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 book 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 10 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[#,"организованный 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 treason 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[
  initial Databin];.
```

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.

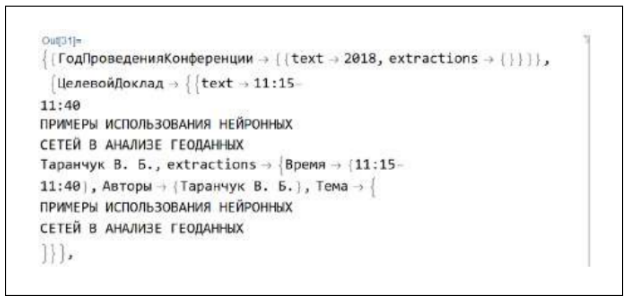

Figure 9. Example of Nevod module results, initial structure.

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.

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

Figure 10. Flat results for the author Таранчук.

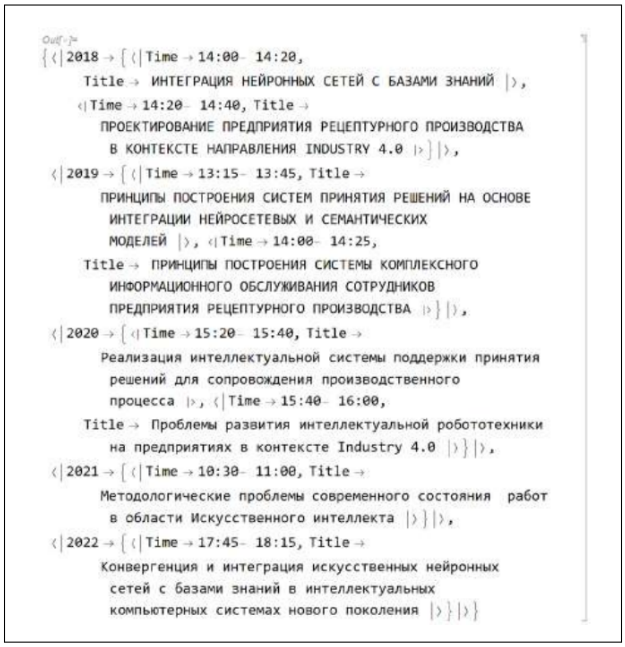

Figure 11. Flat results for the author Головко.

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*", " ∼∼ " – 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