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 contaned 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 metainformation, 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 pseudo-random 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

Figure 1: Detailed information about a thematic block.

Figure 2: 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,* {*entry1, entry2, ...*}] *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.

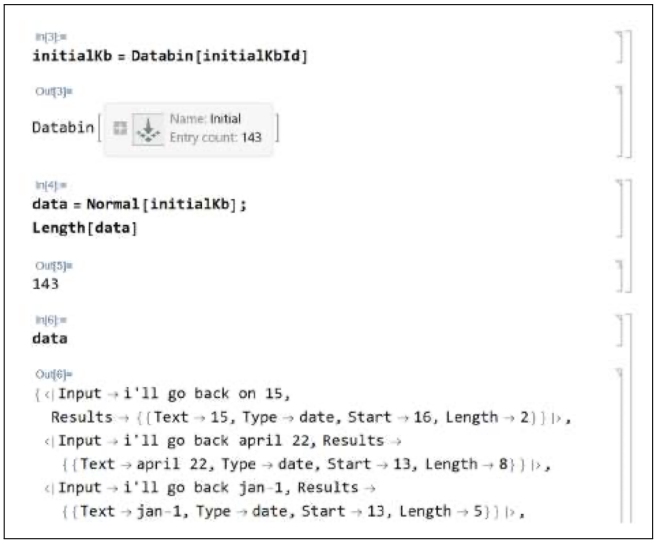

Figure 3: Extracting all the data from the thematic block.

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 metainformation of the test dataset.

Figure 4: Extracting elements 1 through 7 from the thematic block.

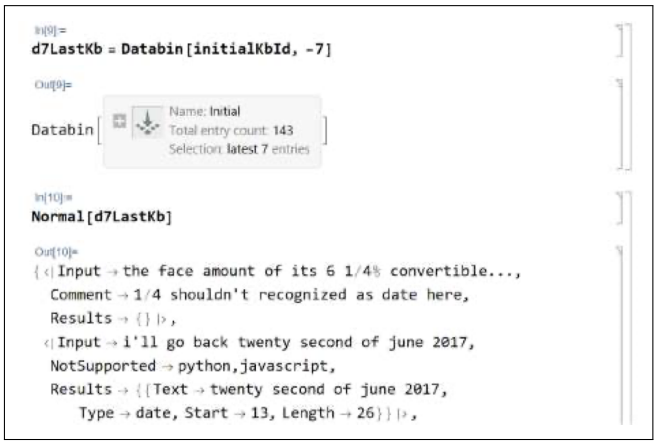

Figure 5: Extracting the last 7 elements from the thematic block.

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.

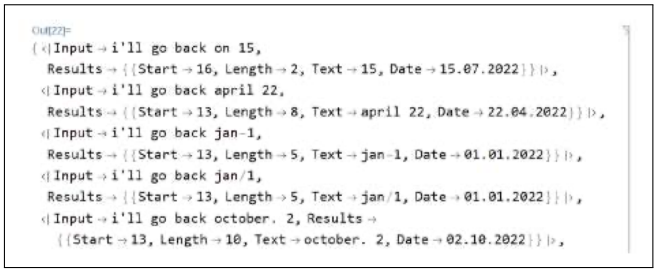

Figure 6: Temporal markers extraction results by mNevod module.

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