ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 6, Num. 1, 2021

SOFTWARE SERVICE WITH A PLUG-IN ARCHITECTURE FOR TEXT READABILITY ASSESSMENT

Bohdan Tsebryk, Alexey Botchkaryov

Lviv Polytechnic National University, 12, Bandera Str, Lviv, 79013, Ukraine.
Authors' e-mail*: alb@lp.edu.ua*

https://doi.org/10.23939/acps2021.01.___
Submitted on 01.05.2021

***Abstract* – The problem of developing a software service with a plug-in architecture for assessing the readability of text has been considered. The problem of text readability assessment has been analyzed. Approaches to the development of a software service for text readability assessment have been considered. The structure of the service for text readability assessment has been proposed. The structure of the service has been implemented using the Python programming language and the library Natural Language Toolkit (NLTK). The results of testing the service for text readability assessment have been presented.**

***Index Terms*: text readability, software service, plug-in architecture**

I. INTRODUCTION

The development of modern society, including information technology and digital content, is characterized by a sharp increase in the amount of textual information of different origins and different purposes. Hence the problem of human perception and assimilation of textual information arises. One of the important aspects of this problem is readability - a property of the text that characterizes the ease of its comprehension in the process of reading. Readability depends on both the presentation of the text (for example, in print) and the linguistic features of the text material (for example, the complexity of syntactic constructions or the level of complexity of vocabulary). Among the factors influencing the readability of the text there are the following: the length of words and sentences, the speed of perception of the text, the frequency characteristics of words, the presence of complex words and terms in the text, the level of abstractness of the text, etc. [[1]-[7]].

The first steps in solving the problem of assessing the readability of the text were taken in the early 20th century. At that time, the issues of accessibility of educational texts for students of different years of study were mainly studied. Subsequently, the range of research questions has expanded significantly, and began to include the assessment of the readability of a wide range of textual information for various purposes (technical reports, advertising materials, text content of websites, scientific articles, etc.). At present, research and development of methods and services for assessing the readability of the text are particularly relevant. Many aspects of the problem of text readability assessment are given a lot of attention by researchers [[8]-[16]].

The paper considers the problem of text readability assessment, analyzes approaches to developing a software service for readability assessment, proposes the use of a generalized text readability index, which allows to develop and use more flexible methods of readability assessment by analogy with the concept of multicriteria optimization. The paper considers the structure of the developed service for text readability assessment. The developed structure is implemented using the Python programming language and the Natural Language Toolkit (NLTK) library. Also, in the paper the question of testing the developed service for text readability assessment is considered.

II. THE PROBLEM OF ASSESSING THE READABILITY OF THE TEXT

Currently, there are dozens of methods for assessing the readability of texts in different languages. Among the most commonly used methods and metrics there are the following: Gunning Fog Index, Flesch reading ease test, Flesch-Kincaid grade level, Automated readability index (ARI), Coleman-Liau Index, Dale-Chall readability formula, McLaughlin's SMOG formula etc. [[1]-[16]]. In some countries, such as the United States, legal documents are required to be readable at no higher than the ninth grade (14-15 years) level of comprehension. The Flesch reading ease test is used to assess this.

Assessing the readability of a text is an attempt to assess how easy it is to read and understand a text. Different parameters can be used for such an assessment, for example, the number of syllables in a sentence or the variety (uniqueness) of words in the text. The method of readability assessment includes finding values of key parameters of the text with the subsequent calculation of the result using one or more mathematical formulas.

The most popular metrics for readability assessment are Flesch-Kincaid grade level and Flesch reading ease test, which take into account the number of syllables in 100 words of the text and the average number of words in sentences of the text. These metrics are widely used in various software services that have readability assessment functionality. Over time, linguists conducted more and more

research on readability, and formulas became increasingly complex. Currently, there are more than 200 formulas (metrics) for readability assessment.

There are many different factors that affect how easy it is to read and understand a text. When analyzing a short fragment or a small part of a large text, the question arises as to what extent the obtained linguistic "picture" can indicate the readability of the whole text. For example, in a large text, there are usually more unique words than in a short text. Because different readability metrics are calculated using different text parameters and different mathematical formulas, the assessment results usually differ significantly, depending on the chosen metric. That is why when analyzing texts, it is always useful to apply several readability metrics at once.

III. APPROACHES TO DEVELOPING A SOFTWARE SERVICE FOR TEXT READABILITY ASSESSMENT

Many modern applications implement the function of assessing the readability of the text. In addition, there are specialized services that are designed exclusively for this purpose. These services provide an opportunity to assess readability by different metrics for texts in different languages. These include services that are integrated into the system or implemented as a web browser extension. Some of these services not only evaluate readability, but also offer options to improve the text.

One of the most well-known text readability assessment services is Grammarly. It is also considered one of the best in the field. This service is implemented as an extension for Google Chrome, desktop application and mobile application. The particularity of the Grammarly is that it not only analyzes and evaluates the text, but also corrects errors and advises possible options for improving the text. Grammarly performs analysis on 1) the correctness of the text, namely checks spelling, grammar and punctuation; 2) readability of the text, i.e., how easy it is to understand the text; 3) the reader's interest in the text, i.e., how interesting and effective the text is; 4) the impact of the text on the reader (the level of formality, confidence and friendliness in the text). In addition, Grammarly determines the overall assessment of text quality. It is displayed in the range from 0 to 100, and is calculated based on the following parameters of the text: 1) the number of words, letters and sentences (as well as the time required to read the text); 2) readability (including the length of words, sentences, and result of the Flesch reading ease test); 3) vocabulary of the text (taking into account the uniqueness and rarity of the words of the text in comparison with the texts of other users of the service).

Another very popular service for assessing the readability of text is the Readable. Compared to Grammarly, the Readable is more flexible, but in some respects less efficient. The flexibility of the Readable means the ability to choose how to assess the readability of the text, integration with other software services, the presence of its own API, the ability to scan emails and websites, and more.

IV. GENERALIZED INDEX OF TEXT READABILITY

There are many metrics that reflect different aspects of text readability. It would be convenient to have some integral index that would summarize several different metrics in the form of one value. This would allow the development of more flexible ways to assess readability by analogy with the concept of multicriteria optimization. The idea of using a generalized readability index is that each user can form their own version of the index, taking into account the aspects of readability they need in the form of relevant metrics.

Let's define a generalized text readability index based on such user-selected subset of readability metrics as

$$Q = w_1 \cdot f(m_1) + w_2 \cdot f(m_2) + \ldots + w_N \cdot f(m_N), \quad (1)$$

where,

$M=\{m_1, m_2, \ldots, m_N\}$ is a subset of readability metrics, which are taken into account in the generalized index,

$f(m_i)$ is the normalized value of the metric mi.

For each readability metric in the subset M, the user can assign some weighting factor $w(m_i)$. By default, the weighting factor of the metric mi is equal to:

$$w(m_i) = 1/N, \quad (2)$$

where N is the number of metrics in the subset M.

To calculate the generalized index, the values of all metrics are pre-normalized, i.e., reduced to the range from 0 to 1: $f(m_i) \in [0,1]$.

Different variants of generalized index can be used for different types of texts in the form of different subsets of metrics $\{M_1, M_2, \ldots, M_K\}$ and different sets of weights $\{W_1, W_2, \ldots, W_L\}$, where $W_i=\{w_1, w_2,...\}$. Then define a variant of the generalized index in the form:

$$H=\{M_i, W_j\}, \quad (3)$$

where

M_i is a subset of metrics used to calculate the index,

W_i is a set of corresponding weights of normalized values of metrics M_i.

Different texts can be classified by volume, level of difficulty for perception, level of complexity in terms of the volume of the dictionary used and other parameters. That is, we can consider a set of text parameters $P=\{p_1, p_2,\ldots,p_n\}$, each of which relates the text to a particular class of texts. Based on this, we can offer the following additional function of the service for text readability assessment. The service can recommend to the user this or that variant of the generalized index of readability, depending on the parameters of the text recognized by service and corresponding class of the text:

$$H = F(P), \quad (4)$$

where F is some way to convert a set of text parameters into a variant of the generalized readability index. That is, the service can offer the user a version of the generalized index, which for one reason or another is most suitable for the text of a particular class. To develop the way of converting text parameters into a variant of the

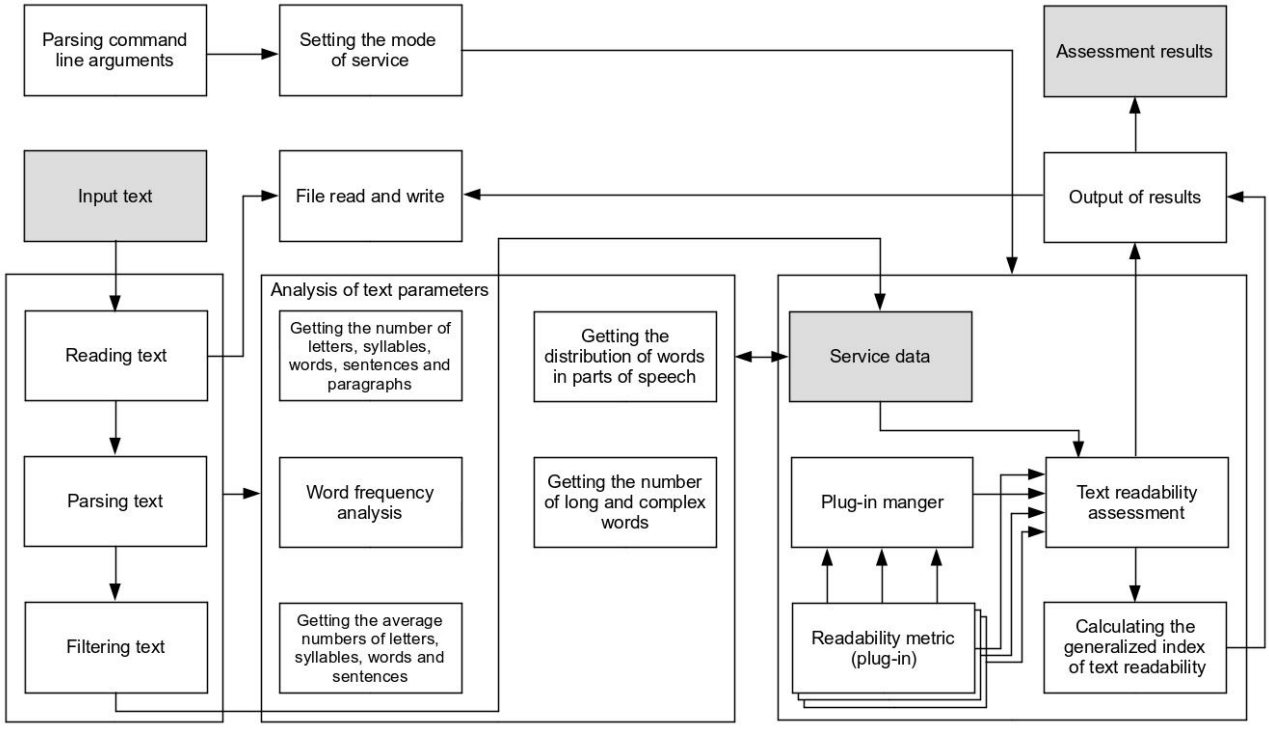

Fig. 1. The structure of the service for text readability assessment.

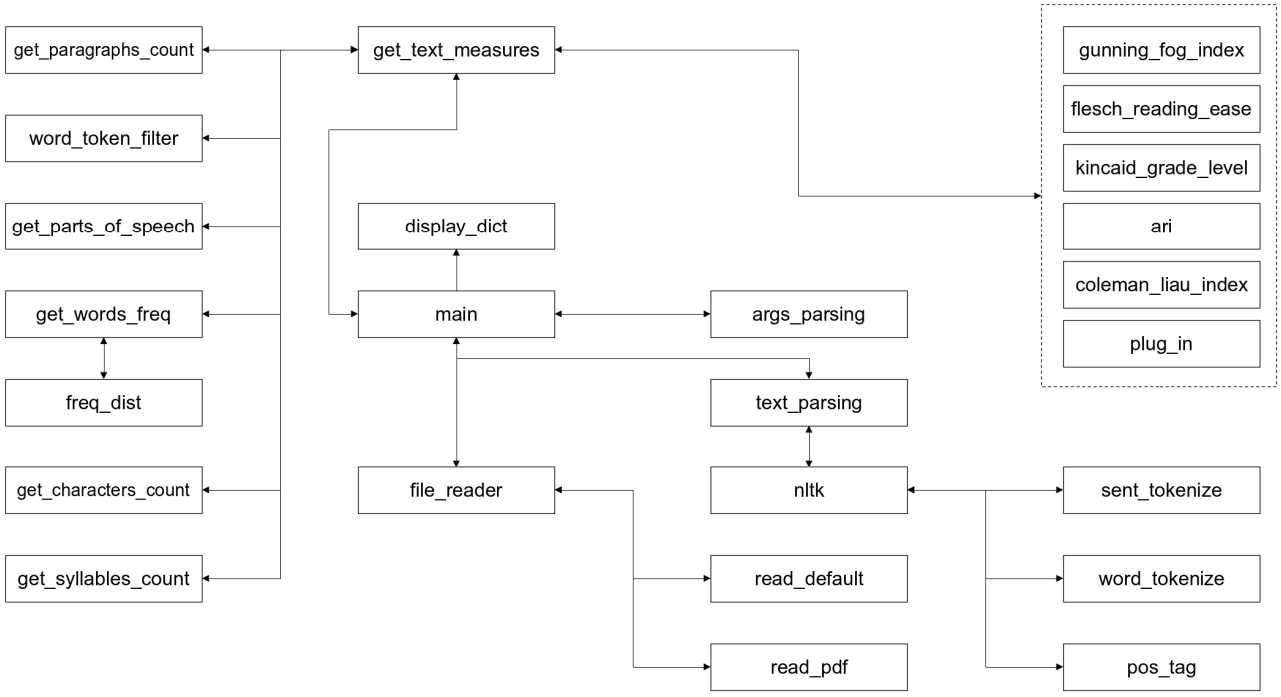

Fig. 2. Function calls diagram.

```
    '|by|of|in|to|near|of|from')
pronoun_en = (
    'i|me|we|us|you|he|him|she|her|it|they'
    '|them|thou|thee|ye|myself|yourself|himself'
    '|herself|itself|ourselves|yourselves|themselves'
    '|oneself|my|mine|his|hers|yours|ours|theirs|its'
    '|our|that|their|these|this|those|your')
words_en = collections.OrderedDict([
    ('Be verbs', re.compile(
        r'\b(be|being|was|were|been|are|is|am)\b', re.IGNORECASE)),
    ('Auxiliary verb', re.compile(
        r"\b(will|shall|cannot|may|need to|would|should"
        r"|could|might|must|ought|ought to|can't|can)\b", re.IGNORECASE)),
    ('Conjunction', re.compile(
        '\\b(%s)\\b' % conjuction_en, re.IGNORECASE)),
    ('Pronouns', re.compile(
        '\\b(%s)\\b' % pronoun_en, re.IGNORECASE)),
    ('Preposition', re.compile(
        '\\b(%s)\\b' % preposition_en, re.IGNORECASE)),
    ('Nominalization', re.compile(
        r'\b\w{3,}(tion|ment|ence|ance)\b', re.IGNORECASE | re.UNICODE)),
])
```

Fig. 3. Data structures used in the text parameter analysis module.

```
--------------------
 TEXT INFO
--------------------
Average number of characters per word : 5.446280991735537
Average number of syllables per word : 1.7768595041322315
Average number of words per sentence : 24.2
Sentences per paragraph : 2.5
Number of characters : 659
Syllables : 215
Number of words : 121
Unique words : 93
Number of sentences : 5
Number of paragraphs : 2
Number of long words : 42
Number of complex words : 20
10 most used words:
        "the : 6
        "have : 4
        "a : 4
        "killed : 3
        "of : 3
        "in : 3
        "on : 3
        "including : 2
        "been : 2
        "week : 2
```

Fig. 4. Test results of the analysis modules of text parameters: output of text parameters.

generalized readability index one can use ideas from the recommender systems [[23]-[30]], context-aware computing [[31]-[34]], and machine learning [[35]-[38]].

V. THE STRUCTURE OF THE SERVICE FOR TEXT READABILITY ASSESSMENT

Developed service for assessing the readability of the text is based on a plug-in architecture [[17]-[22]]. This solution allows one to add to the service new readability metrics in the form of separate plug-ins (Fig. 1. The structure of the service for text readability assessment.). Therefore, the user can extend the functionality of the service using third-party plug-ins or plug-ins developed by him. The developed structure is implemented using the Python programming language (Fig. 2. Function calls diagram.) using the software library Natural Language Toolkit (NLTK).

Among the main modules that are part of the service, we can distinguish the following groups of modules.

1) Text processing modules, which include: reading input text, parsing input text, filtering input text. The text reader can work with 4 file types (.txt, .pdf, .odt, .docx) using the *textract* library. The *PyPDF2* library is used to read pdf-files. The parsing module breaks the input text into sentences and the sentences into tokens using the *NLTK* library. Token tagging is also performed in this module.

2) Modules for analyzing the parameters of the input text, which in particular determine the number of letters, syllables, words, sentences and paragraphs in the text, determine the distribution of words in parts of speech (Fig. 3. Data structures used in the text parameter analysis module., Fig. 5. Test results of the text parameters analysis modules: output of word parameters.), perform frequency analysis of words, determine the number of long and complex words in the text, determine the average quantitative indicators (average number of letters per word, average number of syllables per word, average number of words per sentence, average number of sentences per paragraph).

3) Modules of text readability assessment, which include: text readability metrics (in the form of separate plug-ins), text readability assessment (according to user-specified metrics), calculation of generalized index of text readability (according to the user-selected index variant). The following text readability metrics are implemented in the service: Gunning Fog Index, Flesch reading ease test, Flesch-Kincaid grade level, Automated readability index (ARI), Coleman-Liau Index [[1]-[7]].

4) Auxiliary modules, which include: parsing command line arguments, setting the service mode, reading and writing to files, plug-in manager.

VI. TESTING THE SERVICE FOR TEXT READABILITY ASSESSMENT

The operation of the text readability assessment service was tested in two aspects: correctness of work (detection of errors) and productivity of operation with input texts of different volume. During testing, the absence of critical errors in the service and acceptable performance indicators of its operation were confirmed. Testing was performed modularly. The black box testing method was used. Testing the service for errors helped to improve the performance of corresponding algorithms and to perform optimization of program code. Fig. 4. Test results of the analysis modules of text parameters: output of text parameters. and Fig. 5. Test results of the text parameters analysis modules: output of word parameters. show the results of testing the modules of analysis of text parameters. Fig. 6. Test results of the text readability assessment modules. shows the results of testing modules for assessing the readability of the text.

```
--------------------
 PARTS OF SPEECH
--------------------
Pronouns : 2
Modals : 2
Conjunctions : 2
Foreign words : 0
Prepositions : 0
Cardinal digits : 5
Adverbs : 1
Verbs : 28
Nouns : 41
```

Fig. 5. Test results of the text parameters analysis modules: output of word parameters.

```
--------------------
 READABILITY GRADES
--------------------
Kincaid : 14.814942148760334
ARI : 16.321983471074383
Coleman-Liau : 14.999901859504131
Flesch Reading Ease : 31.949685950413254
Gunning Fog Index : 16.291570247933887
```

Fig. 6. Test results of the text readability assessment modules.

VII. CONCLUSION

The problem of developing a software service with a plug-in architecture for assessing the readability of text was considered. The problem of text readability assessment was analyzed. Approaches to the development of a software service for text readability assessment were regarded. The structure of the service for text readability assessment was proposed. The structure of the service was implemented using the Python programming language and the library Natural Language Toolkit (NLTK). The results of testing the service for text readability assessment were presented.

References

[1] Gunning, R. (1952) *The Technique of Clear Writing*. McGraw-Hill, pp. 36–37.

[2] Flesch, R. (1948) A new readability yardstick. *Journal of Applied Psychology*. 32 (3), pp. 221–233.

[3] Farr, J.N., Jenkins, J.J. and Paterson, D.G. (1951) Simplification of Flesch Reading Ease Formula. *Journal of Applied Psychology*. 35 (5), pp. 333–337.

[4] McClure, G. (1987) Readability formulas: Useful or useless. (an interview with J. Peter Kincaid). *IEEE Transactions on Professional Communication*. 30, pp. 12–15.

[5] Kincaid, J.P., Aagard, J.A., O'Hara, J.W. and Cottrell, L.K. (1981) Computer Readability Editing System. *IEEE Transactions on Professional Communication*. 24 (1), pp. 38–42.

[6] Senter, R.J. and Smith, E.A. (1967) *Automated Readability Index*. Aerospace Medical Research Laboratories, University of Cincinnati, Wright-Patterson Air Force Base, Ohio, AMRL-TR-66-20. - 14 p.

[7] Coleman, M. and Liau, T. L. (1975) A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, Vol. 60, pp. 283–284.

[8] Zhou, S., Jeong, H. and Green, P. (2017) How Consistent Are the Best-Known Readability Equations in Estimating the

Readability of Design Standards?. IEEE Transactions on Professional Communication, vol. 60, no. 1, pp. 97-111.
[9] Karmakar, S. and Zhu, Y. (2010) Visualizing multiple text readability indexes. In: *2010 International Conference on Education and Management Technology*, pp. 133-137.
[10] Karmakar, S. and Ying Zhu (2010) Visualizing text readability. In: 2010 6th International Conference on Advanced Information Management and Service (IMS), pp. 291-296.
[11] Iram, N., Zafar, S. and Zahra, R. (*2018*) Web content readability evaluation using fuzzy logic. In: International Conference on Advancements in Computational Sciences (ICACS), pp. 1-8.
[12] Antunes, H. and Lopes, C. (2019) Readability of web content. In: 2019 14th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1-4.
[13] Naderi, B., Mohtaj, S., Karan, K. and Möller, S. (2019) Automated Text Readability Assessment for German Language: A Quality of Experience Approach. In: 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), pp. 1-3.
[14] Tra My, H., N., Suzuki, S. and Miyazaki, Y. (2017) Building Personalized Readability Equation and Personalized English Vocabulary List for Continued Study". In: 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), pp. 791-795.
[15] Qumsiyeh, R. and Ng, Y. (2011) ReadAid: A Robust and Fully-Automated Readability Assessment Tool. In: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, pp. 539-546.
[16] Liu, Y., Ji, M., Lin, S., Zhao, M. and Lyv, Z. (2021) Combining Readability Formulas and Machine Learning for Reader-oriented Evaluation of Online Health Resources. IEEE Access, vol. 9.
[17] Decasper, D., Dittia, Z., Parulkar, G. and Plattner, B. (2000) Router plugins: a software architecture for next-generation routers. *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 2-15.
[18] Zhu, J., Yin, Q., Zhu, R., Guo, C., Wang, H. and Wu, Q. (2008) A Plugin-Based Software Production Line Integrated Framework". In: *International Conference on Computer Science and Software Engineering*, pp. 562-565.
[19] Schleinzer, B., Cabac, L., Moldt, D. and Duvigenau, M. (2008) From Agents and Plugins to Plugin-Agents, Concepts for Flexible Architectures. In: *New Technologies, Mobility and Security*, pp. 1-5.
[20] Adhikari, S. and Jones, B. (2019) A Modular Plugin Architecture for Literate Programming Editors". In: *Proceedings of the of 2019 SoutheastCon (IEEE Region 3 Technical, Professional, and Student Conference)*, pp. 1-4.
[21] Minh Vu and Thompson, C. (2005) E2 agent plugin architecture. In: *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pp. 26-31.
[22] Bako, B., Borchert, A., Heidenbluth, N. and J. Mayer (2006) Linearly Ordered Plugins through Self-Organization. In: *International Conference on Autonomic and Autonomous Systems (ICAS'06)*.
[23] Ricci, F., Rokach, L., Shapira, B. and Kantor, P. (eds.) (2015) *Recommender Systems Handbook*. 2nd ed., Springer. - 1020 p.
[24] Aggarwal, C. (2016) *Recommender Systems: The Textbook*. Springer. - 519 p.
[25] Schrage, M. (2020) *Recommendation Engines*. The MIT Press. - 296 p.
[26] Falk, K. (2019) *Practical Recommender Systems*. Manning Publications. - 432 p.
[27] Robillard, M., Maalej, W., Walker, R. and Zimmermann, T. (eds.) (2014) *Recommendation Systems in Software Engineering*. Springer-Verlag Berlin Heidelberg. - 560 p.
[28] Jannach, D. (2010) *Recommender Systems: An Introduction*. Cambridge University Press. - 352 p.
[29] Jie Lu, Qian Zhang, Guangquan Zhang (2020) *Recommender Systems: Advanced Developments*. WSPC. - 362 p.
[30] Suresh Kumar Gorakala (2017) *Building Recommendation Engines.* Packt Publishing. - 357 p.
[31] Schilit, B., Adams, N. and Want, R. (1994) Context-aware computing applications. In: *Proceedings of the IEEE Workshop on "Mobile Computing Systems and Applications"*, IEEE Computer Society, pp. 85-90.
[32] Perera, C., Zaslavsky, A., Christen, P. and Georgakopoulos, D. (2014) Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, First Quarter, pp. 414-454.
[33] Grifoni, P., D'Ulizia, A., and Ferri, F. (2018) Context-Awareness in Location Based Services in the Big Data Era, In: Skourletopoulos, G., Mastorakis, G., Mavromoustakis, C., Dobre C. and Pallis, E. (eds.) *Mobile Big Data. Lecture Notes on Data Engineering and Communications Technologies*, Springer, vol. 10, pp. 85–127.
[34] Capurso, N., Bo Mei, Tianyi Song and Xiuzhen Cheng (2018) A survey on key fields of context awareness for mobile devices. *Journal of Network and Computer Applications*, Volume 118, pp. 44-60.
[35] Sutton, R.S., Barto, A.G. (2018) *Reinforcement Learning: An Introduction*. 2nd ed., A Bradford Book. - 532 p.
[36] Weber, C., Elshaw, M. and Mayer, N. (eds.) (2008) Reinforcement Learning: Theory and Applications. Vienna: I-Tech Education and Publishing. - 424 p.
[37] Wiering, M., van Otterlo, M. (eds.) (2012) *Reinforcement Learning: State-of-the-Art*. Springer. - 672 p.
[38] Bertsekas, D. (2019) *Reinforcement Learning and Optimal Control*. Athena Scientific. - 388 p.

Bohdan Tsebryk was born in 2000 in Lviv, Ukraine. Currently, he is a student of the Computer Engineering Department at Lviv Polytechnic National University. He has been doing scientific and research work since 2020. He has two years of professional experience working in IT as a Quality Assurance Engineer. His research interests include software architecture, computational linguistics and natural language processing.

Alexey Botchkaryov was born in 1975 in Lviv, Ukraine. He received the B.S. and M.S. degrees in computer engineering from Lviv Polytechnic National University, in 1998 and the Ph.D. degree in computer systems and components at Lviv Polytechnic National University in 2019. He has been doing scientific and research work since 1994. Currently, he is an associate professor at the Computer Engineering Department, Lviv Polytechnic National University. His research interests include self-organization in complex systems, structural adaptation, intelligent information-gathering agents and multi-agent systems.