Tools for Creating and Maintaining a Knowledge Base by Integrating Wolfram Mathematica System and Nevod Package

Valery B. Taranchuk
Department of Computer Applications and Systems
Belarusian State University
Minsk, Republic of Belarus
taranchuk@bsu.by

Vladislav A. Savionok
Department of Software for Information Technologies
Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus
v.savenok@bsuir.by

***Abstract*—One of the outcomes of the current review of the state of knowledge base design and analysis technologies, software and hardware platforms for the implementation of semantically compatible intelligent computer systems is the conclusion about the need to formulate the principles of collective design, development, verification of knowledge bases. Accordingly, it is important not only to formulate and justify the theory, to formalize the requirements, but also to develop tools to represent such formal theory in the form (format) of the knowledge base of the corresponding scientific knowledge portal. It is this concept that is the goal of implementation and development of the OSTIS Ecosystem, which, in particular, is intended to solve problems of convergence and merging of functional properties of systems of different classes; range expansion, organizational and technical unification, realization of coordination of software, computing and telecommunication means; unification of intelligent computer systems. A special place in such unification should be given to the solution of problems of integration of OSTIS Ecosystem tools with computer mathematics systems, especially with computer algebra systems.**

This paper presents an example of integration of the local intelligent computer system based on Nevod library with the knowledge base of Wolfram Mathematica computer algebra system, which can be interpreted as an analogue of the integration of knowledge bases of the corporate OSTIS-system into the OSTIS Ecosystem. Examples of the use of tools to analyze the local knowledge base, its transfer from virtual to real status are presented and explained.

***Keywords*—Semantic analysis, OSTIS technology, Wolfram Mathematica, Wolfram Knowledgebase, Entity, temporal markers, Nevod**

I. Introduction

According to the assessment of the current state of the field of artificial intelligence (AI), given in [1], there is an active development of many different areas, such as formal ontologies, artificial neural networks, machine learning, multi-agent systems, etc. However, this activity does not bring an aggregate increase in the level of intelligence of modern intelligent computer systems (ICS). This is due to the current isolation between methods and designing tools in each of the areas of AI. The solution of this problem is seen in the construction of a general formal theory of ICS, and designing a comprehensive technology for their development and life cycle support — the OSTIS technology [1]. This will allow to achieve convergence of all areas of AI through their mutual integration and joint development.

Modern design support frameworks in the field of AI are mainly aimed at the development of highly specialized solutions, which can act as individual components of the ICS. In order to obtain guaranteed compatibility of all developed components, it is necessary to transform these tools into a unified technology for comprehensive design and support of the full life cycle of the ICS. Despite the independent development of the ICS, special attention should also be paid to their external interfaces, since the intercommunication of ICS between each other will be required as part of complex systems for the automation of various human activities. In other words, there is a need for unification and convergence of next-generation ICS, along with their components. This will open the way to the design of optimized complexes that include all the necessary AI to solve the tasks at hand. It is important to note that in order to meet the optimization requirements when solving certain tasks and achieve maximum performance, it is necessary to organize the effective interaction of the ICS with the connected information resources. The main actions for solving the key methodological problems, which are the reason for the current state of the field of AI, are also given in [1]. Note that similar problems are solved in the field of computer algebra systems: in their design, development, content update and functionality expansion [2], [3].

Methodological and technical solutions for integration of various types of knowledge implemented in the computer algebra system Wolfram Mathematica (WM), Wolfram Language (WL) are described in this paper.

The software solutions implemented in the Wolfram Knowledgebase are marked and illustrated with examples. From the standpoint of the need of unification of next-generation ICS, integration with the Wolfram Knowledgebase is performed on the general basis, meaning that the same method can be applied to integrate the WM with other knowledge systems, including the OSTIS Metasystem. The examples in this paper demonstrate several methods of integration of various tools implemented in Wolfram Mathematica computer algebra system, and by means of independent library Nevod [4].

II. Creating a thematic block for temporal markers analysis

A. Temporal markers analysis

One of the main directions in the field of text processing is the extraction of their semantic component — semantic analysis. In this direction a number of problems are solved, such as document search in local and global networks, automatic annotation and abstracting, classification and clustering of documents, synthesis of texts and machine translation, text tone analysis, and fact extraction (publications mentioned in [5]).

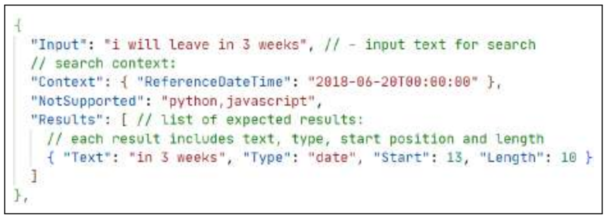
An integral part of the task of extraction of facts and determination of relations between objects is localization in time of the event corresponding to the fact. The information, allowing to localize the event on a time axis, is transferred by means of various in the form and content textual expressions — temporal markers (pointers). The final result of the extraction of temporal markers from the text is their representation and interpretation within the framework of the formal model set in the process of semantic analysis [6].

To solve the problem of extracting temporal references from text the toolkit of one of the leaders in the field of entity recognition Microsoft.Recognizers.Text [7] is widely used.

B. Preparation of data for the thematic block of temporal markers analysis

The MS Recognizers Text (MRT) library provides the ability to recognize entities in texts of various languages and is widely used in Microsoft products, for example: in pre-defined templates for LUIS (Language Understanding Intelligent Service), in the platform for creating dialog bots Power Virtual Agents [8], in cognitive language services in Azure cloud infrastructure — NER (Named Entity Recognition). The library is distributed under an open source and free software license from MIT; along with the source code in the repository on GitHub [7] test dataset for different languages are published.

The Microsoft.Recognizers.Text.DateTime module, and in particular its BaseDateExtractor component, is used in MRT to search for temporal markers in the text. This component corresponds to a test dataset represented in JSON format – the DateExtractor.json file [9]. The dataset contains 143 elements that include absolute and relative dates in different forms, as well as metainformation, which is used to check the correctness of the extraction results. A search context, a reference date that indicates the point in time used to translate relative temporal markers into absolute ones, can be attached to the dataset element. An example of a test dataset element with comments is shown in Fig. 1.



```
{
  "Input": "i will leave in 3 weeks", // - input text for search
  // search context:
  "Context": { "ReferenceDateTime": "2018-06-20T00:00:00" },
  "NotSupported": "python,javascript",
  "Results": [ // list of expected results:
    // each result includes text, type, start position and length
    { "Text": "in 3 weeks", "Type": "date", "Start": 13, "Length": 10 }
  ]
},
```

Figure 1. Example of a test dataset element.

In [5] the results of comparing the capabilities in temporal pointers extraction of MRT and Nevod library [4], which implements the search method in the text [10], are described. For this purpose two software modules were developed: mMRT and mNevod, which provide search and extraction of temporal markers from text. Comparative testing of the software modules was performed on the described test dataset using the means of the computer algebra system Wolfram Mathematica to analyze the results.

Input data for mNevod and mMRT modules is *Input* string. The results of temporal pointer extraction modules are compared with the *Results* dictionary, which contains the expected position in the text, length and contents of the extracted temporal pointer. The DateExtractor test dataset is used to confirm the functional completeness of the libraries that extract temporary pointers from text.

When checking and tuning fact extraction tools, in particular temporal pointers, an important position to evaluate is the focus on recognition rather than unambiguous identification of entities in the text. The original DateExtractor test dataset of the MRT library does not allow to fully analyze the functionality of corresponding tools of this type — it covers most variants of dates writing in English, includes common abbreviations, but does not take into account the possibility of distortion of the input text. It seems advisable to compile a new test dataset that takes this aspect into account when evaluating fact extraction tools. The methodology for forming a representative test dataset is outlined in [5].

C. Using Wolfram Mathematica to form a test dataset

Focusing on the tools for extracting temporal markers in the text, using fragments from DateExtractor, a new test dataset was prepared. In the resulting dataset of 141 items, distortions (errors) most typical for manual typing are introduced, so that they affect the text fragments that represent the target for extraction.

The following types of situations have been selected as typical manual input errors that do not affect the word length:

1) replacing a single letter;
2) transposing a pair of neighboring letters in a word.

It should be noted that when modeling distortions of type 1, there is a natural heuristic to limit the set of letters that can be used incorrectly instead of a given correct letter. This is based on the assumption that the standard means of entering text data for computers, the keyboard, is used. In this case the most common substitutions will be the neighboring letters by the location of the keys on the keyboard. For example, for the word "Monday" one of the most frequent variants of such an error for the QWERTY layout is the replacement of the letter "d" by the letter "s" — "Monsay".

When modeling errors of type 2, a similar natural heuristic can be applied. Taking into account the blind printing method, it is logical to assume that most often a transposition error will occur for characters, which are typed by fingers of different hands [5]. For example, for the word "Sunday" a variant of such distortion in the QWERTY layout is the transposition of the letters "a" and "y" — "Sundya".

The listed types of input text distortions can be multiplied and combined: one word can contain several errors of the same type, or errors of several types simultaneously. In [5] the process of modeling each type of errors separately is described, without taking into account their combinations. According to each type of errors, the following distortions are included in the resulting set of 141 items:

- the letter "d" was replaced with the letter "s" in the word "monday" (corresponding to 2.8% of the set, the total word is contained in 3.5% of the set);
- the letters "a" and "y" in the word "sunday" have been rearranged (corresponding to 5.7% of the set, the total word is contained in 7% of the set).

When evaluating the correctness of the processing of the obtained dataset by mNevod and mMRT software modules, identical results were obtained: 91.4%. Due to the extensibility of templates in the Nevod package, rules were added to level out the corresponding erroneous situations. For letter substitution recognition, different variants of distortion of the word "monday" at the position corresponding to the letter "d" were introduced. A rule covering possible permutations of neighboring letters of the word "sunday" has been added. Taking into account the previously described heuristics, these are the variants "sundya" and "usnday". The software module using the Nevod library with the updated rule set correctly processed 100% of the test dataset. Thus, the use of the Nevod library tools allows to adapt the software module for different variants of input data distortions by making local changes in the existing pattern sets. The application of the proposed heuristics when composing new rule sets will allow to implement the initial processing of typical input errors.

Wolfram Mathematica tools usage. To compare the functionality of the mNevod and mMRT modules when solving the problem of extracting temporal pointers in text not only from the DateExtractor test dataset, but also by forming other representative datasets, Wolfram Mathematica has developed the mDataWM service application. It contains software tools that enable you to separate the dataset to be processed from the meta-information, evaluate and compare the quality of the results of processing a modified dataset with mMRT and mNevod modules, distort any dataset, and test the performance of the libraries. The mDataWM application provides for creating test datasets in any language and analyzing the results of their processing. The tools of the mDataWM application implement the following functions:

- distort initial dataset and form a modified one;
- import/export to interface Mathematica with the mMRT and mNevod modules (handling files and separating data from meta-information);
- evaluate the quality of the results of dataset processing.

The following Mathematica kernel functions are used in mDataWM:

- $Import[source]$ — imports data from $source$, returning a Wolfram Language representation of it.
- $Export[dest, expr, "format"]$ — exports $data$ in the specified format $"format"$.
- $Map[f, expr]$ — applies f to each element on the first level in $expr$.
- $MapIndexed[f, expr]$ — applies f to the elements of $expr$, giving the part specification of each element as a second argument to f.
- $Association[key_1->val_1, key_2->val_2, ...]$ — represents an association between keys and values.
- $AssociateTo[a, key->val]$ — changes the association a by adding the key-value pair $key->val$.
- $SortBy[list, f]$ — sorts the elements of $list$ in the order defined by applying f to each of them.
- $KeyMemberQ[assoc, form]$ — yields True if a key in the association $assoc$ matches $form$, and False otherwise.
- $KeyDrop[assoc, \{key_1, key_2, ...\}]$ — yields an association from which elements with keys key_i have been dropped.
- $KeyTake[assoc, \{key_1, key_2, ...\}]$ — yields an association containing only the elements with keys key_i.
- $RandomSample[\{e_1, e_2, ...\}, n]$ — gives a pseudorandom sample of n of the e_i.

- $Select[list, crit]$ — picks out all elements e_i of list for which $crit[e_i]$ is True.
- $Delete[expr, n]$ — deletes the element at position n in $expr$. If n is negative, the position is counted from the end.
- $StringReplace[''string'', s->sp]$ — replaces the string expression s by sp wherever it appears in $''string''$.
- $Count[list, pattern]$ — gives the number of elements in $list$ that match $pattern$.

In the next examples, the original test dataset is extracted from WDR, and on its basis the correctness of temporal pointer extraction and target search pattern processing tools (mMRT tool based on MS Recognizers and mNevod tool based on Nevod) are tested. The results of the check can also be uploaded to WDR.

WDR supports the ability to work in parallel with multiple programs, nodes, clients, which allows you to organize a kind of (Mass Servicing System):

- one client uploads dataset items to the WDR;
- another client retrieves the set and runs the mNevod tool, uploading the results back into WDR;
- the third client reads the set in parallel and runs the mMRT tool, and uploads the results back into WDR.

D. Creation of a thematic block, inclusion of temporal markers analysis tools in the WDR

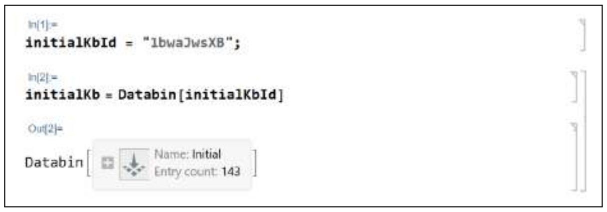
The upload of previously prepared data from other Information Resources into the WDR is implemented. The existing data can be modified on any computer using any tool or with Wolfram Mathematica (or WolframAlpha) toolsets. In particular, in the arsenal of tools from WM most often used functions are as follows: lists manipulation, imposition of various kinds of noise and distortions by random number generators.

WDR creation, data downloading, limitations of the free version. $CreateDatabin[]$ creates a databin in the Wolfram Data Drop and returns the corresponding $Databin$ object *(CreateDatabin[options] creates a databin with the specified options)*. When creating a WDR, it is possible to pre-define the semantics of the data that will be contained in this databin [11]. As a result of executing the code

$initialKb = CreateDatabin[];$

the thematic block shown in Fig. 2 and Fig. 3 is created. When creating a new block, the system assigns an identifier to it. Using the identifier, basic operations on uploading, selecting and deleting data can be performed with the block. In our case, the obtained identifier is placed in the variable $initialKb$, with which we operate further.

The free version of WM has a number of restrictions on the use of WDRs. In particular, the size of one element

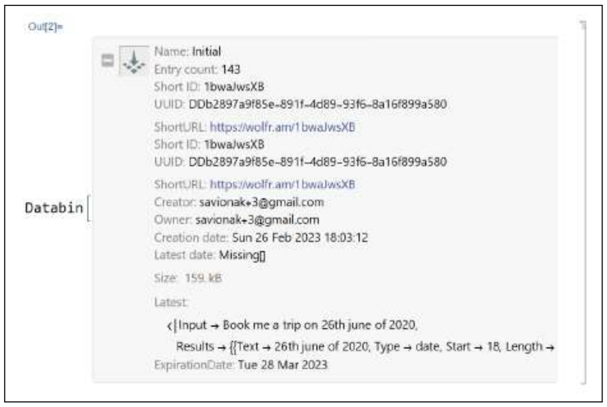


```
In[1]=
initialKbId = "1bwaJwsXB";

In[2]=
initialKb = Databin[initialKbId]

Out[2]=
Databin[ Name: Initial
         Entry count: 143 ]
```

Figure 2. The result of creating a thematic block.



```
Out[2]=
Databin[ Name: Initial
         Entry count: 143
         Short ID: 1bwaJwsXB
         UUID: DDb2897a9f85e-891f-4d89-93f6-8a16f899a580
         ShortURL: https://wolfr.am/1bwaJwsXB
         Short ID: 1bwaJwsXB
         UUID: DDb2897a9f85e-891f-4d89-93f6-8a16f899a580
         ShortURL: https://wolfr.am/1bwaJwsXB
         Creator: savionak+3@gmail.com
         Owner: savionak+3@gmail.com
         Creation date: Sun 26 Feb 2023 18:03:12
         Latest date: Missing[]
         Size: 159 kB
         Latest:
           <|Input → Book me a trip on 26th june of 2020,
             Results → {{Text → 26th june of 2020, Type → date, Start → 18, Length →
         ExpirationDate: Tue 28 Mar 2023
```

Figure 3. Detailed information about a thematic block.

in WDR cannot exceed 25 KBytes. Because of this, each element of the test dataset in the example is uploaded separately. Another restriction is applied to the frequency of uploads – no more than 60 per hour. To circumvent this limitation, the original test dataset is divided into parts of 60 or less items, and then each part is uploaded at one hour intervals. The code used to upload the test dataset into WDR is –

(* first batch of 60 elements: *)
$specsToUpload = Take[specs, \{1, 60\}];$
(* second batch of 60 elements: *)
$specsToUpload = Take[specs, \{61, 120\}];$
(* third batch of elements: *)
$specsToUpload = Take[specs, \{121, 142\}];$
(* upload batches, one hour interval: *)
$DatabinUpload[initialKb, specsToUpload];.$

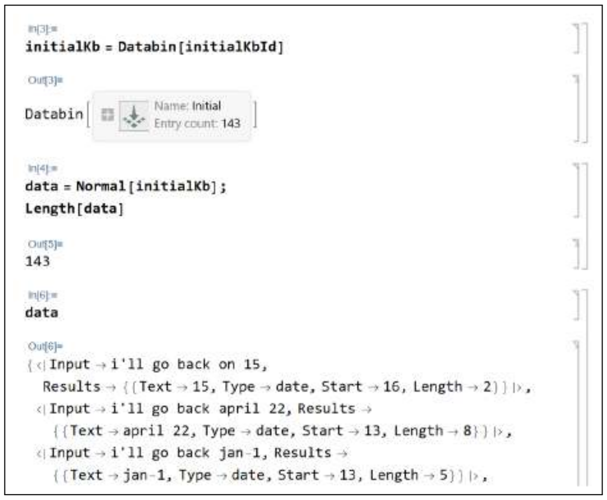
The $Take$ function is used *($Take[list, n]$ gives the first n elements of list; $Take[list, -n]$ gives the last n elements of list; $Take[list, \{m, n\}]$ gives elements m through n of list)* to divide the dataset into parts. Each part is uploaded separately using the $DatabinUpload$ function *($DatabinUpload[bin, \{entry_1, entry_2, ...\}]$ bulk uploads all the entries $Subscript[entry, i]$ to a databin; $DatabinUpload[bin, EventSeries[...]]$ bulk uploads all entries in an event series to a databin)*.

In WM, it is possible to add a single item to the WDR – for this purpose the $DatabinAdd$ function is used. In the following example, the last element is uploaded separately:

$DatabinAdd[initialKb, specs[[143]]];.$

The above steps are given as one of the options that allow you to bypass the restrictions imposed, while maintaining the simplicity of retrieving the uploaded test dataset. Another option, which is more efficient in terms of storage and time spent on uploading the elements of the dataset, is to build a hierarchy. It was noticed, that each element in the previously described procedure does not exceed the size of 2 KBytes. Thus, by combining the elements in a two-level hierarchy, in batches of approximate size of 10, you can significantly reduce the upload time of the entire dataset. However, in this case, the extraction procedure becomes more complicated, because it is necessary to perform additional transformations and flatten the hierarchy, for example, using the $Flatten$ function *(*$Flatten[list]$ *– flattens out nested lists;* $Flatten[list, n]$ *– flattens to level* n*;* $Flatten[list, n, h]$ *– flattens subexpressions with head* h*).*

Extracting data from the WDR. Data extraction from the WDR is performed using the functions $Databin$ *(represents a databin in the Wolfram Data Drop)* [12] and $Normal[expr]$ *(converts* $expr$ *to a normal expression from a variety of special forms).* An example of getting the full content of a thematic block is shown in Fig. 4. Examples of obtaining part of the content with a given element extraction step are shown in Fig. 5 and Fig. 6.



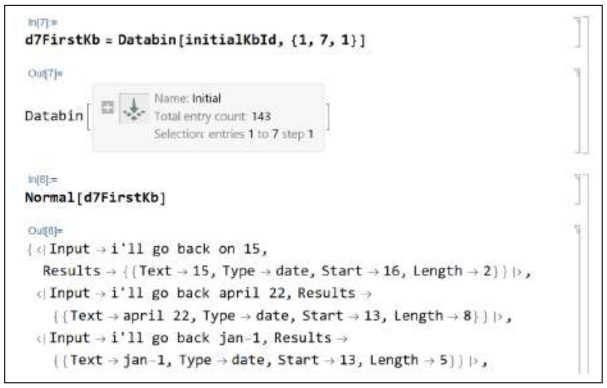
```
In[3]:=
initialKb = Databin[initialKbId]
Out[3]=
Databin[ Name: Initial
         Entry count: 143 ]
In[4]:=
data = Normal[initialKb];
Length[data]
Out[5]=
143
In[6]:=
data
Out[6]=
{<|Input → i'll go back on 15,
  Results → {{Text → 15, Type → date, Start → 16, Length → 2}}|>,
 <|Input → i'll go back april 22, Results →
   {{Text → april 22, Type → date, Start → 13, Length → 8}}|>,
 <|Input → i'll go back jan-1, Results →
   {{Text → jan-1, Type → date, Start → 13, Length → 5}}|>,
```

Figure 4. Extracting all the data from the thematic block.

E. Using WDR during verification of functional completeness of temporal markers extraction tools

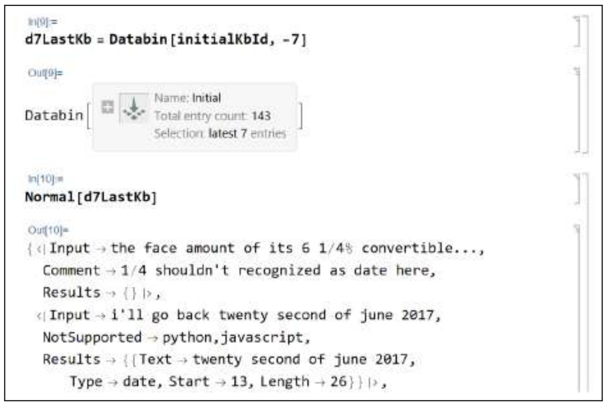
Basic steps to check the functional completeness of temporal markers extraction tool [5] with WDR integration are as follows:

1) Retrieve the test dataset from the thematic block.
2) Start the tool to be tested (e.g. mNevod, mMRT).
3) Read the obtained extraction results.
4) Compare with the expected results from the meta-information of the test dataset.



```
In[7]:=
d7FirstKb = Databin[initialKbId, {1, 7, 1}]
Out[7]=
Databin[ Name: Initial
         Total entry count: 143
         Selection: entries 1 to 7 step 1 ]
In[8]:=
Normal[d7FirstKb]
Out[8]=
{<|Input → i'll go back on 15,
  Results → {{Text → 15, Type → date, Start → 16, Length → 2}}|>,
 <|Input → i'll go back april 22, Results →
   {{Text → april 22, Type → date, Start → 13, Length → 8}}|>,
 <|Input → i'll go back jan-1, Results →
   {{Text → jan-1, Type → date, Start → 13, Length → 5}}|>,
```

Figure 5. Extracting elements 1 through 7 from the thematic block.

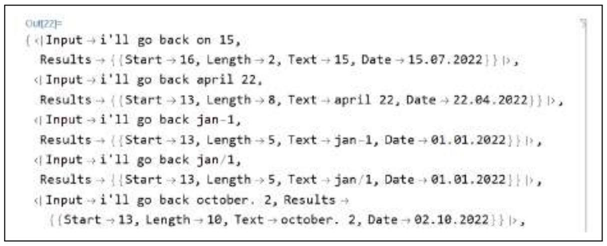


```
In[9]:=
d7LastKb = Databin[initialKbId, -7]
Out[9]=
Databin[ Name: Initial
         Total entry count: 143
         Selection: latest 7 entries ]
In[10]:=
Normal[d7LastKb]
Out[10]=
{<|Input → the face amount of its 6 1/4% convertible...,
  Comment → 1/4 shouldn't recognized as date here,
  Results → {}|>,
 <|Input → i'll go back twenty second of june 2017,
  NotSupported → python,javascript,
  Results → {{Text → twenty second of june 2017,
     Type → date, Start → 13, Length → 26}}|>,
```

Figure 6. Extracting the last 7 elements from the thematic block.

5) Upload the results into the WDR.

An example of mNevod results is shown in Fig. 7. The form of representation is the same as that of the mMRT module: for each *Input* string, the module lists extracted temporal markers in the *Results* list in text and numeric form.



```
Out[22]=
{<|Input → i'll go back on 15,
  Results → {{Start → 16, Length → 2, Text → 15, Date → 15.07.2022}}|>,
 <|Input → i'll go back april 22,
  Results → {{Start → 13, Length → 8, Text → april 22, Date → 22.04.2022}}|>,
 <|Input → i'll go back jan-1,
  Results → {{Start → 13, Length → 5, Text → jan-1, Date → 01.01.2022}}|>,
 <|Input → i'll go back jan/1,
  Results → {{Start → 13, Length → 5, Text → jan/1, Date → 01.01.2022}}|>,
 <|Input → i'll go back october. 2, Results →
   {{Start → 13, Length → 10, Text → october. 2, Date → 02.10.2022}}|>,
```

Figure 7. Temporal markers extraction results by mNevod module.

It should be noted that Nevod library, due to its structure, opens an additional possibility to use WDR. Nevod is a multipurpose library designed to search for pattern matches in text. Patterns are defined independently from the library in a special language of their description, they allow to flexibly configure the search and extraction of entities from the text [13]. Previously, to solve the problem of extracting temporal pointers from text, a standard date search set from Nevod's library of basic patterns was used. When testing the functional completeness, the disadvantages of this pattern set were revealed, it was supplemented and included as a component of the mNevod module. Taking into account independence of patterns from the library, it's expedient to place the received supplemented set of patterns in WDR. It will allow to make publicly available the actual version of the set, and at the same time will simplify the task of its subsequent correction.

III. Creating a thematic block for OSTIS conference materials analysis

The purpose of this part of the work is to demonstrate the simplest WM tools to create, maintain, use Wolfram Data Repository to form a centralized repository and integration with any other platforms and systems.

Tools of sampling and placement in WDR of programs of last five OSTIS conferences, examples of intellectual analysis, in particular, selection of reports of the indicated authors are explained. It is essential, that (for the purpose of demonstration of possibilities of integration of means of different programs, packages, systems) the information processing and analysis are carried out by means of Nevod library, and also by means of WM tools.

A. Preparing material for analysis

The examples below illustrate the preprocessing and placement in WDR of information extracted from the programs of five most recent OSTIS conferences.

There are four steps involved in preparing the materials:

1) Extract materials from the conference website, according to the list of years, programs of meetings (plenary, breakout sessions, as well as exhibition and poster demonstrations) and generate PDF files "ostis-2018", "ostis-2019", "ostis-2020", "ostis-2021", "ostis-2022" with subsequent placement in the folder with the current WM notebook.
2) Convert received PDF files into plain text format.
3) Remove introductory explanations and summaries.
4) Upload prepared materials in WDR.

Step 1. Materials retrieval can be performed either manually or automatically using the WM tools for working with WWW resources, for example, described in [3].

Step 2. For each prepared file, its contents are loaded and converted into a text format for further processing. The Import function is used to specify the data format to which the file contents will be converted when loaded. To apply the function to more than one file, the /@ function, a shortened version of the Map function *(apply the function to each item in the list)*, is used. The code for loading and converting files is the following:

$dataDir = NotebookDirectory[];$
$files = FileNames["*.pdf", dataDir <> "pdf/"];$
$contents = Import[\#, "Plaintext"]\&/@files;.$

Step 3. The introductory explanations (introductions from organizing committee to conference schedule) and summaries (abstracts) are deleted. During this procedure, it is advisable to save the cover pages of the programs to extract the year of the conference later. The processing is done using the function $StringSplit["string", patt, n]$ *(splits into substrings separated by delimiters matching the string expression patt, into at most n substrings)*. The implementation uses the additional option to search for a case insensitive pattern ($IgnoreCase- > True$). The code for material cleaning is as follows:

$getTitlePage = If[Length[\#] > 1, \#[[1]], ""]\&$
$[StringSplit[\#,$"организационный комитет"$, 2,$
$IgnoreCase- > True]]\&;$
$getMainBody = \#[[Length[\#]]]\&$
$[StringSplit[\#,$"график работы конференции"$, 2,$
$IgnoreCase- > True]]\&;$
$getClean = getTitlePage[\#] <> getMainBody[\#]\&;$
$preparedContents = Map[getClean, contents];.$

The final $getClean$ function is a composite of two other functions: $getTitlePage$ extracts the title page and $getMainBody$ extracts the main conference program text. The $getTitlePage$ function is designed taking into account the fact, that some input documents were represented by the conference web-pages saved in PDF format, in which this page is missing. The If function is used to check the presence of the title page.

Step 4. To upload data into the WDR, WM's capabilities are applied. The creation of the thematic block using the $CreateDatabin$ function is described above. At this stage peculiarities of work with text data, in particular with the Cyrillic alphabet were found out. In spite of the fact that each prepared document does not exceed the limit of 25KB per element, the final size of the loaded document increased several times and did not correspond to the specified limit. It turned out that the reason was the presence of Cyrillic letters in the materials. When converting the text to the Wolfram Language format, Cyrillic characters are represented in the wrong encoding, due to which the final size of the downloaded document increases many times. To get around this drawback, the resulting materials are compressed using WM $Compress$ function. Accordingly, when extracting materials for analysis, the restoration of the original data is performed using the paired function $Uncompress$. Thus, the code of WDR creation, uploading and extraction of materials is the following:

$initialDatabin = Databin[initialDatabinId];$
$DatabinUpload[initialDatabin, Compress/@$
$preparedContents];$
$downloadedContents = Uncompress/@Normal[$
$initialDatabin];.$

B. Examples of knowledge extraction, interpretation using the Nevod library

Preparing data for analysis by Nevod library — exporting to plain text files

The preprocessed documents of the conference programs extracted from the thematic block are saved to text files using the $Export$ function. File names are generated as integers in order using the $Range$ function. The path to the files is specified in the data folder in the subfolder "txt":

$plainTextFiles = FileNameJoin[\{dataDir <> "./txt", ToString[\#] <> ".txt"\}]\&/@Range[Length[downloadedContents]];.$

Then the $Export$ function is applied to each element of the list using the $MapIndexed$ function. Along with the contents of the element (parameter #1) it passes the sequence number of the element in the list (parameter #2 – list of one number element):

$MapIndexed[Export[plainTextFiles[[\#2[[1]]]], \#1]\&, downloadedContents];.$

Preparing to launch the Nevod library

To integrate with the Nevod library, the Nevod utility module was developed. Integration is performed through intermediate files. Mandatory parameters are passed to the input of the module in the following order:

1) path to the file with search patterns;
2) path to the file to perform search in;
3) path to the output file for writing results in JSON format.

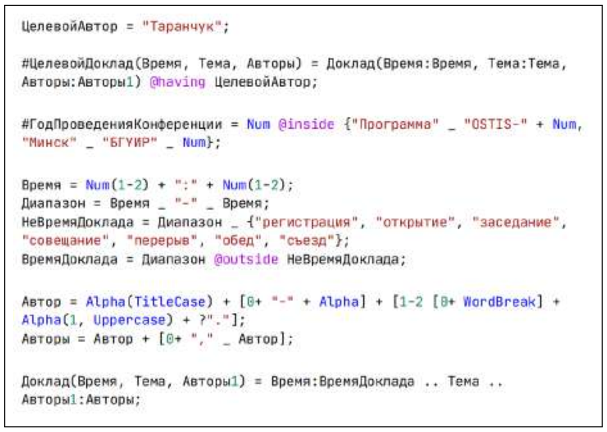
In the example, reports of the author whose name is specified in the template or query are selected and prepared for subsequent output. The files with search patterns for the two authors are in the data folder.

The following code generates the full paths to the files with the patterns to search the reports of the authors "Таранчук" *(patterns-taranchuk.np)* and "Головко" *(patterns-golovko.np)*:

$taranchukPatternsPath = dataDir <> "patterns-taranchuk.np";$
$golovkoPatternsPath = dataDir <> "patterns-golovko.np";.$

Fig. 8 shows the contents of the *patterns-taranchuk.np* file to search for the reports of the author "Таранчук". The main patterns whose matches are returned as results are "ЦелевойДоклад". (extracts the list of reports by the searched author with the report topic, full list of authors, and time) and "ГодПроведенияКонференции" (extracts the year of the conference). Other templates are internal and describe the constructs for extracting the target author ("ЦелевойАвтор"), time range ("Диапазон"), authors list ("Авторы"), and report list ("Доклад").

To start the Nevod module $RunProcess$ function is used, particularly, its variant, which allows to pass a list of startup arguments. The path to the module to be launched is saved in $nevodPath$. File names for saving Nevod module results are generated beforehand from input file names by replacing the extension from "txt" to "json" and placing them in a different folder:



```
ЦелевойАвтор = "Таранчук";

#ЦелевойДоклад(Время, Тема, Авторы) = Доклад(Время:Время, Тема:Тема,
Авторы:Авторы1) @having ЦелевойАвтор;

#ГодПроведенияКонференции = Num @inside {"Программа" _ "OSTIS-" + Num,
"Минск" _ "БГУИР" _ Num};

Время = Num(1-2) + ":" + Num(1-2);
Диапазон = Время _ "-" _ Время;
НеВремяДоклада = Диапазон _ {"регистрация", "открытие", "заседание",
"совещание", "перерыв", "обед", "съезд"};
ВремяДоклада = Диапазон @outside НеВремяДоклада;

Автор = Alpha(TitleCase) + [0+ "-" + Alpha] + [1-2 [0+ WordBreak] +
Alpha(1, Uppercase) + ?"."];
Авторы = Автор + [0+ "," _ Автор];

Доклад(Время, Тема, Авторы1) = Время:ВремяДоклада .. Тема ..
Авторы1:Авторы;
```

Figure 8. Nevod patterns to search for the target author's reports.

$fileBaseNames = FileBaseName/@FileNames[plainTextFiles];$
$resultTaranchukFileNames = FileNameJoin[\{dataDir <> "/json", \# <> "-taranchuk" <> ".json"\}]\&/@fileBaseNames;$
$resultGolovkoFileNames = FileNameJoin[\{dataDir <> "/json", \# <> "-golovko" <> ".json"\}]\&/@fileBaseNames;.$

For example, for the input text file *"1.txt"* the names of the output files with the search results will be (taking into account relative paths) *"./json/1-taranchuk.json"* and *"./json/1-golovko.json"* respectively.

The $MapIndexed$ function runs the Nevod module separately for each input file and places the results in the corresponding output file from the $resultFileNames$ list:

(* Таранчук *)
$MapIndexed[RunProcess[\{nevodPath, taranchukPatternsPath, \#1, resultTaranchukFileNames[[\#2[[1]]]]\}]\&, plainTextFiles];$

(* Головко *)
$MapIndexed[RunProcess[\{nevodPath, golovkoPatternsPath, \#1, resultGolovkoFileNames[[\#2[[1]]]]\}]\&, plainTextFiles]$

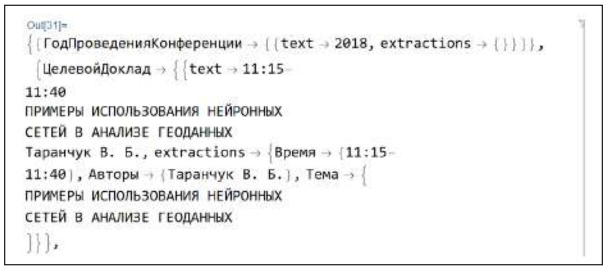
```
{<|ExitCode->0,StandardOutput->,StandardError-
>|>,<|ExitCode->0,StandardOutput->,StandardError-
>|>,<|ExitCode->0,StandardOutput->,StandardError-
>|>,<|ExitCode->0,StandardOutput->,StandardError-
>|>,<|ExitCode->0,StandardOutput->,StandardError->|>}
```

The presence of the value $ExtiCode-> 0$ in the results of the $RunProcess$ function indicates that the run of the module for each input file was successful.

Output the extraction results for the specified authors. The search results for each of the authors searched by Nevod are saved in JSON format. To read these files, $Import$ function with this format is used. The function is applied to each file, which allows to get a list of results.

```
taranchukResultsNV = Import[#,"JSON"]&/@
    resultTaranchukFileNames;
golovkoResultsNV = Import[#,"JSON"]&/@
    resultGolovkoFileNames;
```

The results obtained in JSON format have a specific structure, an example of which (with line feeds saved) is shown in Fig. 9.



```
Out[31]=
{{ГодПроведенияКонференции → {{text → 2018, extractions → {}}}},
 {ЦелевойДоклад → {{text → 11:15-
11:40
ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ НЕЙРОННЫХ
СЕТЕЙ В АНАЛИЗЕ ГЕОДАННЫХ
Таранчук В. Б., extractions → {Время → {11:15-
11:40}, Авторы → {Таранчук В. Б.}, Тема → {
ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ НЕЙРОННЫХ
СЕТЕЙ В АНАЛИЗЕ ГЕОДАННЫХ
}}},
```

Figure 9. Example of Nevod module results, initial structure.

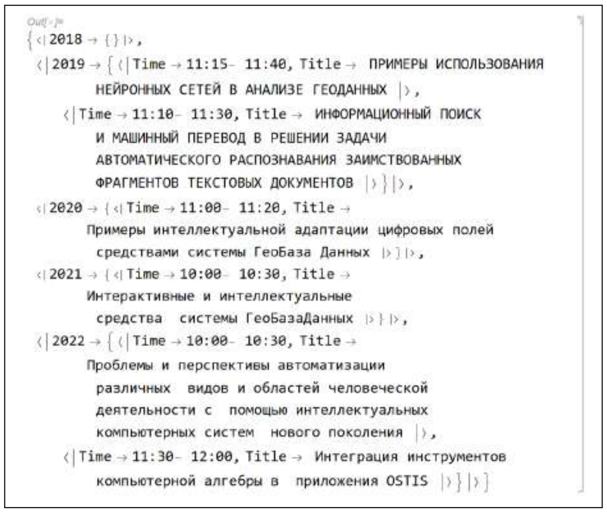
In subsequent processing with WM tools, the results are reduced to a flat representation, eliminating redundant line feeds. The functions for converting to such a representation are given below:

```
getYearNV = "text"/.
    ("ГодПроведенияКонференции"/.#)[[1]]&;
getReportTitleNV = StringReplace["\r\n"-> " "]
    [("Тема"/.("extractions"/.#))[[1]]]&;
getReportTimeNV = StringReplace["\r\n"-> " "]
    [("Время"/.("extractions"/.#))[[1]]]&;
getReportDetailsNV =< |
    "Time"-> getReportTimeNV[#],
    "Title"-> getReportTitleNV[#]| > &;
getReportListNV = getReportDetailsNV/@
    ("ЦелевойДоклад"/.#)&;
getTargetReportsNV = If[KeyExistsQ[#,
    "ЦелевойДоклад"], getReportListNV[#], {}]&;
getResultsNV =< |
    getYearNV[#]-> getTargetReportsNV[#]| >
&;.
```

The code for getting the final results is shown below:

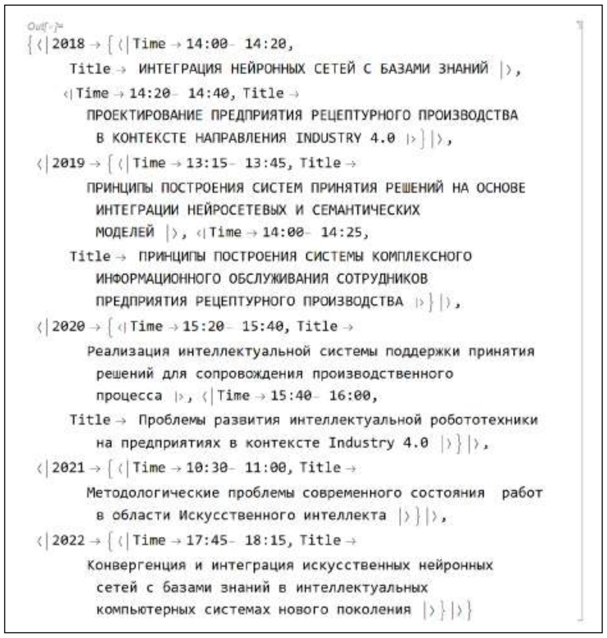
```
taranchukResults = getResultsNV/@
    taranchukResultsNV
golovkoResults = getResultsNV/@
    golovkoResultsNV.
```

The final results for each of the authors are shown in Fig. 10 and Fig. 11.



```
Out[-]=
{<|2018 → {}|>,
 <|2019 → {<|Time → 11:15- 11:40, Title → ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ
       НЕЙРОННЫХ СЕТЕЙ В АНАЛИЗЕ ГЕОДАННЫХ |>,
   <|Time → 11:10- 11:30, Title → ИНФОРМАЦИОННЫЙ ПОИСК
       И МАШИННЫЙ ПЕРЕВОД В РЕШЕНИИ ЗАДАЧИ
       АВТОМАТИЧЕСКОГО РАСПОЗНАВАНИЯ ЗАИМСТВОВАННЫХ
       ФРАГМЕНТОВ ТЕКСТОВЫХ ДОКУМЕНТОВ |>}|>,
 <|2020 → {<|Time → 11:00- 11:20, Title →
      Примеры интеллектуальной адаптации цифровых полей
       средствами системы ГеоБаза Данных |>}|>,
 <|2021 → {<|Time → 10:00- 10:30, Title →
      Интерактивные и интеллектуальные
       средства  системы ГеоБазаДанных |>}|>,
 <|2022 → {<|Time → 10:00- 10:30, Title →
      Проблемы и перспективы автоматизации
       различных  видов и областей человеческой
       деятельности с  помощью интеллектуальных
       компьютерных систем  нового поколения |>,
   <|Time → 11:30- 12:00, Title → Интеграция инструментов
       компьютерной алгебры в  приложения OSTIS |>}|>}
```

Figure 10. Flat results for the author Таранчук.



```
Out[-]=
{<|2018 → {<|Time → 14:00- 14:20,
     Title → ИНТЕГРАЦИЯ НЕЙРОННЫХ СЕТЕЙ С БАЗАМИ ЗНАНИЙ |>,
    <|Time → 14:20- 14:40, Title →
      ПРОЕКТИРОВАНИЕ ПРЕДПРИЯТИЯ РЕЦЕПТУРНОГО ПРОИЗВОДСТВА
       В КОНТЕКСТЕ НАПРАВЛЕНИЯ INDUSTRY 4.0 |>}|>,
 <|2019 → {<|Time → 13:15- 13:45, Title →
      ПРИНЦИПЫ ПОСТРОЕНИЯ СИСТЕМ ПРИНЯТИЯ РЕШЕНИЙ НА ОСНОВЕ
       ИНТЕГРАЦИИ НЕЙРОСЕТЕВЫХ И СЕМАНТИЧЕСКИХ
       МОДЕЛЕЙ |>, <|Time → 14:00- 14:25,
     Title → ПРИНЦИПЫ ПОСТРОЕНИЯ СИСТЕМЫ КОМПЛЕКСНОГО
       ИНФОРМАЦИОННОГО ОБСЛУЖИВАНИЯ СОТРУДНИКОВ
       ПРЕДПРИЯТИЯ РЕЦЕПТУРНОГО ПРОИЗВОДСТВА |>}|>,
 <|2020 → {<|Time → 15:20- 15:40, Title →
      Реализация интеллектуальной системы поддержки принятия
       решений для сопровождения производственного
       процесса |>, <|Time → 15:40- 16:00,
     Title → Проблемы развития интеллектуальной робототехники
       на предприятиях в контексте Industry 4.0 |>}|>,
 <|2021 → {<|Time → 10:30- 11:00, Title →
      Методологические проблемы современного состояния  работ
       в области Искусственного интеллекта |>}|>,
 <|2022 → {<|Time → 17:45- 18:15, Title →
      Конвергенция и интеграция искусственных нейронных
       сетей с базами знаний в интеллектуальных
       компьютерных системах нового поколения |>}|>}
```

Figure 11. Flat results for the author Головко.

The results show that author Таранчук had no reports in 2018, two reports in 2019, one each in 2020 and 2021, and two in 2022; author Головко published in each of the five years listed.

C. Examples of knowledge extraction and interpretation using Wolfram Mathematica tools

Extracting and processing information from thematic blocks is possible with Mathematica tools [14], some of which were listed above. It should be added that any kernel and application package functions and proprietary program modules can be used. The above examples can be repeated with string templates, list manipulations, rearranging text phrases, and layouts. Below are a few examples and the codes for getting the results with these WM tools.

Extracting the year of the conference. To extract the conference year, $StringCases$ function is used to extract patterns from strings. $"WhitespaceCharacter"$ (including string translation), $"DigitCharacter"$, $"\sim\sim"$ – strict following, $"x..."$ – repeat one or more times, $"a|b"$ – alternative choice between a and b. Additional work with nested lists: $Flatten$ – flatten nested lists, $Part$ – get a part of the list. Parameter $IgnoreCase$ allows you to match with the template without taking into account the upper or lower case of the string. Extracting the year from the title page of the program or in the first footer:

```
getYearWM = Part[#, 1]&@ ∗ Flatten@∗
    StringCases[DigitCharacter..]@ ∗ StringCases[
    ("Программа"~~ WhitespaceCharacter.. ~~
    ”OSTIS − ” ~~ DigitCharacter..)
    |("Минск"~~ WhitespaceCharacter.. ~~
    "БГУИР"~~ WhitespaceCharacter.. ~~
    DigitCharacter..), IgnoreCase− > True];
```

Extracting one participant of the conference.

The ”*LetterCharacter*” pattern describes a letter character. The first pattern below checks that the last name starts with a capital letter using the *UpperCaseQ* function. *Longest* pattern allows to select the longest match. It should be noted that in this particular case, the order of the alternatives is important. First is the longer one, second is the shorter one, since the second one is a superset of the first one; if written in reverse order, it returns the shortest match.

```
upperLetter = LetterCharacter?UpperCaseQ;
authorPattern = upperLetter ~~
    LetterCharacter.. ~~ WhitespaceCharacter.. ~~
    Longest[(upperLetter ~~ ”.” ~~
    WhitespaceCharacter... ~~ upperLetter ~~ ”.”)
    |(upperLetter ~~ ”.”)];
```

Extracting several participants. Template ”...” is used to repeat a particular part zero or more times.

```
multipleAuthorsPattern = Longest[
    authorPattern ~~ WhitespaceCharacter... ~~
    (”,” ~~ WhitespaceCharacter.. ~~
    authorPattern)...];
```

Extracting a time range.

```
timePattern = DigitCharacter ~~
    DigitCharacter ~~ ” : ” ~~ DigitCharacter ~~
    DigitCharacter;
timeIntervalPattern = timePattern ~~ ” − ” ~~
    (WhitespaceCharacter...) ~~ timePattern;
```

Extracting elements of the program schedule, except reports.

```
getNotReports = StringCases[
    timeIntervalPattern.. ~~ WhitespaceCharacter..
    ~~"регистрация"|"открытие"|"заседание"|
    "совещание"|"перерыв"|"обед"|"съезд",
    IgnoreCase− > True];
```

Removing all the elements of the program schedule, except reports.

```
getTrimmedText = StringReplace[#,
    getNotReports[#]− > ””]&;
```

Extracting a list of all reports.

```
getReports = StringCases[Shortest[
    timeIntervalPattern ~~ __ ~~
    multipleAuthorsPattern]]@ ∗ getTrimmedText;
```

Extracting reports with the specified author. This extraction is represented by a configurable function. The input is a string parameter, specifying the author's surname. The result is a new function that filters the list of reports in search of reports of the specified author.

```
getReportsWithAuthor = Function[{author},
    Select[#, StringContainsQ[StringCases[#,
    multipleAuthorsPattern][[1]], author]&]&];
```

Extracting report details: time and title.

```
getReportTime = StringCases[{
    timeIntervalPattern, ”\n”− > ” ”}];
getReportTitle = StringReplace[{
    (timeIntervalPattern|
    multipleAuthorsPattern)− > ””, ”\n”− > ” ”}];
getReportDetailsWm =< |
    ”Time”− > getReportTime[#][[1]],
    ”Title”− > getReportTitle[#]| > &;
```

Final function applied to the input file.

```
getResultsWM = Function[{author},
    < |getYear[#]− > getReportDetailsWm/@
    getReportsWithAuthor[author][getReports[#]]
    | > &];
```

Extraction results for the specified author.
Extraction result for the author Таранчук is shown in Fig. 12, obtained with the following code:

```
taranchukResults2 = getResultsWM["Таранчук"]/@
    downloadedContents.
```

```
{<|2018->{ }|>, <|2019->{ <|Time->11:15- 11:40,Title-> ПРИМЕРЫ ИС-
ПОЛЬЗОВАНИЯ НЕЙРОННЫХ СЕТЕЙ В АНАЛИЗЕ ГЕОДАННЫХ
|>, <|Time->11:10- 11:30,Title-> ИНФОРМАЦИОННЫЙ ПОИСК И МА-
ШИННЫЙ ПЕРЕВОД В РЕШЕНИИ ЗАДАЧИ АВТОМАТИЧЕСКОГО
РАСПОЗНАВАНИЯ ЗАИМСТВОВАННЫХ ФРАГМЕНТОВ ТЕКСТО-
ВЫХ ДОКУМЕНТОВ |>}|>, <|2020->{ <|Time->11:00- 11:20,Title->
Примеры интеллектуальной адаптации цифровых полей средствами
системы ГеоБаза Данных |>}|>, <|2021->{ <|Time->10:00- 10:30,Title->
Интерактивные и интеллектуальные средства системы ГеоБазаДанных
|>}|>, <|2022->{ <|Time->10:00- 10:30,Title-> Проблемы и перспективы
автоматизации различных видов и областей человеческой деятельности
с помощью интеллектуальных компьютерных систем нового поколения
|>, <|Time->11:30- 12:00,Title-> Интеграция инструментов компьютер-
ной алгебры в приложения OSTIS |>}|>}
```

Figure 12. Extraction results for the author Таранчук.

IV. Conclusion

As an analogue of the integration of knowledge bases of the corporate OSTIS-system into the OSTIS Ecosystem, the example of integration of the local intelligent computer system based on Nevod library with the knowledge base of Wolfram Mathematica computer algebra system (Wolfram Data Repository) is presented.

The possibilities of working with Wolfram Data Repository are described by the example of creating a thematic block of temporal markers analysis. In particular, functions for creating a thematic block, data upload and extraction from Wolfram Data Repository with the given sampling parameters are shown. The methodology for checking the functional completeness of temporal markers extraction tools from text, with focus on recognition rather than unambiguous identification, is described. This methodology is supplemented by the possibility to publish a thematic block in the public domain in order to maintain the current state of the test dataset.

The process of analyzing OSTIS conference materials for finding the reports written by a specific author is described in detail. Preparation of materials and their uploading in Wolfram Data Repository carried out by means of Wolfram Mathematica. For the purpose of demonstration of possibilities of integration with other platforms, systems, extraction and interpretation of knowledge are made independently by means of Nevod library, and by means of Wolfram Mathematica. Usage of Wolfram Mathematica tools for text manipulation and pattern search is described in detail. A step-by-step explanation of a final extraction pattern construction, as well as a brief description of main functions used to build it, is provided. The knowledge extraction results, produced by Nevod and by Wolfram Mathematica tools, are shown and uploaded to Wolfram Data Repository.

References

[1] V. V. Golenkov, N. A. Guliakina, D. V. Shunkevich Otkrytaja tehnologija ontologicheskogo proektirovanija, proizvodstva i jekspluatacii semanticheski sovmestimyh gibridnyh intellektual'nyh komp'juternyh sistem [Open technology of ontological design, production and operation of semantically compatible hybrid intelligent computer systems], Bestprint, 2021, P. 690

[2] Wolfram Mathematica: Modern Technical Computing. Available at: https://www.wolfram.com/mathematica (accessed 2023, Feb).

[3] V. B. Taranchuk Integration of computer algebra tools into OSTIS applications. *Open semantic technologies for intelligent systems*, 2022, no 6, pp. 369-374.

[4] Nevod is a language and technology for pattern-based text search. Available at: https://github.com/nezaboodka/nevod (accessed 2022, Apr).

[5] V. A. Savenok, V. B. Taranchuk Vozmozhnosti i sredstva biblioteki Nevod pri reshenii zadach izvlecheniya vremennykh ukazatelei v tekste [Features and tools of the Nevod library in solving problems of extracting temporal markers in the text]. *Problemy fiziki, matematiki i tekhniki [Problems of Physics, Mathematics and Technics]*, 2022, no 4, pp. 84-92, https://doi.org/10.54341/20778708_2022_4_53_84.

[6] E. A. Suleimanova Semanticheskii analiz kontekstnykh dat [Semantic analysis of contextual dates]. *Programmnye sistemy: teoriya i prilozheniya [Program systems: theory and applications]*, 2015, vol. 6, no 4, pp. 367-399.

[7] Microsoft.Recognizers.Text provides recognition and resolution of numbers, units, and date/time expressed in multiple languages. Available at: https://github.com/microsoft/Recognizers-Text (accessed 2022, Apr).

[8] Intelligent Virtual Agents and Bots | Microsoft Power Virtual Agents. Available at: https://powervirtualagents.microsoft.com/en-us (accessed 2022, Apr).

[9] Recognizers Test Cases Specs for Date Extractor. Available at: https://github.com/microsoft/Recognizers-Text/blob/master/Specs/DateTime/English/DateExtractor.json (accessed 2022, Apr).

[10] D. A. Surkov, I. V. Shimko, V. A. Savenok et al. Sposob poiska v tekste sovpadenii s shablonami : pat. 037156, Belarus, MPK G06F 17/27, G06F 17/24, Evraziiskaya patentnaya organizatsiya, 2021, byul. no 2.

[11] Data Semantics | Wolfram Datadrop Quick Reference. Available at: https://www.wolfram.com/datadrop/quick-reference/data-semantics (accessed 2023, Feb).

[12] Databin — Wolfram Language Documentation. Available at: https://reference.wolfram.com/language/ref/Databin.html (accessed 2023, Feb).

[13] Nevod Basic Patterns. Available at: https://github.com/nezaboodka/nevod-patterns (accessed 2023, Feb).

[14] Working with String Patterns — Wolfram Language Documentation. Available at: https://reference.wolfram.com/language/tutorial/WorkingWithStringPatt\erns.html (accessed 2023, Feb).

Инструменты создания и сопровождения базы знаний путем интеграции системы Wolfram Mathematica и пакета Nevod

Таранчук В. Б., Савёнок В. А.

Одним из итогов текущего рассмотрения состояния технологий проектирования и анализа баз знаний, программных и аппаратных платформ реализации семантически совместимых интеллектуальных компьютерных систем является заключение о необходимости формулировки принципов коллективного проектирования, разработки, верификации баз знаний. Соответственно, важно не только сформулировать и обосновать теорию, формализовать требования, но и разработать инструменты представления такой формальной теории в виде (формате) базы знаний соответствующего портала научных знаний. Именно такая концепция является целью реализации и развития Экосистемы OSTIS, которая, в частности, предназначена для решения задач сближения и слияния функциональных свойств систем различных классов; расширения спектра, организационно-технического объединения, осуществления координации программных, вычислительных и телекоммуникационных средств; унификации интеллектуальных компьютерных систем. Особое место в таком объединении следует отвести решению вопросов интеграции средств Экосистемы OSTIS с системами компьютерной математики, особенно с системами компьютерной алгебры.

В данной работе приведен пример интеграции локальной интеллектуальной компьютерной системы на основе библиотеки Nevod с базой знаний системы компьютерной алгебры Wolfram Mathematica, что можно интерпретировать как аналог действий по локализации баз знаний корпоративной ostis-системы в состав Экосистемы OSTIS. Представлены и поясняются примеры использования инструментов анализа локальной базы знаний, ее перевода из статуса виртуальный в статус реальной.

Received 25.03.2023