity ICATE, May 2021, Craiova, Romania

ABSTRACT

Application of artificial intelligence to the terminal for remote control of renewable energy sources powered station for electric vehicles charging has been presented in this paper. As the terminal enables remote control of electric vehicle chargers, smart storage batteries, smart electricity meters, cash registers, as well as, remote control of renewable energy sources and other devices within the station for electric vehicles charging, it is necessary to define and develop an algorithm for managing the terminal. In this paper, the realization of that management using artificial intelligence is considered. In this way, such charging stations for electric vehicles become completely autonomous in their work and give optimal results, which raises their availability to users of electric vehicles. It potentially encourages an increase in the use of electric vehicles for which energy is provided from renewable sources, which reduces the degree of air pollution as well as the negative effects it brings.

Application of Artificial Intelligence to the Terminal for Remote Control of Renewable Energy Sources Powered Station for Electric Vehicles Charging

Jovan Vujasinović, Goran Savić

 ISBN 978-86-7466-930-3

Prepoznavanje imena na slikama lekarskih izveštaja na srpskom jeziku u cilju zaštite ličnih podataka

Aldina Avdić, Ulfeta Marovac

***Apstrakt*— Savremeni način života neizostavno uključuje upotrebu računara i mobilnih telefona u svim svojim segmentima, pa i kada je u pitanju zdravstvo. Pored elektronskog zdravstva koje se sve više razvija, pogotovo od kada se svet suočio sa pandemijom korona virusa, sve više ljudi traži i savete o zdravlju na društvenim mrežama. Tom prilikom dodaju slike koje sadrže njihove zdravstvene rezultate, ne mareći o tome da na taj način ostavljaju i svoje lične podatke. U ovom radu data je metoda za prepoznavanje imena na slikama medicinskih izveštajima napisanih na srpskom jeziku u cilju njihove de-identifikacije. Ova metoda bazirana je na optičkom prepoznavanju karaktera, metodama obrade prirodnog jezika i pravilima i ima široku upotrebu, jer je de-identifikacija elektronskih medicinskih izveštaja neophodan korak za njihovu bilo kakvu dalju analizu.**

***Ključne reči*— de-identifikacija, elektronski medicinski izveštaji, prepoznavanje imenovanih entiteta, obrada prirodnog jezika, optičko prepoznavanje karaktera, zaštita privatnosti, srpski jezik.**

I. Uvod

Informacione i komunikacione tehnologije već duže vreme nalaze svoju primenu u zdravstvenom sistemu. E-zdravstvo (engl. *e-Health*) je termin nastao krajem dvadesetog veka i zasniva se na efikasnijem i kvalitetnijem pružanju zdravstvenih usluga uz pomoć savremenih tehnologija, mogućnostima za obezbeđivanje mobilnosti lekara i pacijenata uz integraciju sa postojećim sistemima [1].

Savremene tehnologije u zdravstvu omogućavaju kompletno elektronsko vođenje zdravstvene dokumentacije u svim segmentima rada (elektronski karton pacijenta), telemedicinu, telekonsultacije, upravljanje znanjem o zdravlju i mobilno i sveprisutno zdravstvo. Na ovaj način omogućava se lakše i uspešnije lečenje pacijenata i smanjenje administrativnih ograničenja prilikom pružanja medicinskih usluga.

Prednosti e-zdravstva su elektronsko praćenje i beleženje zdravstvenog stanja pacijenata, bolja dijagnostika, pristup medicinskim podacima bilo gde i bilo kada, kao i brz prenos informacija korisnicima putem telemedicine i Internet servisa.

Aldina Avdić – Departman za tehničko-tehnološke nauke, Državni Univerzitet u Novom Pazaru, Vuka Karadžića 9, 36300 Novi Pazar, Srbija (e-mail: apljaskovic@np.ac.rs).

Ulfeta Marovac – Departman za tehničko-tehnološke nauke, Državni Univerzitet u Novom Pazaru, Vuka Karadžića 9, 36300 Novi Pazar, Srbija (e-mail: umarovac@ np.ac.rs).

Elektronski medicinski zapis (engl. *EMR* ili *EHR* - *electronic medical (health) report*) je izveštaj pacijenta koji čuva podatke o zdravstvenom stanju, dijagnozama i terapijama. Oni se kreiraju pomoću medicinskih informacionih sistema i sadrže privatne podatke o pacijentu (ime, prezime, lični broj, datum rođenja, broj kartice, broj osiguranja, adresu itd.), beleške lekara, dijagnoze, laboratorijske izveštaje, terapije itd [2].

Motivacija za ovaj rad je pripremanje elektronskih izveštaja za dalju analizu korišćenjem metoda veštačke inteligencije, mašinskog učenja i metoda obrade prirodnog jezika. U kakvom god da su obliku elektronski medicinski izveštaji (tekst ili slika) prvi korak ka njihovoj etičkoj obradi jeste uklanjanje ličnih podataka (ovo je uređeno regulativom – engl. *General Data Protection Regulation - GDPR*). Za potrebe rada prikupljen je skup podataka, i kreirana i primenjena metoda za prepoznavanje imena i prezimena na srpskom jeziku sa slika medicinskih izveštaja bazirana na optičkom prepoznavanju slova (engl. *Optical Caracter Recognition - OCR*) i metodama obrade prirodnog jezika (engl. *Natural Language Processing - NLP*) i prepoznavanja imenovanih entiteta (engl. *Named Entity Recognition – NER*). Stoga će nadalje ovi pojmovi biti detaljno objašnjeni.

Uredba Evropske unije o zaštiti podataka o ličnosti GDPR definiše propise o zaštiti podataka ličnosti u Evropskoj uniji, ali se ona odnosi i na kompanije sa sedištem u zemljama izvan Evropske unije, ukoliko one obrađuju podatke o ličnosti rezidenata Evropske unije. Po članu 3 GDPR regulative potrebno je da se uskladi zaštita osnovnih prava fizičkih lica u vezi sa aktivnostima obrade ličnih podataka. U Srbiji je na snazi Zakon o zaštiti podataka o ličnosti iz 2018. godine koji je usklađen sa GDPR [3]. Ona obezbeđuje pojedincima veću kontrolu nad ličnim podacima i obavezuje kompanije koje prikupljaju i analiziraju lične podatke da promene svoj model poslovanja u skladu s njom. Pod ličnim podacima podrazumeva se bilo koja kombinacija ličnih činjenica koja tačno određuje jednog pojedinca, to su, između ostalog, ime i prezime, JMBG, podaci o lokaciji, fizički, fiziološki, genetski, mentalni, ekonomski, socijalni, kulturni ili bilo koji drugi faktori.

Osnovni zadatak OCR softvera je da digitalne slike, na kojima se nalaze skenirani ili fotografisani tekstovi sa štampača, pisaćih mašina, knjiga, novina, časopisa ili poslovne dokumentacije, pretvori u promenljivi digitalni tekstualni oblik, tako što će sa rasterskih slika, prepoznati

 ISBN 978-86-7466-930-3

slova, reči i čitave tekstove.[4].

Procesiranje prirodnog jezika je grana veštačke inteligencije koja omogućava sistemu da razume značenje podataka na ljudskom jeziku. Njen krajnji cilj je čitanje, dešifrovanje, razumevanje i nalaženje smislenosti u prirodnom jeziku. Većina NLP tehnika oslanja se na mašinsko učenje da bi se zaključilo o značenju podataka na prirodnim jezicima. Prepoznavanje imenovanih entiteta je grana NLP-a kojom se u tekstu označavaju sintagme u nestrukturiranom tekstu u unapred definisane kategorije kao što su imena osoba, organizacije, lokacije, medicinski kodovi, vremenski izrazi itd [5] [6].

Ovaj rad je organizovan na sledeći način. U drugom poglavlju dat je pregled radova iz oblasti. U trećem poglavlju opisani su korišćeni podaci i metoda za prepoznavanje imena na slikama medicinskih izveštajima napisanih na srpskom jeziku u cilju njihove de-identifikacije. Zatim sledi opis korišćenih tehnologija, dobijeni rezultati i njihova diskusija. Na kraju je dat zaključak i pravci daljeg istraživanja.

II. Stanje u oblasti

Povezana istraživanja na ovu temu opisuju metode za normalizaciju i izdvajanje znanja iz medicinske evidencije koje nisu direktno povezane sa primenom na srpskom jeziku. Većina istraživanja odnosi se na korpuse engleskog govornog područja i leksičke resurse koji su javno dostupni. Za bugarski jezik postoje rezultati izdvajanja informacija iz velikog korpusa nestrukturiranih podataka i njihovog pribavljanja u strukturiranom obliku [7].

Normalizacija je prvi korak koji se koristi u klasifikaciji i obeležavanju medicinskih termina [8]. Obrada medicinskog izveštaja sastoji se od nekoliko koraka kao što su prečišćavanje podataka, integracija, transformacija, redukcija i na kraju zaštita podataka. Izvlačenjem kliničkih odnosa iz medicinskih izveštaja mogu se identifikovati odnosi između referenci na lekove i njihovih atributa. Pregled medicinskih informacionih sistema pokazuje da 60% komercijalnih sistema koristi metode zasnovane na pravilima, dok se u naučnim istraživanjima više koriste metode mašinskog učenja [9]. Najpopularniji sistemi za obeležavanje medicinskog teksta su CTAKES i CLAMP sistemi [9] [10]. Autori su se u radu bavili identifikacijom medicinskih pojmova u tekstovima koje su napisali pacijenti, koristeći kraudsoursing [11] .

Na jezicima bivše Jugoslavije nisu pronađeni radovi koji bi se bavili procesom slobodnog teksta u medicinskim izveštajima i javno dostupnim leksičkim izvorima u medicinskom domenu. „Wordnet za biomedicinske nauke“ za srpski jezik sadrži skupove sinonimnih reči ili tačnije različite delove govora (engl. PoS, parts of speech) sa novim konceptom za šest ontoloških kategorija (genetika, virus, bakterije, ćelije, naučna polja i mikroorganizam) [12].

Klasifikacija u medicinskim izveštajima na srpskom jeziku opisana je u radu [13]. De-identifikacija ličnih podataka u srpskom jeziku opisana je u radu [14].

Primena OCR u zdravstvu opisana je u radu [15].

III. Podaci i metodologija

Za potrebe ovog istraživanja sakupljena je 71 slika sa društvene mreže Facebook, koja sadrži medicinski izveštaje koje sadrži lične podatke. Ove izveštaje su okačili korisnici zatvorene grupe Insulinska rezistencija. Neki učesnici u grupi svesni su izlaganja privatnih podataka i pre postavljanja slike svoje ime oboje ili iseku sliku tako da lični podaci nisu dostupni, ali budući da su ovo samo slike od januara 2022. godine, to govori koliko je jednostavno naći elektronske izveštaje koji sadrže lične podatke na internetu na sličnim grupama.

Metoda za prepoznavanje imena sadrži sledeće korake:

- na sve slike je prvo primenjen OCR softver otvorenog koda *Tessaract* [17], čime se izdvaja tekst sa slike (Sl. 1),

```
from PIL import Image
import pytesseract
import numpy as np
import spacy
from spacy import displacy
import translators as ts
import os
import re

directory = 'C:/Users/aldin/Desktop/Podaci/Insulinska'
NER = spacy.load("en_core_web_sm")

for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    if os.path.isfile(f):
        filename = f
        print(f)
        img1 = np.array(Image.open(filename))
        pytesseract.pytesseract.tesseract_cmd =
        r'C:\Users\aldin\AppData\Local\Tesseract-OCR\tesseract.exe'
        text = pytesseract.image_to_string(img1)
```

Sl. 1 Izdvajanje teksta iz slika

- zatim su primenjena pravila i regularni izrazi za obeležavanje imena i prezimena (Sl. 2). Pravila su definisana od strane autora, a odnose se na to da je u liniji koja se čita veća verovatnoća da se nađe ime i prezime, ako se u toj liniji nalaze reči „prezime“, „pacijent“, „ime i prezime“ i sl. Takođe, ako je na medicinskom izveštaju neka sintagma napisana velikim slovima, ili prvim početnim slovima sa blanko znakovima između, i to treba uzeti u obzir kao moguće ime i prezime pacijenta. Do ovih pravila došlo se analizom sadržaja prikupljenog skupa podataka.

```
        lines = text.split('\n')
        for line in lines:
            if "prezime" in line:
                print(line)
            if "pacijent" in line:
                print(line)
            rl = re.findall('^[a-zA-Z]+(?:[\s.]+[a-zA-Z]+)*$', line)
```

Sl. 2 Primeri pravila

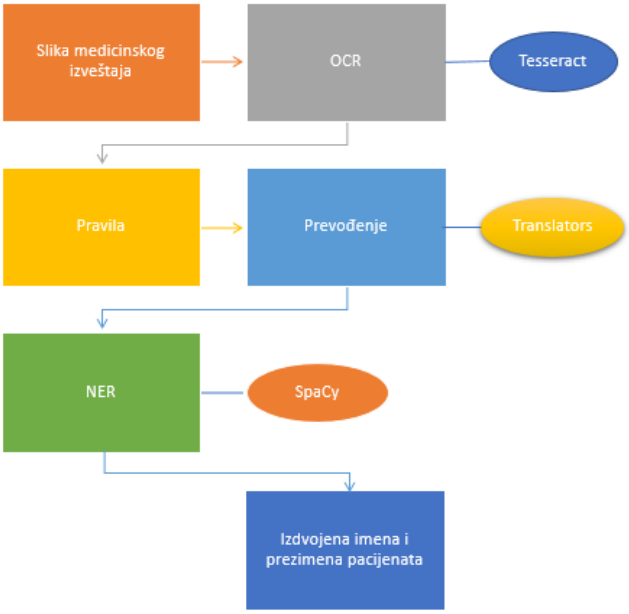
- u koliko je prethodni korak neuspešan, sledi prevođenje teksta na engleski korišćenjem *Python* paketa *Translators* [18],
- zatim se koristi *Spacy* softver za NER na engleskom za prepoznavanje vlastitih imena [19] (Sl. 3).

 ISBN 978-86-7466-930-3

U program su uključene biblioteke *Image* za rad sa slikama, *PyTesseract* za OCR, *NumPy* za jednostavniju obradu i programiranje, zatim *SpaCy* za prepoznavanje imenskih entiteta, *Translators* za prevođenje reči, *os* za rad sa fajlovima i *re* za rad sa regularnim izrazima.

```
        if rl:
            textt = ts.google(rl[0], from_language='sr',
            to_language='en')
            textl= NER(textt)
            for word in textl.ents:
                if word.label_=="PERSON":
                    print(word.text)
```

Sl. 3 Primeri prevođenja i primene Spacy alata za prepoznavanje imenovanih entiteta



Sl. 4. Koraci u prepoznavanju ličnih imena u slikama medicinskih izveštaja na srpskom jeziku

Ova metoda ne zahteva prethodno obeležavanje niti treniranje modela metodama mašinskog učenja (Sl. 4). To su koraci koji su pravci daljeg istraživanja, uz povećanje skupa podataka kako bi se prvobitni rezultati popravili. Uklanjanje ovih podataka na izvornoj slici nije rađeno, ali jeste na tekstovima EHR-ova koji se čuvaju u bazi podataka.

IV. Rezultati i diskusija

U tabeli 1 dati su rezultati primene metode za izdvajanje ličnih imena prezimena. Rezultati uključuju samo one podatke gde je izvučeno samo ime i prezime pacijenta.

Pod preciznošću se smatra odnos broja slika sa izdvojenim ličnim podacima i ukupnog broja slika, a pod odzivom odnos broja tačno izdvojenih imena i prezimena i ukupnog broja slika.

Ono što utiče na rezultate jeste i kvalitet slika i sama preciznost OCR alata, pa je kod jednog broja izveštaja izdvajanje teksta bilo nemoguće. Zato je potrebno uključiti i dodatna predprocesiranja slike. Rezultati pokazuju da je ovo dobra osnova, ali da bi se oni popravili potrebno je razmotriti korišćenje dodatnih resursa za srpski jezik, poput rečnika imena. Takođe, rezultati se mogu popraviti i korišćenjem modela mašinskog učenja i proširenjem trening skupa kao i proširenjem skupa pravila.

TABELA I
Rezultati primene metode

Broj slika	71
Broj slika sa izdvojenim ličnim podacima	41
Broj tačno izdvojenih imena i prezimena	22
Preciznost	58%
Odziv	54%

V. Zaključak

Pacijenti često traže mišljenja o svojim zdravstvenim rezultatima na socijalnim mrežama, u specijalizovanim grupama korisnika sa istom dijagnozom. Tada često postavljaju slike koje sadrže privatne podatke. Alat za de-identifikaciju elektronskih medicinskih izveštaja imao bi široku primenu, jer pored toga što bi se korisniku skrenula pažnja da postavlja dokument sa osetljivim podacima, čak bi se mogli koristiti podaci iz medicinskih informacionih sistema za dalju obradu, analizu i dobijanje znanja. U radu je dat jedan način za izdvajanje imena i prezimena pacijenta zasnovan na OCR i NER. U daljem radu radiće se na dopunjavanju resursa, pravila i uključivanja modela mašinskog učenja, proširenja trening skupa, uz predprocesiranje slika, kako bi se postigli što bolji rezultati. Takođe, jedan od pravaca daljeg istraživanja jeste uklanjanje (zamagljivanje) pronađenih ličnih podataka na procesiranoj slici medicinskog izveštaja.

Zahvalnica

Ovaj rad je delimično finansiran od strane Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije u okviru projekata III44007.

Literatura

[1] A. R. Avdić, U. M. Marovac and D. S. Janković, “Smart Health Services for Epidemic Control,” in 2020 55th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), 2020.

[2] S. Meystre and M. E. f. t. I. Y. S. o. P. Records, "Electronic Patient Records: Some answers to the data representation and reuse challenges: Findings from the section on Patient Records," Yearbook of Medical Informatics, vol. 16, no. 01, p. 47–48, 2007.

[3] M. Luca, E. Lievevrouw and I. V. Hoyweghen, "Fit for purpose? The GDPR and the governance of European digital health," Policy studies, vol. 41, no. 5, pp. 447-467, 2020.

[4] J. Memon, M. Sami, R. A. Khan and M. Uddin, "Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR)," IEEE Access, vol. 8, pp. 142642-142668, 2020).

[5] G. G. Chowdhury, “Natural language processing,” Annual review of information science and technology, vol. 37, no. 1, pp. 51-89, 2003.

[6] D. J. Hand and N. M. Adams, Data mining, Wiley StatsRef: Statistics Reference Online, 2014, pp. 1-7.

 ISBN 978-86-7466-930-3

[7] I. Nikolova, D. Tcharaktchiev, S. Boytcheva, Z. Angelov and G. Angelova, "Applying language technologies on healthcare patient records for better treatment of Bulgarian diabetic patients," in International Conference on Artificial Intelligence: Methodology, Systems, and Applications, Springer, Cham, 2014, September.
[8] A. R. Avdić, U. M. Marovac and D. S. Janković, "Normalization of Health Records in the Serbian Language with the Aim of Smart Health Services Realization," Facta Universitatis, Series: Mathematics and Informatics, pp. 825-841, 2020.
[9] A. M. Milenkovic, P. J. Rajkovic, T. N. Stankovic and D. S. Jankovic, "Application of medical information system MEDIS.NET in professional learning," in 19thTelecommunications Forum (TELFOR) Proceedings of Papers, Belgrade, 2011.
[10] V. Garla, V. L. Re III, Z. Dorey-Stein, F. Kidwai, M. Scotch, J. Womack, A. Justice and C. Brandt, "The Yale cTAKES extensions for document classification: architecture and application," Journal of the American Medical Informatics Association, vol. 18, no. 5, pp. 614-620, 2011.
[11] E. Soysal, J. Wang, M. Jiang, Y. Wu, S. Pakhomov, H. Liu and H. Xu, "CLAMP - a toolkit for eficiently building customized clinical natural language processing pipelines," Journal of the American Medical Informatics Association, vol. 25, no. 3, pp. 331-336, 2018.
[12] D. L. MacLean and J. Heer, "Identifying medical terms in patient-authored text: a crowdsourcing-based approach," Journal of the American Medical Informatics Association, vol. 20, no. 6, pp. 1120-1127, 2013.
[13] N. e. a. Ivković-Berček, "Kooperativan rad na dogradnji Srpskog wordneta.".
[14] A. Avdić, U. Marovac and D. Janković, "Automated labeling of terms in medical reports in Serbian," Turkish Journal of Electrical Engineering & Computer Sciences, vol. 28, no. 6, pp. 3285-3303, 2020.
[15] B. Šandrih, C. Krstev and R. Stanković, "Development and evaluation of three named entity recognition systems for serbian-the case of personal names," in In Proceedings of the International Conference on Recent Advances in Natural Language , 2019, September.
[16] R. Mittal and A. Garg, "Text extraction using OCR: a systematic review. In 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)," 2020, July.
[17] R. Smith, "An overview of the Tesseract OCR engine," in In Ninth international conference on document analysis and recognition (ICDAR 2007), 2007, September.
[18] P. Translators, "https://pypi.org/project/translators/," [Online]. [Accessed May 2022].
[19] X. Schmitt, S. Kubler, J. Robert, M. Papadakis and Y. LeTraon, "A replicable comparison study of NER software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate," in Sixth International Conference on Social Networks Analysis, Management and and Security (SNAMS), 2019, October. *Title of Standard*, Standard number, date.

ABSTRACT

The modern way of life inevitably includes the use of ICT in all its segments, even in healthcare. In addition to e-health, whose development is growing, especially since the world faced the coronary virus pandemic, most people are also looking for health advice on social networks. On that occasion, they upload pictures that contain their health results, with their personal data. This paper presents a method for recognition of personal names in images of medical reports written in Serbian to de-identify them. This method is based on optical character recognition, natural language processing methods and rules and has wide application, as de-identification of electronic medical reports is a necessary step for their further analysis.

Recognition of names on images of medical reports in Serbian to protect personal data

Aldina Avdić, Ulfeta Marovac

 ISBN 978-86-7466-930-3

Sistem za automatizaciju testova za proveru znanja baziran na transformaciji predikatskih iskaza

Ulfeta Marovac, Department for Technical Sciences, State University of Novi Pazar, Serbia
Aldina Avdić, Department for Technical Sciences, State University of Novi Pazar, Serbia

***Apstrakt*— Obradom prirodnog jezika omogućava se da računari uspostave komunikaciju sa čovekom, da razumeju jezik čoveka, ali i da ga transformišu i da se čoveku obrate na njemu prirodnom jeziku. Problem istinitnosne vrednosti iskaza izrečenih prirodnim jezikom je teško utvrditi zbog bogatstva rečnika i dvosmislenosti značenja. Preslikavanjem iskaza prirodnog jezika u iskaze predikatske logike moguće je utvrditi odnose među različitim predikatskim iskazima kao i izvršiti transformaciju iz jednog oblika pojavljivanja u drugi. Jedna od primena ovakvog znanja može biti u automatizovanom kreiranju testova znanja.**

***Ključne reči*—obrada prirodnog jezika, predikatska logika, automatizovani testovi znanja, srpski jezik**

I. UVOD

Testiranje je važna komponenta učenja, jer omogućava da se vidi da li je ispitanik usvojio informacije koje su mu izložene. Sve prisutnije e-učenje zahteva digitalizaciju okruženja za predavanja, ali i ispitivanje. Postoji puno dostupnih softverskih alata koji olakšavaju izradu testova. Njihove prednosti su brojne kao:

- Ušteda vremena u kreiranju testova, jer nema potrebe za tradicionalnim metodama
- Uvek su dostupni na mreži, deljivi su i lakše im pristupaju ispitanici.

Međutim ovi alati se uglavnom koriste kako bi ubrzali proces kreiranja testa i njegovu implementaciju, ali ne i u kreiranju baze znanja za test, i ne poseduju mogućnost provere ispravnosti pitanja koja su data kao slobodan tekst.

Obrada prirodnog jezika (Natural Language Processing - NLP [1]) je oblast veštačke inteligencije koja olakšava komunikaciju čoveka i računara. Naravno da je u procesu obrazovanja uloga nastavnika nezamenljiva, ali se novim tehnologijama može olakšati njihov posao. Ključni faktor u formiranju baze znanja za jedan kurs je uvek nastavnik, jer on daje činjenice koje su relevantne za problem koji se obrađuje. Utvrđivanje tačnosti iskaza koji su napisani na prirodnom jeziku nije jednostavno zbog njihove dvosmislenosti i složenosti, tako da se automatizacija testova svodi na postavljanje pitanja i adekvatnih tačnih i netačnih odgovora koje ispitanik treba da izabere. U ovom procesu određivanje tačnosti nekog pitanja vrši nastavnik. Da bi se pomoglo nastavniku u kreiranju veće količine testova od početnog tačnog ili netačnog iskaza, primenom pravila iskazne logike mogu se dobiti iskazi u drugom obliku čija je istinitnosna vrednost poznata. Sličan postupak se može vršiti i u obrnutom smeru gde bi se provera datog odgovora ispitanika u obliku slobodnog teksta pokušala izjednačiti sa nekim od poznatih odgovora.

Obrada prirodnog jezika je teška zbog specifičnosti jezika na kome se primenjuje. Uvođenjem formalnih jezika formiraju se rečenice na osnovu unapred definisanih pravila. Formalni jezici koriste logičke veznike, kvantifikatore, promenljive (termine), oznake za svojstva i odnose (predikate) i pomoćne znakove (zagrade) za izražavanje rečenica. Jedan primer formalne gramatike koja proizvodi predikatske iskaze na hrvatskom jeziku dat je u diplomskom radu autora Stanić [2].

U ovom radu biće prikazana obrada rečenica datih na srpskom jeziku. Da bi se izvršila obrada ovih rečenica moraju se koristiti pravila srpskog jezika kao i specijalni jezički resursi kojima će se umanjiti pojavljivanje sinonima.

Rad je organizovan na sledeći način: u drugom poglavlju opisani su predikatski iskazi i njihovi odnosi; u trećem poglavlju su prikazani algoritmi za transformaciju predikatskih iskaza unutar logičkog kvadrata; model za automatizaciju testova znanja prikazan je u četvrtom poglavlju i u poslednjem poglavlju dat je zaključak i ideje za proširenje modela.

II. PREDIKATSKI ISKAZI I NJIHOV ODNOS

Iskazna logika je precizan matematički alat za računanje logičkih vrednosti složenih iskaznih formula kao i za donošenje zaključaka iz njih. Iskazna logika se bavi rečenicama u celini i ne zalazi u njihovu unutrašnju strukturu. Predikatska logika razmatra smisao polaznih iskaza.

U predikatskoj logici razlikujemo:

- **Term (T)-** objekta o kome se govori
- **Predikat (P)** – osobina objekta ili veza među objektima.

Term može biti konstanta (konkretan objekat) ili promenljiva (nedovoljno određen objekat). Atomični predikatski iskaz sastoji se od terma i predikata.

Neki iskazi sadrže neodređen objekat pa ne mogu imati jedinstvenu vrednost. Upotrebom negacije, veznika i kvantifikatora formira se predikatska formula [3].

Kvantifikacija je tema koja spaja lingvistiku, logiku i filozofiju. Kvantifikatori su osnovni alati pomoću kojih, u jeziku ili logici, upućujemo na količinu stvari [4]. Upotrebom kvantifikatora izražava se koliko je neko tvrđenje tačno i

Ulfeta Marovac is with Department for Technical Sciences, State University of Novi Pazar, Vuka Karadžića bb, 36300 Novi Pazar ,Serbia (e-mail: umarovac@np.ac.rs), https://orcid.org/0000-0001-7232-3755

Aldina Avdić is with Department for Technical Sciences, State University of Novi Pazar, Vuka Karadžića bb, 36300 Novi Pazar ,Serbia (e-mail: apljaskovic@np.ac.rs) , https://orcid.org/0000-0003-4312-3839

 ISBN 978-86-7466-930-3

dobijaju se rečenice koje imaju jedinstvenu logičku vrednost. Univerzalni kvantifikator (∀) se koristi da prikaže izraze koji su tačni za sve vrednosti terma. Sa druge strane, egzistencijalni kvantifikator se koristi da prikaže da postoji term za koji je tvrđenje tačno.

Predikatske rečenice se mogu podeliti prema kvantitetu na:

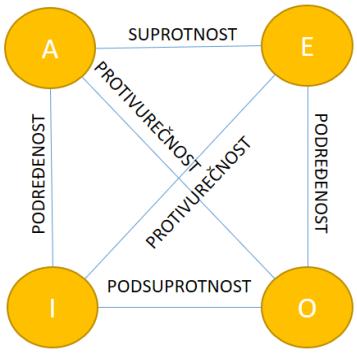
- Univerzalne - opisuju osobinu koju imaju svi pripadnici nekog skupa
- Partikularne - opisuju osobinu koju ima bar jedan element nekog skupa
- Singularne - opisuju osobinu jednog člana nekog skupa.

Predikatske rečenice po kvalitetu se dele na:

- afirmativne - opisuju da subjekat ima neku osobinu,
- negativne - opisuju da subjekat nema neku osobinu,
- limitativne - opisuju da subjekat ima neku osobinu opisanu negacijom.

Izdvojimo četiri tipa predikatskih iskaza koja su prikazana u logičkim kvadratu suprotnosti (Slika 1):

- $(\forall T)P(T)$ - univerzalno-afirmativni iskaz (A)
- $(\forall T)\neg P(T)$ - univerzalno-negativni iskaz (E)
- $(\exists T)P(T)$ - - partikularno-afirmativni iskaz (I)
- $(\exists T)\neg P(T)$- partikularno-negativan iskaz (O).



Slika 1. Logički kvadrat

Predikatske rečenice prikazane u logičkom kvadratu su povezane i ukoliko se zna istinitost jednog tipa može se zaključiti o istinitosti bar još jednog od ostala tri tipa [5].

Prikazani logički kvadrat je motivacija za kreiranje modela za transformaciju jednog predikatskog iskaza za koji je poznata logička vrednost u iskaz kome se može zaključiti tačna logička vrednost.

Može se uočiti povezanost između predikatske logike i prirodnog jezika i to: termi igraju ulogu sličnu onoj koju u prirodnom jeziku igraju imenice i zamenice, a predikati ulogu sličnu glagolima. Prirodni jezik je mnogo bogatiji i nisu dovoljno precizni. Problem prirodnog jezika je i u tome što se isti iskaz može izraziti na više načina.

III. AUTOMATIZOVANO PRESLIKAVANJE PREDIKATSKIH ISKAZA NA SRPSKOM JEZIKU UNUTAR LOGIČKOG KVADRATA

Rečenice u srpskom jeziku mogu se podeliti na proste i složene [6]. Proste rečenice se mogu predstaviti pomoću dva tipa:

1. subjekta (S) i glagolskog predikata (GP)
2. subjekat (S), pomoćni glagol (PG) i imenski predikat (IP).

Prosta rečenica srpskog jezika može se povezati sa atomičnim predikatskim iskazom. Predikatska forma na prirodnom jeziku izražava se upotrebom veznika, negacije i kvantifikatora. U srpskom jeziku se negacija, veznici i kvantifikatori mogu javiti u različitim oblicima. U radu [7] prikazana je veza između predikatske logike i srpskog jezika, kao i pravila prelaza predikatskih formula izraženih na srpskom jeziku unutar logičkog kvadrata.

U ovom radu je dat algoritam za automatizaciju preslikavanja predikatskih iskaza na srpskom jeziku unutar logičkog kvadrata zasnovan na pravilima definisanim u [7]. Za ove potrebe napravljeni su rečnici sa različitim signalima negacije (**D_SN**) kao i rečnik sa potvrdnim (P_PG) i odričnim oblicima (O_PG) pomoćnih glagola (PG) "imam", „biti" i „hteti" pomoću kojih se može graditi negacija (**D_PG**).

Formiran je takođe:

- **D_UKP**-rečnik različitih termina kojima se izražava univerzalni kvantifikator u potvrdnom obliku (UKP);
- **D_UKO**-rečnik različitih termina kojima se izražava univerzalni kvantifikator u odričnom obliku (UKO);
- **D_EK**-rečnik različitih termina kojima se izražava egzistencijalni kvantifikator UEK.

Kvantifikatori u srpskom jeziku se menjaju po licima, rodu i broju pa se ove informacije nalaze u odgovarajući rečnicima (L,R,B).

U partikularnim rečenicama uz neke kvantifikatore (na primer „postoji") se pojavljuje odnosna zamenica (OZ) iza subjekta pa je napravljen i rečnik odnosnih zamenica **D_OZ** sa odgovarajućim informacijama (L,R,B). Za neke oblike egzistencijalnog kvantifikatora se ne upotrebljava odnosna zamenica (na primer „neki"), stoga uz svaki egzistencijalni kvantifikator stoji i opis („tip") da li ide uz odnosnu zamenicu.

Dalje će biti prikazan algoritmi kojim se od iskaza A dobijaju drugi iskazi logičkog kvadrata. Dobijenim iskazima se može pridružiti i njihova logička vrednost (T-tačno , N-netačno, NaN-nepoznata logička vrednost).

Sva četiri tipa predikatskih iskaza u srpskom jeziku se mogu naći u jednom od dva tipa rečenica prethodno navedenih i to:

Univerzalno-afirmativni iskaz (iskaz_A)

- TIP1 - UKP+S+GP
- TIP2 - UKP+S+PG+IP

Univerzalno-negativni iskaz (iskaz_E)

- TIP1 - UKP+S+SN+GP
- TIP2 - UKO+S+O_PG+IP

Partikularno-afirmativni iskaz (iskaz_I)

- TIP1 - EK+S [+ OZ] +GP

 ISBN 978-86-7466-930-3

- TIP2 - EK+S [+OZ]+P_PG+IP

Partikularno-negativan iskaz (iskaz_O)

- TIP1 - EK+S [+ OZ] +SN +GP
- TIP2 - EK+S [+OZ]+O_PG+IP.

Algoritam 1 (**UP_transformacija**) vrši transformaciju univerzalnih iskaza (A i E) u partikularne (I i O). Inverzna funkcija (**PU_transformacija**) može se slično predstaviti i ona vrši transformaciju partikularnih iskaza (I i O) u univerzalne iskaze (A i E).

Algritam 1: Preslikavanje univerzalnih iskaza u partikularne
Input: ($univerzalni_iskaz, $tip_rečenice,)
Output:($niz_partikularnih_iskaza)
UP_transformacija

```
1.  // Odrediti lice, rod i broj
2.  ($L,$R,$B)= SELECT L,R,B FROM D_UKP
3.                  WHERE UKP=$UKP)
4.  // Skup egzistencijalnih kvantifikatora
5.  $Skup_EK=   SELECT EK, TIP FROM D_EK
6.                  WHERE (L,R,B)=( $L, $R, $B)
7.  for $EK, $tip in $Skup_EK do
8.     $p_iskaz=$univerzalni_iskaz ( $UKP<-$EK )
9.     if ($tip=”uz odnosnu zamenicu”)
10.      /*Dodaje se odnosna zamenica u odgovarajućem
11.       *licu , rodu i broju
12.       *ako egzistencijalni kvatifikator zahteva
13.       */
14.      $SkupOZ=   SELECT OZ FROM D_OZ
                    WHERE (L,R,B)=( $L, $R, $B)
15.       for $OZ in $SkupOZ do
16.            $p_iskaz=$p_iskaz ( $S<-($S+$OZ))
17.       end_for
18.    end_if
19.    $niz_partikularnih_iskaza += p_iskaz
20.  end_for
21.  RETURN  $niz_partikularnih_iskaza
```

Algoritam 2 (**AN_transformacija**) vrši transformaciju afirmativnih iskaza (A i I) u negativne (E i O). Inverzna funkcija (**NA_transformacija**) može se slično predstaviti i ona vrši transformaciju negativnih iskaza (E i O) u afirmativne (A i I).

Alogritam 2: Preslikavanje afirmativnih iskaza u negativne
Ulaz: ($afirmativni_iskaz, $tip_rečenice)
Izlaz:($niz_negativnih_iskaza)
AN_transformacija

```
1.  /* TIP1:  iskaz_A($UKP+$S+$GP)
2.  *          iskaz_I($EK+$S+$OZ+$GP)
3.  * Dodavanje singala negacije negacije “ne”
4.  */
5.  if ($tip_rečenice== TIP1)
6.        $SN=“ne“
7.        $n_iskaz=$afirmativni_iskaz ($GP<-$SN+$GP )
8.        $niz_negativnih_iskaza += $n_iskaz
9.  end_if
10. /* TIP2:  iskaz_A($UKP+$S+$P_PG+$IP)
11. *          iskaz_I($EK+$S+$OZ+$P_PG+$IP)
12. * Zamena potvrdnog pomoćnog glagola sa
13. * odričnim.
14. */
15. if ($tip_rečenice== TIP2)
16.    $O_PG=SELECT O_PG FROM D_PG
17.                WHERE  P_PG=$P_PG
18.    $n_iskaz=$afirmativni_iskaz ( $P_PG<-$O_PG )
19.    /* Ako je kvantifikator univerzalni predstavlja se
20.     * negativnim rečenicama odričnim
21.     * kvantifikatorom
22.     */
23.    $kvantifikator= kvantifikator(n_iskaz)
24.    If $kvantifikator in DUKP
25.      // Odrediti lice, rod i broj
26.      ($L,$R,$B)= SELECT L,R,B FROM D_UKP
                     WHERE UKP=$kvantifikator
27.      //Skup odričnih kvantifikatora
28.      $Skup_UKO=   SELECT UKO FROM D_UKO
                      WHERE (L,R,B)=( $L, $R, $B)
29.    for $UKO in $Skup_UKO do
30.         $n_iskaz=$n_iskaz ( $UKP<-$UKO )
31.         $niz_negativnih_iskaza += n_iskaz
32.    end_for
33.   else
34.     $niz_negativnih_iskaza += n_iskaz
35.   end_if
36. end_if
37. RETURN  $niz_negativnih_iskaza
```

Očigledno je da kombinacijom datih transformacija možemo iz svakog tipa predikatske rečenice dobiti preostala tri, i to u više oblika u zavisnosti od oblika kvantifikatora koji se izabere.

IV. MODEL ZA AUTOMATIZACIJU TESTOVA ZA PROVERU ZNANJA

Prirodni jezici nisu precizni pa predloženi model transformacije zavise od jasnoće ulaza. Većina tvrdnji može se uklopiti u jedan od tipova predikatskih rečenica i ukoliko je njena istinitost poznata može izvršiti transformaciju polazne tvrdnje u skup iskaza koji imaju istu ili suprotnu tačnost. Tabela 1 prikazuje ulazne parametre, primenjene transformacije i rezultat kada se na ulazu nađu iskazi tipa A i E. Slično se može odrediti i za iskaze tipa I i O za koje ukoliko su netačni možemo zaključiti istinitnosne vrednosti o ostala tri tipa predikatskih iskaza koje možemo dobiti odgovarajućim transformacijama ili ukoliko su tačni možemo ih transformisati u netačne iskaze tipa (E i A).

 ISBN 978-86-7466-930-3

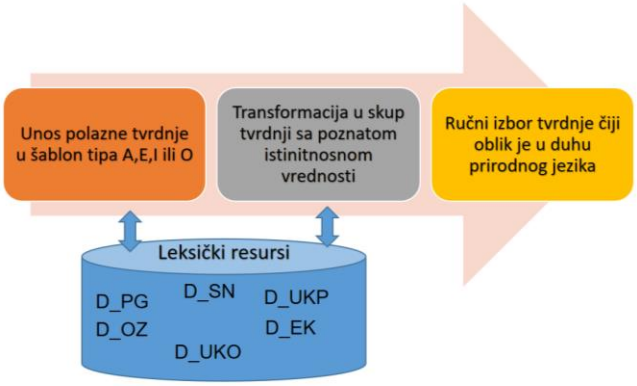
TABELA I
OPIS MOGUĆEG TIPA ULAZA, PRIMENJENIH TRANSFORMACIJA I IZLAZA

Tip ulaznog iskaza	Tačnost	Primenjene transformacije	Tip skupa izlaznih iskaza	
A	Tačno	AN	E	Netačno
A	Tačno	UP	I	Tačno
A	Tačno	AN+UP	O	Netačno
A	Netačno	AN+UP	O	Tačno
E	Tačno	NA	A	Netačno
E	Tačno	UP+NA	I	Netačno
E	Tačno	UP	O	Tačno
E	Netačno	UP+NA	I	Tačno

Postoje dva slučaja koja se pri obradi testova znanja mogu tretirati:

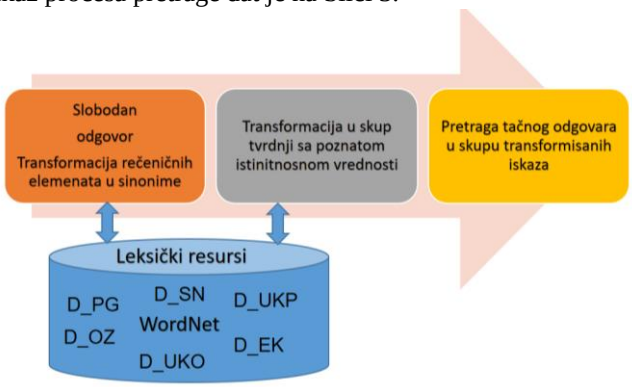
1. postavljanje pitanja
2. provera ispravnosti odgovora.

Kod postavljanja pitanja popularna su zaokruživanja više tačnih odgovora tako da se dubina razumevanja materije može upravo proveriti postavljanjem istog odgovora u nekom od njegovih transformacija unutar logičkog kvadrata. Slika 2. prikazuje proces formiranja liste tvrdnji sa poznatom istinitnosnom vrednošću koje mogu da učestvuju u ispitnim pitanjima.



Slika 2. Proces dopune baze sa tvrdnjama za ispitivanje

Provera ispravnosti odgovora se najlakše rešava zaokruživanjem što je na osnovu poznatih istinitnosnih vrednosti procesom formiranja pitanja moguće. Ukoliko se ispitaniku dozvoli unos slobodnog odgovora u šablon tipa A, E,I ili O mora se proveriti ekvivalentnost odgovora sa datim ključem (tačnim odgovorom zadatim od strane ispitivača). Zbog postojanja različitih oblika za izražavanje iste stvari ovaj proces će biti malo teži. Ovde se mogu koristiti napravljeni leksički resursi kako bi se dobili različiti oblici kvantifikatora, simbola negacije, odnosnih zamenica čijom zamenom se neće promeniti odgovor već njegova forma. Takođe za neke dalje korake može se koristi WordNet za srpski jezik [8] za pretragu sinonima i za druge rečenične delove. Ispravan odgovor može biti dat i u nekoj od drugih predikatskih formi pa zato ukoliko se transformacijama može dobiti odgovor koji se poklapa sa ključem i pritom je tačan pa se može prihvatiti. Prikaz procesa pretrage dat je na Slici 3.



Slika 3. Proces provere da li dati odgovor ekvivalentan rešenju

Primenu predloženih algoritama demonstriraćemo na primeru transformacije tačnog iskaza “Svaki prirodan broj je pozitivan.”, koji se može ubaciti u šablon tipa A. Tabela 2 prikazuje rezultate primene različitih transformacija na ovaj polazni iskaz. Kao rezultat dobijaju se tačni iskazi tipa I kao što su na primer:

- Neki prirodan broj je pozitivan.
- Bar jedan prirodan broj je pozitivan.
- Postoji prirodan broj takav da je pozitivan.
- Postoji prirodan broj koji je pozitivan.

Primenom transformacije AN(+UP) dobijamo netačne iskaze tipa E i O kao što su:

- Svaki prirodan broj nije pozitivan.
- Neki prirodan broj nije pozitivan.
- Postoji prirodan broj takav da nije pozitivan.

TABELA II
PRIMER PRIMENE PRIKAZANOG ALGORITMA NA ISKAZ TIPA A

REČENIČNI ČLAN	ULAZ	IZLAZ			
		TRANSFORMACIJA			
		AN	UP (BEZ OZ)	UP (SA OZ)	AN+UP
KVANTIFIKATOR	Svaki	Svaki	Neki, Bar jedan,…	Postoji, Postoji jedan,…	Neki, Postoji*, …
SUBJEKAT	prirodan broj	prirodan broj			
ODNOSNA ZAMENICA					
SIMBOL NEGACIJE				takav da, koji	[takav da, koji]*
POMOĆNI GLAGOL	je	nije	je	je	nije
PREDIKAT	pozitivan.	pozitivan.			
LOGIČKA VREDNOST ISKAZA	T	N	T	T	N

 ISBN 978-86-7466-930-3

IV. ZAKLJUČAK

U ovom radu je predstavljen sistem za automatizaciju testova za proveru znanja na srpskom jeziku baziran na transformaciji predikatskih iskaza. Predikatski iskazi iz logičkog kvadrata se predloženim transformacijama mogu preslikati jedni u druge. Ova osobina je primenjena na tvrdnje u testovima znanja koje se transformacijama mogu pretvoriti u druge oblike sa poznatom istinitnosnom vrednosti i kao takvi mogu se uključiti u testove. Doprinos ovog rada su i kreirani rečnici kvantifikatora u različitim oblicima pojavljivanja kao i metode za transformaciju. Sinonimi su jedan od velikih problema pri obradi prirodnog jezika. Primena obrade sinonima prilikom validacije slobodnih odgovora jeste jedan od sledećih puteva razvoja sistema. Primena De Morganovih pravila i drugih tautologija takođe može proširiti skup ekvivalentnih iskaza.

ZAHVALNICA

Ovaj rad je delemično podržan od strane Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije po projektu III44007.

REFERENCES

[1] Chowdhury, G. G. (2003). Natural language processing. Annual review of information science and technology, 37(1), 51-89.
[2] Stanić, R. Kvantifikacija i negaciju u logici i hrvatskom jeziku, diplomski rad. 2014.
[3] Nebojša Ikodinović, Uvod u matematičku logiku, Beograd 2015
[4] Peters, S., & Westerståhl, D. (2006). Quantifiers in language and logic. OUP Oxford.
[5] Nermin Okačić, Tautologije i valjane formule kao principi zaključivanja, Prirodno matematički fakultet, Tuzla, 2015. godine.
[6] Ivan Klajn, Gramatika srpskog jezika, Zavod za udžbenike i nastavna sredstva, 2005.
[7] Marovac, U. Avdić, A. Čuljević, N. Metoda za obradu predikatskih iskaza na srpskom jeziku, Kopaonik, 2022.
[8] Krstev, C. Pavlović-Lažetić, G. and Obradović, I. "Using textual and lexical resources in developing serbian wordnet." *Romanian Journal of Information Science and Technology* 7.1-2 (2004): 147-161.

ABSTRACT

By processing natural language, we enable computers to establish communication with humans, to understand human language, but also to transform it and to address human in natural language. The problem of the truth value of statements uttered in natural language is difficult to determine due to the richness of vocabulary and ambiguity of meaning. By mapping the statements of natural language into the statements of predicate logic, it is possible to determine the relationships between different predicate statements as well as to perform the transformation from one form of appearance to another. One of the applications of this knowledge can be in the automated creation of knowledge tests.

System for automation of knowledge verification tests based on transformation of predicate statements

Ulfeta Marovac, Aldina Avdić

 ISBN 978-86-7466-930-3