Chapter 9
Algorithm of Building a Matrix Semantic-Syntactic Representation of a Natural Language Text

Abstract This chapter sets forth an original algorithm of constructing a matrix semantic-syntactic representation of a natural language text. This algorithm, called *BuildMatr*1, is multilingual: the input texts (the statements, commands, and questions) may belong to the sublanguages of English, German, and Russian languages (a Latin transcription of Russian texts is considered). A pure syntactic representation of an analyzed text isn't used: the proposed algorithm is oriented at directly finding the conceptual relations between the meanings of the fragments of an NL-text.

9.1 Task Statement

9.1.1 Purpose of Development and Non-detailed Structure of the Algorithm

During the last decade, the Internet has become a multilingual system. That is why one has been able to observe a permanently growing interest in multilingual computer programs for processing NL-texts. It seems that the complete potential of semantics-oriented approaches to developing multilingual algorithms of processing NL-texts is far from being realized.

The purpose of this chapter and Chap. 10 is to make a new step (after the author's monograph [85]) in the realization of this potential. With this aim, a new algorithm of semantic-syntactic analysis of the NL-texts is developed. The input NL-texts may belong to the sublanguages of English, German, and Russian. The proposed algorithm implements a new method of explicating the semantic-syntactic structure of NL-texts being a part of a new method of fulfilling the transformation "Natural Language (NL) text – Semantic Representation (SR) of the text" described in Chap. 8.

The formal notion of a linguistic basis introduced in Chap. 7 is interpreted here as a description of the structure of linguistic database (LDB) used by the algorithm.

V.A. Fomichov, *Semantics-Oriented Natural Language Processing*, IFSR International Series on Systems Science and Engineering 27, DOI 10.1007/978-0-387-72926-8_9,

The considered texts may express the statements, commands, specific questions (i.e. the questions with interrogative words) of several kinds, general questions (i.e. the questions with the answers "Yes" / "No").

To realize the new method of fulfilling the transformation "NL-text ⇒ SR of the text" suggested in Chap. 8, the following task is stated: to develop an algorithm *SemSynt*1 being the composition of the algorithms *BuildMatr*1 and *BuildSem*1 meeting the following requirements:

1. *BuildMatr*1 is an algorithm of transforming the texts from some sublanguages of English, German, and Russian languages being of practical significance to their matrix semantic-syntactic representations (MSSR);
2. *BuildSem*1 is an algorithm constructing a semantic representation of the NL-text from its MSSR. Moreover, this SR of the text is an expression of a certain SK-language (i.e. it is a K-representation of the input text).

One of the input data of the algorithm *BuildMatr*1 is the string variable *lang* with the values *English*, *German*, *Russian*.

The main output data of the algorithm are the string *kindtext* defining the form of the input text (i.e. classifying this text) and the string-numerical matrix *Matr* – an MSSR of the input text.

An important peculiarity of the algorithm *BuildMatr*1 is that it directly discovers the semantic relationships between the meanings of the text's fragments without fulfilling the traditional syntactic analysis of the input text.

The starting point for developing the algorithm *BuildMatr*1 was the analysis of the surface and semantic structure of the texts from the following sublanguages of English, German, and Russian languages being of practical interest:

- the questions and statements in natural language regarding the scientific publications and specialists' participation in scientific conferences (it is supposed that these questions and statements are addressed to an online searching system of a new generation);
- the commands and questions to a transport-loading intelligent robot, in particular, to a robot operating in an automated warehouse and to a robot acting in an airport;
- the questions and statements concerning the manufacture, export, and import of certain products (it is supposed that these questions and statements are addressed to an intelligent database);
- the questions of an automated warehouse operator addressed to an intelligent database;
- the questions of potential clients to an intelligent database of an online shop.

9.1.2 Some Peculiarities of the Approach

It is known that the problem of computer semantic-syntactic analysis (SSA) of NL-texts includes many aspects being not equally worked out. The initial stage of semantic-syntactic analysis of the NL-texts includes finding the basic forms of the

words of the input text, finding the possible values of morphological characteristics (number, case, person, tense, etc.) for these words and splitting the text into the segments corresponding to certain elementary semantic units. These segments include, for example, the following expressions: "were shipped," "will be prepared," "Olympic games," "840 km," "1999th year," "2 h," "five percents."

The questions of designing the morphological analyzers of NL-texts are investigated in many publications.

The problems concerning the automatic selection of the short text segments designating the elementary units of semantic level are not very complicated from the logical standpoint. These problems can be solved directly at the level of programming an algorithm.

A literature analysis and own experience of the author show that the main logical difficulties concerning the automation of SSA of NL-texts are related to the search of semantic relationships between the components of the input text.

That is why the algorithm *BuildMatr*1 is focused on formalization and algorithmization of the process of searching the semantic relationships between the components of the input text. This focus is achieved as follows:

1. In many cases the input text T of the algorithm is a certain abstraction as compared with the real input text of a linguistic processor. That is because the units of the input text of the algorithm could be represented not only by the words but also by the short phrases designating the elementary semantic, or conceptual, units ("were shipped," "will be prepared," "Olympic games," "840 km," "1999th year," "2 h").
2. It is assumed that there is a mapping associating the words and their basic forms with the sets of values of morphological characteristics. A mapping of the kind also associates each construct (a numerical value of a certain parameter) with a certain semantic, or conceptual, unit. The concept of a text-forming system was introduced in Chap. 7 just with this aim.

There are two main approaches to the algorithm development: top-to-bottom (descending) and bottom-to-top (ascending) design. It seemed to be expedient to combine both these methods during the development of a strongly structured algorithm *Semsynt*1. This combination of descending and ascending methods of algorithm design is reflected in the description of the algorithm *BuildMatr*1 in the next sections of this book and in the description of the algorithm *BuildSem*1 in Chap. 10.

In cases when an auxiliary algorithm is very simple (can be directly programmed) or its development does not represent any theoretical difficulties considering existing scientific publications, only an external specification of such algorithm is included in the structured descriptions of the algorithms *BuildMatr*1 and *BuildSem*1, i.e., a description of the purpose of development, input and output data of an algorithm.

One of possible variants of a language for algorithm development (or pseudocode, see [140]) is used for describing the algorithms in this and next chapters. The following service words are used in the algorithms: $begin$, end, if, $then$, $else$, $end-if$, $loop$, $loop-until$, $end-of-loop$, $case-of$, $end-case-of$.

9.2 Initial Stages of Developing the Algorithm BuildMatr1

This section contains an external specification and a plan of the algorithm *Build Matr*1 to be developed in the sections of this chapter.

9.2.1 External Specification of the Algorithm BuildMatr1

Input:
Lingb – linguistic basis (see Sect. 7.7);
T – input text;
lang – string variable with the values *English*, *German*, *Russian*.
Output:
nt – integer – the quantity of elementary meaningful text units;
Rc – classifying representation of the input text T (see Sect. 8.1);
Rm – morphological representation of the input text (see Sect. 8.1);
kindtext – string variable; its value allows to attribute the input text to one of the text subclasses;
Arls – two-dimensional array – projection of the lexico-semantic dictionary *Lsdic* on the input text T;
Arvfr – two-dimensional array – projection of the dictionary of verbal-prepositional frames Vfr on the input text T;
Arfrp – two-dimensional array – projection of the dictionary of prepositional semantic-syntactic frames *Frp* on the input text T (see Sect. 8.2);
Matr – matrix semantic-syntactic representation (MSSR) of the input text (see Sect. 8.3).

9.2.2 Development of a Plan of the Algorithm BuildMatr1

Plan of the Algorithm BuildMatr1

Begin

$Building-compon-morphol-representation\,(T, Rc, nt, Rm)$
$Building-projection-of-lexico-semantic-dictionary$
$(Lsdic, nt, Rc, Rm, Arls)$
$Building-projection-of-verbal-frames-dictionary$
$(Arls, Vfr, nt, Rc, Rm, Arvfr)$
$Building-projection-of-prepos-frames-dictionary$
$(Arls, Frp, nt, Rc, Rm, Arfrp)$
$Forming-initial-values-of-data$
$Defining-form-of-text$
$(nt, Rc, Rm, leftprep, mainpos, kindtext, pos)$
$loop-until$

$pos := pos + 1$
$Class := Rc\,[pos, tclass]$
$case\ class\ of$
$preposition:\ Processing - preposition$
$adjective:\ Processing - adjective$
$cardinal - number:\ Processing - cardinal - number$
$noun:\ Processing - noun$
$verb:\ Processing - verbal - form$
$conjunction:\ Empty - operator$
$construct:\ Processing - construct$
$name:\ Processing - name$
$marker:\ Empty\ operator$
$end - case - of$
$exit - when\ (pos = nt)$
end

$Algorithm$ "$Forming - initial - values - of - data$"

Begin
Set to null all integer variables.
Assign the value *nil* (empty element) to all string variables.
Set to null all numeric columns and fill in with the string *nil* all string positions of the used one-dimensional and two-dimensional arrays.
For each row *k* of the classifying representation *Rc* corresponding to the word of the input text carry out the following actions:
$Matr\,[k, locunit]$:= the minimal number of the row of array *Arls* (projection of the lexico-semantic dictionary *Lsdic* on the considered text) containing the information corresponding to the considered word;
$Matr\,[k, nval]$:= quantity of rows of *Arls* corresponding to this word
end
The next sections of this chapter are dedicated to the detalization of this plan, i.e. to the development of the first part of the algorithm of semantic-syntactic analysis of the texts from sublanguages of English, German, and Russian being of practical interest.

9.3 Description of the Algorithm Classifying the Input Texts

9.3.1 The Purpose of the Algorithm

The algorithm "Defining-form-of-text" is designed for attributing the text to a certain class. The type of the class is a value of the output variable *kindtext*. The range of possible values of the variable *kindtext* can be described as follows:

- Statements – the value *stat* – Example (English): "The joint-stock company 'Sail' has been exporting the goods to Bulgaria since 1999."
- Commands – the value *imp* – Example (English): "Ship the containers to the factory 'Dawn' by 16:30."
- Closed (general) questions – the value *genqs* – Example (English): "Does the joint-stock company 'Sail' export the goods to Bulgaria?"
- Questions about the quantity of objects – the value $specqs-quant1$ – Example (English): "How many articles on organic chemistry by Professor Igor Somov were published last year?"
- Questions about the quantity of events – the value $specqs-quant2$ – Example (English): "How many times this year was a textbook by Korobov requested?"
- Role questions – the value $specqs-role$ - Example (English): "Where three-ton containers came from?"
- Questions with an interrogative word attached to a noun in singular, where the sought-for value is an individual – the value $specqs-relat1$ – Example (English): "What institute does Professor Igor Somov work at?"
- Questions with an interrogative word attached to a noun in plural, where the sought-for value is a set of individuals – the value $specqs-relat2$ – Example (English): "What countries does the joint-stock company 'Rainbow' import the component parts from?"

The value of the variable *kindtext* is used in the algorithm building the semantic representation (SR) of the text from its matrix semantic-syntactic representation. For example, let Qs1 = "Where three-ton containers came from?" Qs2 = "How many times this year was the textbook by Korobov requested?"

For the question Qs1, $kindtext = specqs-role$ and SR of the question Qs1 may be the following K-formula:

$$Question\,(x1,\,(Situation\,(e1,\,delivery2 * (Place1,\,x1)$$

$$(Time,\,x2)\,(Object1,\,certn\,set * (Compos,\,container *$$

$$(Weight,\,3/ton)) : S1)) \wedge Earlier\,(x2,\,\#now\#))).$$

In case of question Q2 $kindtext = specqs-quant2$ and SR of the question Q2 may be the following K-formula:

$$Question\,(x1,\,(x1 \equiv Quant-elem\,(all\,inquiry1 *$$

$$(Time, current-year)\,(Subject-of-inquiry,\,certn\,textbook *$$

$$(Author,\,certn\,person * (Surname,\,"Korobov'') : x2))))).$$

Besides the variable *kindtext*, the output data of the algorithm "Defining-form-of-text" also include the variables *mainpos* and *pos*. The value of the variable *mainpos* is the position number of an interrogative word in the text.

For example, for the question Qs3 = "From which countries the joint-stock company 'Rainbow' imports the component parts?" and Qs1 = "Where three-ton

containers came from?" the variable *mainpos* takes the values 2 and 1 correspondingly. For the commands, statements, and general questions the variable *mainpos* takes the value 1.

9.3.2 Compound Designations of the Subclasses of Lexical Units

The proposed algorithm of building a matrix semantic-syntactic representation (MSSR) of an input NL-text is a modification of the algorithm *BuildMatr* described in Russian in the monograph [85], taking into account some peculiarities of the Russian language.

In order to develop a multilingual algorithm of constructing an MSSR of an input NL-text from the sublanguages of English, German, and Russian, it turned out to be reasonable to introduce the compound designations of the subclasses of lexical units.

The principal impulse was given by the following observation. The interrogative word "what" in English can be used at least in two different contexts illustrated by the questions Qs1 = "What has John received" and Qs2 = "What books are being edited by John?".

Let's agree to say that in the question Qs1, "what" is a role interrogative word, and in the question Qs2, "what" is a noun-attaching interrogative word.

That is why it is proposed to consider the following interlingual subclasses of interrogative words:

- *roleqswd* ("who," "whom" in English; "wer", "wem", "woher" in German; "kto", "komu," etc., in Russian);
- *noun_attach_qswd* ("welche" in German; "kakoy", "kakie" in Russian);
- *roleqswd* | *noun_attach_qswd* ("what" in English);
- *qswd_quant*1 | *qswd_quant*2 ("how many" in English; "wieviel" in German; "skol'ko" in Russian), where the component *qswd_quant*1 is realized in the questions about the quantity of events ("How many times" in English; "Wieviel Mal" in German; "Skol'ko raz" in Russian), and the component *qswd_quant*2 is realized in the questions about the quantity of objects.

Let *Lexsubclasses* be the set of strings denoting all subclasses of considered lexical units. Then let the mapping *Setlextypes* be defined as follows:

- for every simple (i.e. containing no symbol |) d from *Lexsubclasses*,

$$Setlextypes(d) = \{d\};$$

- for every compound string h of the form

$$d_1 \mid d_2 \mid \ldots \mid d_n$$

from *Lexsubclasses*,

$$Setlextypes(h) = \{d_1, d_2, \ldots d_n\}.$$

Example. $Setlextypes(roleqswd \mid noun\_attach\_qswd) = \{roleqswd, noun\_attach\_qswd\}$.

9.3.3 External Specification of the Algorithm "Defining-form-of-text"

Input:

Lingb – linguistic basis,

Rc and *Rm* – classifying and morphological representations of the input text *T* (see Sect. 8.1);

lang – string variable with the values *English*, *German*, *Russian*.

Output:

kindtext – string variable designating the form of the input text;

$leftprep$ – string variable designating the preposition in the beginning of the text;

mainpos – integer variable designating the rightest position of an interrogative word;

pos – integer variable designating such position in the input text that it is required to continue processing of the text after this position.

External specification of auxiliary algorithms

Specification of the algorithm – function "*Number*"

Input:

d – element of the morphological space $Spmorph(Tform(Lingb))$ corresponding to the word which may have a singular or a plural number.

Output:

*number*1 – value 1 for singular number, 2 for plural number, 3 in the cases when the word could be attributed to both singular and plural.

Specification of the algorithm "*Form – of – verb*"

Input:

d – element of the morphological space $Spmorph(Tform(Lingb))$ corresponding to the verb.

Output:

$form1$ – string with the value "infinitive" if *d* corresponds to the verb in infinitive form; the value "indicative" for representing the verb in indicative mood; the value "imperative" if *d* corresponds to the verb in imperative mood.

Specification of the algorithm

"*Right_noun*(*pos*1, *posnoun*1)"

Input:

Rc and *Rm* – classifying and morphological representations of the input text *T* (see Sect. 8.1);

*pos*1 – integer – position of a lexical unit in the input text, i.e. position in the classifying representation *Rc*

Output:

*posnoun*1 – integer – either 0 or the position of a noun from the input text to the right from *posnoun*1 having the minimal distance from *pos*1.

AuxiliaryAlgorithm

Right_noun(*pos*1, *posnoun*1)

Comment (condition of calling the algorithm):

$$Setlextypes(Rc[pos1, subclass]) \ni noun\_attach\_qswd$$

begin posnoun1 := 0; $k := pos1$
$loop\_until$
$k := k + 1$
$part1 := Rc[k, tclass]$)
$if\ part1 = noun\ then\ posnoun1 := k\ end - if$
$until\ ((k = nt)\ or\ (part1 = verb)$
$or\ (part1 = adverb)\ or\ (part1 = preposition)$
$or\ (part1 = conjunction)\ or\ (part1 = marker)$
$or\ (part1 = constr)\ or\ (part1 = name))$
$end - of - loop$
end

9.3.4 Algorithm "Defining-form-of-text"

Begin $unit1 := Rc[1, unit]$
$part1 := Rc[1, tclass]$
$unit2 := Rc[2, unit]$
$part2 := Rc[2, tclass]$
$last\_unit := Rc[nt, unit]$
$last\_class := Rc[nt, tclass]$
$before\_last\_unit := Rc[nt - 1, unit]$
$before\_last\_class := Rc[nt - 1, tclass]$
$prep\_end := nil$
$if(lang = English)\ and\ (last\_unit := '?')$

$thenprep\_end := before\_last\_unit$
$if(part1 = verb)\ and$
$(Rc[1, subclass] \in \{imp, infin\})$
$and\ (last\_unit \neq '?')$
$then\ kindtext := imp$

Comment
The input text is a command.
Example: "Ship the containers to the factory 'Dawn' by 16:30."
End-of-comment

$if(part1 = verb)\ and$
$(Rc[1, subclass] = ftm)$

Comment
ftm means "form with time," it is the introduced (non traditional) property of the verbs in indicative and subjunctive moods.
End-of-comment

$and\ (last\_unit = '?')$
$then\ kindtext := genqs$

Comment
The input text is a general question, i.e., it is a question with the answer "Yes/No."
Example: "Does the joint-stock company 'Sail' export the goods to Bulgaria?"
End-of-comment

$loghelp1 := false$
$loghelp2 := false$
$loghelp3 := false$
$if\ qswd\_quant1 \mid qswd\_quant2$
$belongs\ toSetlextypes(Rc[1, subclass])$
$then\ if\ (lang = English)\ and\ (unit2 = 'times')$
$then\ loghelp1 := true$
$end - if$
$if(lang = German)and(unit2 = 'Mal')$
$then\ loghelp2 := true$
$end - if$
$if(lang = Russian)and(unit2 = 'raz')$
$then\ loghelp3 := true$
$end - if$
$if(loghelp1 = true)or(loghelp2 = true)$
$or(loghelp3 = true)$
$then\ begin\ mainpos := 1$
$pos := 2$
$kindtext := specqs\text{-}quant1$
end

Comment
Example: "How many times this year the textbook by Korobov was requested?"
End-of-comment

$elsebegin\ mainpos := 1$
$pos := 1$
$kindtext := specqs - quant2$
$var1 := 'S1'$
end
$end - if$

Comment

Example: "How many articles on organic chemistry by professor Igor Somov were published last year?"

End-of-comment

$if\ kindtext := nil$
$then$
$begin$
$loghelp7 := ((part1 = pronoun)$
$or\ ((part1 = preposition)\ and\ (part2 = pronoun)));$
$if(part1 = pronoun)$
$thenmainpos := 1$
$end - if$
$if(part1 = preposition)\ and\ (part2 = pronoun))$
$thenmainpos := 2$
$end - if$
$Set1 := Setlextypes(Rc[mainpos, subclass]);$
$n1 := Number\_of\_elem(Set1);$
$if\ (roleqswd \in Set1) and (n1 = 1)$
$thenkindtext := specqs - role$
$Processing - of - role - interrog - words$

Comment

The algorithm $Processing - of - role - interrog - words$ is described in next section.

End-of-comment

$(pos, mainpos)$
$end - if$
$if\ (roleqswd \in Set1)\ and\ (noun\_attach\_qswd \in Set1)$
$then$

Comment

Recognize one of two cases:

Case 1: Example – "What has he received?"

Case 2: Example – "What scholars from France did participate in this conference?"

End-of-comment

$Calltheprocedure"Right - noun(mainpos, posnoun)$
$If\ (posnoun = 0)$
$thenkindtext := specqs - role$
$Processing - of - role - interrog - words$

Comment

The algorithm $Processing-of-role-interrog-words$ is described in next section.

End-of-comment

$(pos, mainpos)$

Comment

Example: “Where did three-ton containers come from?”

Example (German): “Wo arbeitet Herr Professor Dr. Schulz?”

End-of-comment

$else$

Comment: $(posnoun > 0)$ end-of-comment

$noun1 = Rc[posnoun, unit]$

$if\ (noun1\ is associated with singular)$

$then\ var1 := 'x1'$

$kindtext := specqs-relat1$

Comment

Example: “What institute does Professor Igor Somov work at?”

End-of-comment

$else\ var1 := 'S1'$

$kindtext := specqs-relat2$

$end-if$

Comment

“What countries does the joint-stock company ‘Rainbow’ import the component parts from?”

End-of-comment

$End-if$

$End-if$

$if(kindtext = nil)\ and\ (nbqswd = 0)$

$then\ kindtext := stat$

$End-if$

Comment

“The joint-stock company ‘Rainbow’ has been exporting the goods to Bulgaria since 1999”

End-of-comment

end

9.4 Principles of Processing the Role Interrogative Words

Let’s agree to say that a sentence starts with an interrogative word *wd* if the first word of this sentence is either *wd* or a certain preposition followed by *wd*. For example, let’s agree that each of the questions Q1 = “Whom have you sent on a business trip to Prague?” and Q2 = “Whom did 3 two-ton containers come for?” starts with the interrogative word “whom.”

Let's split all interrogative words the considered questions could begin with into two groups. The first group includes, in particular, the expressions "how many," "how many times," and "what." The second group includes, in particular, the pronouns "who," "whom," "what," etc., and the adverbs "where," "when." Each word of this group together with a certain preposition is used to express a certain thematic role, i.e., semantic relationship between the verb and its dependent expression.

If an interrogative word *wd* does not require a preposition, then let's say that this word is used with the void (or empty) preposition *nil*. For example, the pair $(nil, whom)$ in a question Q1 is used to describe the thematic role "Object of action." The pair $(for, whom)$ in the question Q2 expresses the thematic role "Recipient." With respect to these observations, the words of the second group will be called *the role interrogative word combinations.*

An initial processing of the interrogative pronouns from the first group is done by the algorithm "Defining-form-of-text." A position of an interrogative word is the value of the output variable *mainpos*. The value of the variable *kindtext* indicates a subclass the considered interrogative word belongs to.

The role interrogative words (i.e. the words of the second group) together with the prepositions may form the sequences being the left segments of the questions. The question Q3 = "When and where three aluminum containers came from?" could serve as an example. That is why a special algorithm "Processing-of-role-interrog-words" is required. This algorithm called by the algorithm "Defining-form-of-text" uses a dictionary of the role interrogative words and word combinations *Rqs* being a part of the considered linguistic basis.

Let's agree to say that a sentence begins with an interrogative word *wd* if the first word of this sentence is either *wd* or a certain preposition followed by *wd*. For example, let's agree that each of the questions Q1 = "Whom have you sent on a business trip to Prague?" and Q2 = "Whom did 3 two-ton containers come for?"

Algorithm "Processing-of-role-interrog-words" is used for processing the role interrogative words located in the beginning of many questions. Such words are the interrogative pronouns ("who," "whom," "what," etc.) or pronominal adverbs ("where," "when").

External specification of the algorithm

"*Processing − of − role − interrog − words*"

Input:

nt – integer – the quantity of the text units;

Rc – classifying representation of the input text *T* (see Sect. 7.1);

pos – integer – the position of the interrogative word in *Rc*;

Rqs – dictionary of the role interrogative words (one of the components of *Lingb*);

Matr – matrix semantic-syntactic representation (MSSR) of the input text (see Sect. 8.3);

leftprep – string – the value of the preposition on the left;

numbent – integer – the quantity of objects mentioned in the text;

numbqswd – integer – quantity of the interrogative words that have been already found in the text;

posqswd – one-dimensional array of the length *nt*, where for $k \geq 1$, $posqswd\,[k]$ is either a position of the interrogative word having the number *k* in *Rc* or 0.

Output:

Matr, *numbent*, *numbqswd*, *posqswd*, *leftprep*.

$$Algorithm\ \text{“}Processing-of-role-interrog-words''$$

Begin

Comment (condition of application):

$$(roleqswd \in Setlextypes(Rc[mainpos, subclass])$$

$$and\ (kindtext = 'roleqs')$$

End-of-comment

$k1 := mainpos - 1$
$loop-until$
$k1 := k1 + 1$
$class1 := Rc[k1, tclass]$

The new book of the fragment to be obtained:

$k1 := k1 + 1$
$class1 := RC[k1, tclass]$
$if\ class1 = pronoun\ then\ numbqswd := numbqswd + 1$
$numbqswd := numbqswd + 1$
$if class1 = pronoun then$

Comment

Quantity of interrogative words already encountered in the text

End-of-comment

$posqswd\,[numbqswd] := k1$
$end-if$
$until\ class1 \neq \notin \{pronoun, preposition, conjunction, marker\}$
$end-of-loop$
$loop\ for\ m1\ from\ 1\ to\ numbqswd$
$p1 := posqswd[m1]$
$word1 := Rc[p1, unit]$
$if Rc[p1, tclass] = pronoun$
$then if(Rc[p1 - 1,\ tclass] = preposition$
$then leftprep = Rc[p1 - 1,\ unit]$
$else$
$begin$
$if(lang = 'German') or (lang = 'Russian')$
$then leftprep = nil$
$end-if$
$if(lang = 'English')\ and\ (m1 < numbqswd)$
$then\ leftprep := nil$

$end-if$
$if\ (lang = 'English')\ and\ (m1 = numbqswd)$
$then\ if\ prep\_end \neq nil$
$then\ leftprep = prep\_end$
$end-if$
$end-if$
end
$end-if$
$else\ leftprep := nil$ Comment: "when"in English, "wohin" in German
$end-if$
$if(Rc[p1, tclass] = pronoun)$
Find in the dictionary of the role interrogative word combinations *Rqs*
being a component of the considered linguistic basis *Lingb*
such ordered four-tuple with the least possible number of a row $n1$ that

$$Rqs[n1, prep] = leftprep$$

$$Rqs[n1, qswd] = word1$$

$else\ \{thecase\ of\ Rc[p1, tclass] = adverb\}$
Find in the dictionary of the role interrogative word combinations *Rqs*
being a component of the considered linguistic basis *Lingb*
such ordered four-tuple with the least possible number of a row $n1$ that
$Rqs[n1, qswd] = word1$
$end-if$
$role := Rqs[n1, relq]$
$Matr[p1, reldir, m1] := role$
$numbent := m1$
Comment
Quantity of entities mentioned in already analyzed fragment of the text
End-of-comment
$var1 := var('x', numbent)$
$Matr[p1, mark] := var1$
end
$end-of-loop\{for\ m1\}$
$end\ (of\ algorithm)$

9.5 An Algorithm of Searching for Semantic Connections of the Verbs

Let us start to formalize the conditions required for the existence of a semantic relationship between a meaning of a verbal form and a meaning of a word or word combination depending in a sentence on this verbal form.

9.5.1 Key Ideas of the Algorithm

Let's agree to use the term "noun group" for designation of the nouns and the nouns together with the dependent words representing the concepts, objects, and sets of objects. For example, let T1 = "When and where two aluminum containers with ceramic tiles have been delivered from?", T2 = "When the article by Professor P. Somov was delivered?" and T3 = "Put the blue box on the green case." Then the phrases "two aluminum containers," "the article by professor P. Somov," "blue box" are the noun groups.

Let's call "a verbal form" either a verb in personal or infinitive form or a participle. A discovery of possible semantic relationships between a verbal form and a phrase including a noun or an interrogative pronoun plays an important role in the process of semantic-syntactic analysis of NL-text.

Let's suppose that *posvb* is the position of a verbal form in the representation *Rc*, *posdepword* is the position of a noun or an interrogative pronoun in the representation *Rc*.

The input data of the algorithm "Find-set-of-thematic-roles" are real numbers *posvb*, *posdepword*, and two-dimensional arrays *Arls*, *Arvfr*, where *Arls* is the projection of the lexico-semantic dictionary *Lsdic* on the input text, $Arvfr$ is the projection of the dictionary of verbal-prepositional frames Vfr on the input text.

The purpose of the algorithm "Find-set-of-thematic-roles" is firstly to find the integer number *nrelvbdep* – the quantity of possible semantic relationships between the values of the text units with the numbers $p1$ and $p2$ in the representation *Rc*.

Secondly, this algorithm should build an auxiliary two-dimensional array *Arrelvbdep* keeping the information about possible semantic connections between the units of *Rc* with the numbers $p1$ and $p2$. The rows of this array represent the information about the combinations of a meaning of the verbal form and a meaning of the dependent group of words (or one word).

The structure of each row of the two-dimensional array *Arrelvbdep* with the indices of columns

$$linenoun,\ linevb,\ trole,\ example$$

is as follows.

For the filled-in row with the number k of the array *Arrelvbdep* $(k \geq 1)$

- *linenoun* is the ordered number of the row of the array *Arls* corresponding to the word in the position $p1$;
- *linevb* is the ordered number of the row of the array *Arls* corresponding to the verbal form in the position $p2$;
- *trole* is the designation of the semantic relationship (thematic role) connecting the verbal form in the position $p2$ with the dependent word in the position $p1$;
- *example* is an example of an expression in NL realizing the same thematic role.

The search of the possible semantic relationships between a meaning of the verbal form (VF) and a meaning of the dependent group of words (DGW) is done with the help of the projection of the dictionary of verbal-prepositional frames (d.v.p.f.) $Arvfr$ on the input text. In this dictionary such a template (or templates) is searched

that it would be compatible with the certain semantic-syntactic characteristics of the VF in the position *posvb* and the DGW with the number *posdepword* in *Rc*.

Such characteristics include, first of all, the set of codes of grammatic cases *Grcases* associated with the text-forming unit having the ordered number – value *posdepwd* ("the position of dependent word") in *Rc*.

Let's suppose that $Rc[posvb, tclass] = verb$. Then *Grcases* is a set of grammatic cases corresponding to the noun in the position $posdepword$.

9.5.2 Description of an Algorithm of Searching for Semantic Connections Between a Verb and a Noun Group

9.5.2.1 Purpose of the Algorithm "Find-set-relations-verb-noun"

The algorithm is to establish a thematic role connecting a verbal form in the position *posvb* with a word (noun or connective word) in the position *posdepword* taking into account a possible preposition before this word. As a consequence, to select one of the several possible values of a verbal form and one of the several possible values of a word in the position *posdepword*, three enclosed loops are required: (1) with the parameter being a possible value of the word in the position *posdepword*; (2) with the parameter being a possible value of the verbal form; (3) with the parameter being a verbal-prepositional frame connected with this verbal form.

9.5.2.2 External Specification of the Algorithm "Find-set-relations-verb-noun"

Input:

Rc – classifying representation;

nt – integer – quantity of the text units in the classifying representation *Rc*, i.e. the quantity of rows in *Rc*;

Rm – morphological representation of the lexic units of *Rc*;

posvb – integer – position of a verbal form (a verb in a personal or infinitive form, or a participle);

posdepword – integer – position of a noun;

Matr – initial value of MSSR of the text;

Arls – array – projection of the lexico-semantic dictionary *Lsdic* on the input text *T*;

Arvfr – array – projection of the dictionary of verbal-prepositional frames *Vfr* on the input text *T*.

Output:

arrelvbdep – one-dimensional array designed to represent the information about (a) a meaning of a dependent word, (b) a meaning of a verbal form, and (c) about a semantic relationship between the verbal form in the position *posvb* and the dependent word in the position *posdepword*;

nrelvbdep – integer – quantity of meaningful rows in the array *arrelvbdep*.

9.5.2.3 External Specifications of Auxiliary Algorithms

Specification of the algorithm

"*Characteristics − of − verbal − form*"

Input:

$p1$ – number of a row of *Rc* corresponding to a verb or a participle.

Output:

$form1$, $refl1$, $voice1$ – strings; their values are defined in the following way:

If $p1$ is the position of a verb, then $form1$ may have one of the following values: *indicat* (the sign of the indicative mood), *infinit* (the sign of the infinitive form of a verb), *imperat* (the sign of the imperative mood).

If $p1$ is the position of a participle, then $form1 := indicat$.

The string $refl1$ represents a value of the property "reflexivity"

The string $voice1$ takes the value *active* (the sign of the active voice) or *passive* (the sign of the passive voice).

The values of the parameters $form1$, $refl1$, $voice1$ are calculated based on the set of the numeric codes of the values of the morphological characteristics of the text unit with the ordered number $p1$.

Specification of the algorithm "*Range − of − sort*"

Input:

z – sort, i.e., an element of the set $St(B(Cb(Lingb)))$, where *Lingb* is a linguistic basis.

Output:

spectrum – set of all sorts being the generalizations of the sort z including the sort z itself.

9.5.2.4 Algorithm "Find-set-relations-verb-noun"

Algorithm

Begin

$Characteristics - of - verbal - form(posvb, form1, refl1, voice1)$

$nrelvbdep := 0$

Comment

Now the preposition is being defined

End-of-comment

$prep := leftprep$

Comment

Calculation of $posn1$ – position of the noun that defines the set of sorts of the text unit in the position *posdepword*

End-of-comment

$posn1 := posdepword$

Comment

Then the set of grammatic cases *Grcases* is being formed. This set will be connected with the word in the position *posdepword* in order to find a set of semantic relationships between the words in the positions *posvb* and *posdepword*.

End-of-comment

$t1 := Rc[posvb, tclass]$

$t2 := Rc[posvb, subclass]$

$p1 := Rc[posdepword, mcoord]$

$Grcases := Cases(Rm[p1, morph])$

$line1 := Matr[posn1, locunit]$

$numb1 := Matr[posn1, nval]$

Comment

The quantity of the rows with the noun meanings in *Arls*

End-of-comment

$loop\ for\ i1\ from\ line1\ to\ line1 + numb1 - 1$

Comment

A loop with the parameter being the ordered number of the row of the array *Arls* corresponding to the noun in the position $posn1$

End-of-comment

$Set1 := empty\ set$

$loop\ for\ j\ from\ 1\ to\ m$

Comment

m – semantic dimension of the sort system $S(B(Cb(Lingb)))$, i.e., the maximal quantity of incomparable sorts that may characterize one essence

End-of-comment

$current - sort := Arls[i1, st_j]$

$if\ current - sort \neq nil$

$then\ Range - of - sort(current - sort, spectrum)$

$Set1 := Set1 \cup spectrum$

$end - if$

Comment

For an arbitrary sort z the value *spectrum* is the set of all sorts being the generalizations of the sort z including the sort z itself

End-of-comment

$end - of - loop$

Comment

End of the loop with the parameter j

End-of-comment

Comment

Then the loop with the parameter being a value of the verbal form follows

End-of-comment

$line2 := Matr[posvb, locunit]$

$numQ2 := Matr[posvb, nval]$

Comment
Quantity of rows with the values of the verbal form in *Arls*
End-of-comment

$loop\ for\ i2\ from\ line2\ to\ line2 + numQ2 - 1$

Comment
A loop with the parameter being the ordered number of the row of the array *Arls* corresponding to the verb in the position *posvb*
End-of-comment

$current - pred := Arls\,[i2, sem]$
$loop\ for\ k1\ from\ 1\ to\ narvfr$
$if\ Arvfr\,[k1, semsit] = current - pred$
$then\ begin\ s1 := Arvfr\,[k1, str]$
$if\ ((prep = Arvfr\,[k1, sprep])\ and\ (s1 \in Set1)\ and\ (form1 = Arvfr\,[k1, form])\ and\ (refl1 = Arvfr\,[k1, refl])\ and\ (voice1 = Arvfr\,[k1, voice]))$
$then\ grc := arvfr\,[k1, grcase]$
$if\ (grc \in Grcases)$
$then$

Comment
The relationship exists
End-of-comment

$nrelvbdep := nrelvbdep + 1$
$arrelvbdep\,[nrelvbdep, linevb] := i2$
$arrelvbdep\,[nrelvbdep, linenoun] := i1$
$arrelvbdep\,[nrelvbdep, gr] := grc$
$arrelvbdep\,[nrelvbdep,\ role] := arvfr\,[k1, trole]$
$end - if$
$end - if$
end
$end - if$
$end - of - loop$
$end - of - loop$
$end - of - loop$

Comment
End of loops with the parameters $i1,\ i2,\ k1$
End-of-comment
end

9.5.2.5 Commentary on the Algorithm "Find-set-relations-verb-noun"

The quantity *nrelvbdep* of the semantic relationships between the verbal form and a noun depending on it in the considered sentence is found. Let's consider such sublanguages of English, German, and Russian languages that in all input texts a verb is always followed (at certain distance) by at least one noun.

The information about such combinations of the meanings of the verb V and the noun $N1$ that give at least one semantic relationship between V and $N1$ is represented in the auxiliary array *arrelvbdep* with the indices of the columns

$$linenoun,\ linevb,\ trole,\ example.$$

For arbitrary row of the array *arrelvbdep*, the column *linenoun* contains $c1$ – the number of such row of the array *Arls* that $Arls\,[c1, ord] = posn1$ (position of the noun $n1$).

For example, for Q1 = "When and where three aluminum containers with ceramics have been delivered from?" $Arls\,[c1, sem] = container.$

The column *linevb* contains $c2$ – the number of the row of the array *Arls* for which $Arls\,[c2, ord] = posvb$, i.e. the row $c2$ indicates a certain meaning of the verb V in the position *posvb*.

For example, for Q1 = "When and where three aluminum containers with ceramics have been delivered from?" the column $Arls\,[c2, sem] = delivery2$. The column *role* is designed to represent the possible semantic relationships between the verb V and the noun $N1$.

If $nrelvbdep = 0$ then the semantic relationships have not been found. Let's assume that this is not possible for the considered input language.

If $nrelvbdep = 1$ then the following meanings have been clearly defined: the meaning of the noun $N1$ (by the row $c1$), the meaning of the verb V (by the row $c2$), and the semantic relationship $arrelvbdep\,[nrelvbdep, role]$.

For example, for the question Q1 the following relationships are true: $V =$ "*delivered*," $N1 =$ "*containers*," $nrelvbdep = 1$, $arrelvbdep\,[nrelvbdep, role] = Object1$.

If $nrelvbdep > 1$ then it is required to apply the procedure that addresses clarifying questions to the user and to form these questions based on the examples from the column *example*.

9.5.3 Description of the Algorithm Processing the Constructs

$$Purpose\ of\ the\ algorithm\ "Find-set-relations-verb-construct"$$

The algorithm is used for establishing a thematic role connecting a verbal form in the position *posvb* with a construct in the position *posdep*, taking into account a possible preposition before this construct. As a consequence, the algorithm selects one of the several possible values of a verbal form. In order to do this, two concentric loops are required: (1) with the parameter enumerating a possible meaning of a verbal form; (2) with the parameter enumerating a possible verbal-prepositional frame associated with this verbal form.

External specification of the algorithm

Input:

posvb – integer – position of a verbal form (a verb in indicative mood or imperative mood or in infinitive form);

posdep (abbreviation of "position of dependent word") – integer – position of a construct (an expression representing a numerical value of a parameter);

*subclass*1 – string – designation of a construct's sort;

Matr – MSSR of the text;

Arls – projection of the lexico-semantic dictionary on the input text;

Arvfr – projection of the dictionary of verbal-prepositional frames on the input text;

*prep*1 – string – preposition related to a construct or a blank preposition *nil*.

Output:

arrelvbdep – two-dimensional array designed to represent the information about (a) a meaning of a verbal form and (b) a semantic relationship between the verbal form in the position *posvb* and the construct in the position *posdep*;

nrelvbdep – integer – the quantity of meaningful rows in the array *arrelvbdep*.

Algorithm "Find − set − relation − verb − construct"

Begin
startrow := *Matr*[*posvb*, *locunit*] − 1
*row*1 := *startrow*
numbvalvb := *Matr*[*posvb*, *nval*]
Comment
The quantity of possible meanings of a verbal form
End-of-comment
loop − *until*
*row*1 := *row*1 + 1
Current − *pred* := *Arls*[*row*1, *sem*]
*K*1 := 0; *log*1 := *false*
loop − *until*
*k*1 := *k*1 + 1
if (*Arvfr*[*k*1, *semsit*] = *current* − *pred*)
then if (*Arvfr*[*k*1, *str*] = *subclass*1) *and* (*Arvfr*[*k*1, *sprep*] = *prep*1)
then
Comment
A relationship exists
End-of-comment
nrelvbdep := 1
arrelvbdep[1, *linevb*] := *row*1
arrelvbdep[1, *role*] := *Arvfr*[*k*1, *trole*]
*log*1 := *true*
end − *if*

$end - if$
$Exit - when\ (log1 = true)$
$End - of - loop$
$Exit - when\ ((log1 = true)\ or\ (row1 = startrow + numbvalvb))$
$end - of - loop$

9.5.4 Description of the Algorithm "Find-set-thematic-roles"

Purpose of the algorithm

The algorithm is used for finding a set of thematic roles connecting a verbal form in the position *posvb* with a word (noun, connective word, construct) in the position *posdep*, taking into account a possible preposition before this word. In order to do this, three concentric loops are required: (1) with the parameter corresponding to a possible meaning of the word in the position *posdep*; (2) with the parameter corresponding to a possible meaning of a verbal form; (3) with the parameter corresponding to a verbal-prepositional frame associated with this verbal form.

External specification

Input:

Rc – classifying representation of a text;

nt – integer – quantity of the text units in *Rc*, i.e., quantity of the rows in *Rc*;

Rm – morphological representation of the lexic units of *Rc*;

posvb – integer – position of a verbal form (verb in a personal or infinitive form);

posdep (abbreviation of "position of dependent word") – integer – position of a dependent word (noun, interrogative pronoun, construct – an expression denoting a numerical value of a parameter);

Matr – string-numeric matrix – initial MSSR of a text;

Arls – array – projection of the lexico-semantic dictionary *Lsdic* on the input text *T*;

Arvfr – array – projection of the verbal-prepositional frame dictionary *Vfr* on the input text *T*.

Output:

*class*1 – string – designation of a class of a text unit in the position *posdep*;

*subclass*1 – designation of a subclass of a text unit in the position *posdep*;

arrelvbdep – two-dimensional array designed to represent the information about (a) a meaning of a dependent word, (b) a meaning of a verbal form, and (c) a semantic relationship between the verbal form in the position *posvb* and the dependent word in the position *posdep*;

nrelvbdep – integer – the quantity of meaningful elements in the array *arrelvbdep*.

Algorithm "*Find* − *set* − *thematic* − *roles*"

Begin

Fill in with 0 all numerical positions of the array *arrelvbdep* and fill in with the void element *nil* all string positions of the array *arrelvbdep*

*class*1 := *Rc* [*posdep*, *tclass*]
*subclass*1 := *Rc* [*posdep*, *subclass*]
*prep*1 := *Matr* [*posdep*, *prep*]
if (*class*1 = *noun*)
then Find − *set* − *relations* − *verb* − *noun*
(*Rc*, *Rm*, *posvb*, *posdep*, *prep*1, *Arls*, *Arvfr*, *Matr*,
nrelvbdep, *arrelvbdep*)
end − *if*
if (*class*1 = *constr*)
then Find − *set* − *relations* − *verb* − *construct*
(*posvb*, *posdep*, *prep*1, *Arls*, *Arvfr*, *Matr*,
nrelvbdep, *arrelvbdep*)
end − *if*

end

9.5.5 An Algorithm of Searching for a Semantic Relationship Between a Verb and a Dependent Expression

Purpose of the algorithm "*Semantic* − *relation* − *verbal* − *form*"

The algorithm is used (a) for finding a thematic role connecting a verbal form in the position *posvb* with an expression (noun or construct) in the position *posdep*, taking into account a possible preposition before this expression. As a consequence, (b) for selecting one of several possible values of a verbal form and one of several possible values of a word in the position *posdep*; (c) for entering the obtained information about a value of a verbal form, a value of a dependent text unit and a semantic relationship (i.e. a thematic role) to the MSSR *Matr*.

External specification of the algorithm

Input:

Rc – classifying representation;

nt – integer – quantity of the text units in *Rc*, i.e., a quantity of the rows in *Rc*;

Rm – morphological representation of the lexic units of *Rc*;

posvb – integer – position of a verbal form (a verb in a personal or infinitive form);

posdep – integer – position of a noun or a construct;

Arls – array – projection of the lexico-semantic dictionary *Lsdic* on the input text *T*;

Arvfr – array – projection of the dictionary of verbal-prepositional frames *Vfr* on the input text *T*;

Matr – initial configuration of MSSR of a text.

Output:

Matr – string-numeric matrix – transformed configuration of an initial matrix *Matr.*

External specification of the algorithm "*Select – thematic – roles*"

Input:

posvb – integer – position of a verbal form;

posdep – integer – position of a dependent text unit (noun or construct);

arrelvbdep – two-dimensional array representing the information about the possible combinations of (a) a meaning of a verbal form in the position *posvb*, (b) a meaning of a dependent unit in the position *posdep*, and (c) a thematic role *rel* realized in this combination;

nrelvbdep – integer – the quantity of meaningful rows in the array *arrelvbdep*, i.e., the quantity of possible semantic relationships between the considered verbal form and the unit depending on it.

Output:

*m*1 – integer – the ordered number of a certain meaningful row of the array *arrelvbdep*.

Parameter *m*1 takes a non-zero value as a result of processing the user's answer to the clarifying question of the linguistic processor. A user is asked to indicate what relationship among several semantic relationships is realized in the combination "a verbal form in the position *posvb* + a dependent unit in the position *posdep*."

With the help of the column *example* a user is given the examples of combinations with the semantic relationships that could be potentially realized between the text units in the positions *posvb* and *posdep*.

Algorithm "*Semantic – relation – verbal – form*"

Begin

Find – set – thematic – roles

(*Rc*, *Rm*, *posvb*, *posdep*, *Arls*, *Arvfr*, *Matr*, *nrelvbdep*, *arrelvbdep*)

Comment

Find the quantity of the elements of the array *nrelvbdep* and form the array *arrelvbdep* describing the possible semantic relationships between a verbal form and a text unit depending on it.

End-of-comment

$if\ nrelvbdep = 1\ then\ m1 := 1$

$else\ Select - thematic - roles$

$(posvb, posdep, nrelvbdep, arrelvbdep, m1)$

Comment

$m1$ – number of row of the array *arrelvbdep* that gives a combination of a value of a verbal form, a value of a dependent text unit, and a semantic relationship between a verbal form in the position *posvb* and a text unit in the position *posdep* taking into account a preposition that may be related to the position *posdep*.

End-of-comment

$end - if$

$rel1 := arrelvbdep[m1, role]$

$locvb := arrelvbdep[m1, linevb]$

Comment

A row of *Arls*

End-of-comment

$if\ (class1 = noun)\ then\ locnoun := arrelvbdep[m1, linenoun]$

Comment

A row of *Arls*

End-of-comment

$end - if$

Comment

⇒ Entering the information to *Matr* (see a description of *Matr* in Chap. 8)

End-of-comment

$Matr[posvb, posdir] := 0$

$Matr[posvb, locunit] := locvb$

$Matr[posvb, nval] := 1$

$if\ (class1 = noun)\ then\ Matr[posdep, locunit] := locnoun$

$end - if$

$Matr[posdep, nval] := 1$

$Matr[posdep1, posdir] := posvb$

$Matr[posdep, reldir] := rel1$

End

Commentary on the algorithm "$Semantic - relation - verbal - form$"

If $nrelvbdep > 1$ then it is required to address the clarifying questions to a user with the help of the column *example* of the array *arrelvbdep*. It would allow finding the following data:

locnoun – the row of *Arls* indicating a meaning of a text unit in the position *posdep* if this text unit is a noun;

locvb – row of *Arls* indicating a meaning of a verbal form in the position *posvb*;

*rel*1 – semantic relationship (thematic role) between the text units in the positions *posvb* and *posdep*.

If $nrelvbdep = 1$ then $m1 := 1$. Therefore,

$$locvb = arrelvbn[1, linevb];$$

$$locnoun = arrelvbn[1, linenoun];$$

$$rel1 = arrelvbn[1, role].$$

Then the obtained information is being entered to the matrix $Matr$:

$$Matr[posdep, locunit] := locnoun;$$

$$Matr[posdep, posdir] := posvb;$$

$$Matr[posndep, reldir] := rel1.$$

As a result of the conducted analysis, the meanings of both a verbal form in the position *posvb* and a noun in the position *posdep* are unambiguously defined. Therefore,

$$Matr[posvb, nval] := 1,$$

$$Matr[posdep, nval] := 1.$$

9.5.6 Final Part of a Description of an Algorithm of Processing Verbs

$$External\ specification$$

$$of\ the\ algorithm\ \text{“}Processing - verbal - form\text{”}$$

Input:

Rc – classifying representation;

nt – integer – the quantity of the text units in *Rc*, i.e. a quantity of the rows in *Rc*;

Rm – morphological representation of the lexic units of *Rc*;

Arls – array – projection of the lexico-semantic dictionary *Lsdic* on the input text *T*;

Arvfr – array – projection of the dictionary of verbal – prepositional frames *Vfr* on the input text *T*;

Matr – initial configuration of a matrix semantic-syntactic representation (MSSR) of the considered text;

pos – integer – the position of the considered verbal form;

posqswd $[1 : nt]$ – one-dimensional array storing the positions of interrogative words;

$pos - free - dep\,[1 : nt]$ – one-dimensional array storing the positions of the text units without discovered governing lexical units;

$numb - free - dep$ – integer – the quantity of lexical units without discovered governing lexical units;

nmasters $[1 : nt]$ – array representing the quantity of governing words for each text unit;

numbqswd – integer – the quantity of found interrogative words;

numbsit – integer – the quantity of situations mentioned in the analyzed text;

class – string representing the part of speech of the verbal form in the position *pos*.

Output:

Transformed configuration of MSSR *Matr*.

Algorithm "*Processing – verbal – form*"

Begin

$nsit := nsit + 1$

Comment

nsit - quantity of the situations already mentioned in a text

End-of-comment

$Matr\,[pos, mark] := Var('e', nsit)$

$verbpos := pos$

$if\ ((class = verb)\ and\ (numbqswd > 0))$

then

Comment

The governing arrows with the marks of the semantic relationships (thematic roles) are being drawn from the position of the verb in the main clause to the positions of the interrogative words

End-of-comment

$loop\ for\ k1\ from\ 1\ to\ numbqswd$

$p1 := posqswd\,[k1]$

$Matr\,[p1, posdir, 1] := pos$

$end - of - loop$

$numbqswd := 0$

$end - if$

$if\ numb - free - dep > 0$

Comment

There are free text units, i.e., such units for which a semantic-syntactic governance by another unit is not yet found

End-of-comment

$then\ loop\ for\ m1\ from\ 1\ to\ numb - free - dep$

$Semantic-relation-verbal-form$
$(Rc, nt, Rm, pos, pos-free-dep[m1],$
$Arls, Arvfr, Matr)$
$end-of-loop$
$end-if$
end

Example. Let Qs1 be the following marked-up representation of the question "When (1) and (2) where (3) the next (4) international (5) scientific (6) conference (7) 'COLING' (8) will be held (9) ? (10)"

If $pos = 9$ then the marked arrows from the position 9 to the positions 1 and 3 will be drawn (in the loop with the parameter $k1$ taking the values from 1 to $numbqswd$).

9.6 Processing of Adjectives, Prepositions, Cardinal Numerals, Names and Nouns

9.6.1 Processing of Adjectives

Description of the algorithm "$Processing-adjective$"

External specification

Input:
pos – integer – ordered number of a text unit being an adjective;
Matr – MSSR of a text;
Arls – array – projection of the lexico-semantic dictionary *Lsdic* on the input text *T*.

Output:
nattr – integer – the quantity of consecutive adjectives;
Attributes – a two-dimensional array with the columns

$place,\ property,$

where for every row k,$Attributes[k,\ place]$ is the position of an adjective in the classifying representation *Rc*, and $Attributes[k,\ property]$ is a compound semantic item corresponding to this adjective and taken from the array *Arls* – the projection of the lexico-semantic dictionary *Ldic* on the input text.

Example 1. Let Q1 = "Where (1) two (2) green (3) aluminum (4) containers (5) came (6) from (7) ? (8)" Then $nattr := 2$ and the array *Attributes* has the following structure:

place	property
3	$Color\,(z1,\, green)$
4	$Material\,(z1,\, aluminum)$
5	nil

Comment

In the end of the algorithm $Processing - noun$ the following operations, in particular, are performed:

- $nattr := 0$;
- the column *place* is being set to 0;
- the column *property* is being filled in with the string *nil* designating the void (empty) string.

$$Algorithm\ \text{“}Processing - adjective''$$

Begin
$nattr := nattr + 1$
$k1 := Matr\,[pos,\, locunit]$
$semprop := Arls\,[k1,\, sem]$
$Attributes\,[nattr,\, place] := pos$
$Attributes\,[nattr,\, prop] := semprop$
end

9.6.2 Processing of Prepositions, Cardinal Numbers, and Names

The algorithms “$Processing - preposition$” and “$Processing - cardinal - numeral$” are very simple. The first algorithm is designed to remember a preposition in the considered position *pos* with the help of the variable $leftprep$ (“preposition on the left”). The second algorithm transforms a lexical unit belonging to the class of cardinal numerals to the integer represented by this lexical unit.

For example, the words “three” and “twenty three” correspond to the numbers 3 and 23. The variable $leftnumber$ (“number on the left”) is designed to remember a number. Input parameters of these algorithms are the classifying representation of a text *Rc* and the variable *pos* (number of a row in *Rc*).

$$Algorithm\ \text{“}Processing - preposition''$$

Begin
$leftprep := Rc\,[pos,\, unit]$
end

$$Algorithm\ \text{“}Processing - cardinal - number''$$

Begin
$Leftnumber := Number(Rc[pos, unit])$
end

Description of the algorithm "*Processing – name*"

External specification of the algorithm

Input:
Rc – classifying representation of a text;
pos – position of an expression in inverted commas or apostrophes;
Matr – MSSR of a text
Output:
Transformed value of *Matr.*

Algorithm

Begin
$Matr[pos, posdir, 1] := pos - 1$
$Matr[pos, reldir, 1] :=$ '*Name*'
Comment
Meaning of these operations: a governing arrow with the mark "Name" is being drawn to an expression in inverted commas or apostrophes from a noun standing to the left from this expression.
End-of-comment
end

9.6.3 An Algorithm Searching for Possible Semantic Connections Between Two Nouns with Respect to a Preposition

Purpose of the algorithm

"*Find – set – relations – noun*1 – *noun*2"

Algorithm "*Find – set – relations – noun*1 – *noun*2" ("Find a set of semantic relationships between the noun 1 and the noun 2") allows to establish the semantic relationships that could exist between the noun in the position *posn*1 (going forward referred to as *noun*1) and the noun in the position *posn*2 (going forward referred to as *noun*2) under the condition that a certain preposition in the position between *posn*1 and *posn*2 is related to the second noon.

To do this, three loops are required: (1) with the parameter being a possible value of the word in the position *posn*1, (2) with the parameter being a possible value of

the word in the position *posn*2, (3) with the parameter being a prepositional frame connected with the considered preposition.

External specification of the algorithm

"*Find* – *set* – *relations* – *noun*1 – *noun*2"

Input:

Rc – classifying representation;

nt – integer – the quantity of the text units in the classifying representation *R*1, i.e., the quantity of rows in *Rc*;

Rm – morphological representation of the lexical units of *R*1;

*Posn*1 – integer – position of the first noun;

*Posn*2 – integer – position of the second noun;

Matr – MSSR of a text;

Arls – array – projection of the lexico-semantic dictionary *Lsdic* on the input text *T*;

Arfrp – array – projection of the dictionary of prepositional frames *Frp* on the input text *T*.

Output:

arrelvbdep – two-dimensional array designed to represent the information about a meaning of the first noun, a meaning of the second noun, and a semantic relationship between the word in the position *posn*1 and the dependent word in the position *posn*2; this array contains the columns *locn*1, *locn*2, *relname*, *example*;

*nreln*1*n*2 – integer – the quantity of meaningful rows in the array *arrelvbdep*.

Algorithm "*Find* – *set* – *relations* – *noun*1 – *noun*2"

Begin

*nreln*1*n*2 := 0

Comment

A preposition is being defined

End-of-comment

*prep*1 := *Matr*[*posn*2, *prep*]

Comment

A set of grammatic cases is being defined

End-of-comment

*p*1 := *Rc*[*posn*2, *mcoord*]

Grcases := *Cases*(*Rm*[*p*1, *morph*])

*line*1 := *Matr*[*posn*1, *locunit*]

*numb*1 := *Matr*[*posn*1, *nval*]

Comment

The quantity of the rows with the meanings of Noun 1 in *Arls*

End-of-comment

*loop for n*1 *from line*1 *to line*1 + *numb*1 − 1

Comment

A loop with the parameter being the row of the array *Arls* corresponding to a noun in the position *posn*1

End-of-comment

$Set1 := empty\ set$

$loop\ for\ j\ from\ 1\ to\ m$

Comment

m – semantic dimension of the sort system $S(B(Cb(Lingb)))$, i.e., the maximal quantity of incomparable sorts that may characterize one entity.

End-of-comment

$current-sort := Arls[n1, st_j]$

$if\ current-sort \neq nil$

$then\ Range-of-sort(current-sort, spectrum)$

$Set1 := Set1 \cup spectrum$

$end-if$

Comment

For an arbitrary sort z $spectrum(z)$ is the set of all sorts being the generalizations of the sort z including the sort z itself.

End-of-comment

$end-of-loop$

Comment

End of the loop with the parameter j

End-of-comment

Comment

Example:

$If\ u = dyn.phis.ob$

$then\ spectrum(u) = \{dyn.phys.ob,\ phys.ob,\ space.ob\}$

End-of-comment

$line2 := Matr[posn2, locunit]$

$numb2 := Matr[posn2, nval]$

Comment

The quantity of the rows of *Arls* with the meanings of *noun*2

End-of-comment

$loop\ for\ n2\ from\ line2\ to\ line2 + numb2 - 1$

Comment

A loop with the parameter being the number of a row of the array *Arls* corresponding to the noun in the position *posn*2

End-of-comment

$Set2 := empty\ set$

$loop\ for\ q\ from\ 1\ to\ m$

Comment

m – semantic dimension of the sort system $S(B(Cb(Lingb)))$, i.e. the maximal quantity of incomparable sorts that may characterize one essence.

End-of-comment

$current-sort := Arls[n2, st_q]$

$if\ current-sort \neq nil$

$then\ Range-of-sort\,(current-sort, spectrum)$
$Set2 := Set2 \cup spectrum$
$end-if$
$end-of-loop$
Comment
End of the loop with the parameter q
End-of-comment
$loop\ for\ k1\ from\ 1\ to\ narfrp$
Comment
Quantity of rows in the array $Arfrp$ – projection of the dictionary of prepositional frames Frp on the input text
End-of-comment
$if\ Arfrp\,[k1, prep] = prep1$
Comment
The required preposition is found
End-of-comment
$then\ begin\ s1 := Arfrp\,[k1, sr1]$
$s2 := Arfrp\,[k1, sr2]$
$if\ (s1 \in Set1)\ and\ (s2 \in Set2)$
$then\ if\ Arfrp\,[k1, grc] \in Grcases$
$then$
Comment
Relationship exists
End-of-comment
$nreln1n2 := nreln1n2 + 1$
$arreln1n2\,[nreln1n2, locn1] := n1$
$arreln1n2\,[nreln1n2, locn2] := n2$
$arreln1n2\,[nreln1n2, relname] := arfrp\,[k1, rel]$
$end-if$
$end-if$
end
$end-if$
$end-of-loop$
$end-of-loop$
$end-of-loop$
Comment
End of loops with the parameters $n1,\ n2,\ k1$
End-of-comment
end

Commentary on the algorithm

"$Find-set-relations-noun1-noun2$"

The quantity $nreln1n2$ of semantic relationships between the nouns in the positions $posn1$ and $posn2$ is found. The information about such combinations of the

meanings of the first and second nouns that give at least one semantic relationship between the elements in the positions $posn1$ and $posn2$ is represented in the auxiliary array $arreln1n2$ with the indices of the columns

$$locn1,\ locn2,\ relname,\ example.$$

For instance, the following relationships may take place for a certain row k :

$$arreln1n2[k, locn1] = p1,$$

$$arreln1n2[k, locn2] = p2,$$

$$arreln1n2[k, relname] = Against2,$$

$$arreln1n2[k, example] = \text{“}a\ remedy\ for\ asthma.\text{”}$$

The interpretation of the columns is as follows:

- The column $locn1$ contains $p1$ – the ordered number of a row of the array $Arls$ defining a possible meaning of the noun in the position $posn1$.
- The column $locn2$ contains $p2$ – the ordered number of a row of the array $Arls$ defining a possible meaning of the noun in the position $posn2$.
- The column *relname* is intended to represent the possible relationships between the nouns in the positions $posn1$ and $posn2$.

If $nreln1n2 = 0$ then semantic relationships are not found. Let's assume that this is impossible for the considered input language.

If $nreln1n2 = 1$ then the following meanings have been found: a meaning of the noun in the position $posn1$ (in the row $p1$), a meaning of the noun in the position $posn2$ (in the row $p2$), and a meaning of the semantic relationship $arreln1n2\,[nreln1n2, relname]$.

If $nreln1n2 > 1$ then it is required to apply a procedure that addresses clarifying questions to a user and to form these questions on the basis of the examples from the column *example*.

9.7 Finding the Connections of a Noun with Other Text Units

Plan of the algorithm “*Processing – noun*”

Begin

Entering to $Matr$ the information about a number (or cardinal numeral) and adjectives possibly standing to the left by applying the algorithm “Recording-attributes”

Generating a mark of an element and the type of a mark (applying the algorithm “Generating-mark”)

$If\ Rc[pos+1, tclass] = proper-noun\ then\ Processing-proper-noun$
$end-if$

Searching a semantic dependence from the noun being closest to the left and governed by the verb in the position *verbpos*

If there is no such dependence

Then in case when $verbpos \neq 0$ searching a semantic dependence from a verbal form in the position *verbpos*

else (i.e. in case when $verbpos = 0$) the position number *pos* is being entered to the array $pos-free-dep$ containing free text units.

end

External specification

of the algorithm "*Processing – noun*"

Input:

Rc – classifying representation;

nt – integer – quantity of text units in the classifying representation *Rc*, i.e., the quantity of rows in *Rc*;

Rm – morphological representation of the lexical units of *Rc*;

pos – integer – position of a noun;

Matr – initial value of MSSR of a text;

Arls – array – projection of the lexico-semantic dictionary *Lsdic* on the input text *T*;

Arvfr – array – projection of the dictionary of verbal – prepositional frames *Vfr* on the input text *T*;

Arfrp – array – projection of the dictionary of prepositional frames *Frp* on the input text *T*.

Output:

pos – integer – position of a text unit;

Matr – transformed value of the initial matrix *Matr.*

External specifications of auxiliary algorithms

Specification of the algorithm "*Find – noun – left*"

Input:

pos – integer – position of a noun.

Output:

posleftnoun – integer – position of the noun satisfying two conditions: (a) it is closest from the left to the position *pos*, (b) it may serve as a governing word for the noun in the position *pos* (see below the subsection "Description of the auxiliary algorithms").

Specification of the algorithm

"*Processing – proper – noun*"

Input:

pos – integer – position of a common noun or a proper noun followed by at least one proper noun;

Arls – projection of the lexico-semantic dictionary *Lsdic* on the input text;

Matr – initial configuration of MSSR of a text.

Output:

Matr – transformed configuration of MSSR of a text (see below the subsection "Description of the auxiliary algorithms").

Specification of the algorithm "*Processing – name*"

Input:

pos – position of a common noun followed by an expression in inverted commas or apostrophes;

Matr – initial configuration of MSSR of a text.

Output:

Matr – transformed value of MSSR of a text.

Specification of the algorithm "*Find – set – thematic – roles*"

The specification and the algorithm itself are described above.

Specification of the algorithm

"*Semantic – relation – verbal – form*"

The specification and the algorithm itself are described above.

Specification of the algorithm

"*Find – set – relations – noun*1 – *noun*2"

The specification and the algorithm itself are described above.

Specification of the algorithm

"*Select – governance – verb – noun*"

Input:

pos – integer – position of a text unit;

posvb – integer – position of a verbal form;

posleftnoun – integer – position of a noun on the left;

prep – string – value of a preposition related to the position *pos*.

Output:

res – string – takes the value 1 or 2 as a result of a clarifying dialogue with a user;

if a noun in the position *pos* directly depends on a verbal form in the position *posvb*, then *res* := 1;

if a noun in the position *pos* (taking into account a preposition) directly depends on the noun standing to the left in the position *posleftnoun*, then *res* := 2.

Specification of the algorithm

"*Select – relation – between – nouns*"

Input:

posleftnoun – integer – position of the noun 1;

pos – integer – position of the noun 2 standing to the right from the noun 1;

prep – string – value of a preposition (it could also be the void (empty) preposition *nil*) related to the noun 2;

*arreln*1*n*2 – two-dimensional array representing the information about possible combinations of the meanings of Noun 1 and Noun 2 and a semantic relationship between them, taking into account the preposition *prep*;

*nreln*1*n*2 – integer – the quantity of meaningful rows in the array *arreln*1*n*2, i.e., the quantity of possible semantic relationships between the considered nouns.

Output:

*m*2 – integer – ordered number of a certain meaningful row of the array *arreln*1*n*2.

The parameter *m*2 takes a non-zero value as a result of processing the user's answer to the clarifying question of the linguistic processor. A user is asked to indicate which of the several semantic relationships is realized in the combination

"Noun 1 in the position *posleftnoun* + Dependent Noun 2 in the position *pos*"

taking into account the preposition *prep*.

With the help of the column *example* a user is given the examples of combinations with the semantic relationships that could be potentially realized between the text units in the positions *posleftnoun* and *pos*.

Specification of the algorithm

"*Select – thematic – roles*"

External specification of this algorithm is described above.

Algorithm "*Processing – noun*"

```
Begin
    if leftnumber > 0 then Matr[pos, qt] := leftnumber
    end – if
    if nattr > 0
    then loop for m from 1 to nattr
        p1 := Attributes[m, place]
        Matr[p1, posdir, 1] := pos
        Semprop := Attributes[m, prop]
```

$Matr[p1, reldir, 1] := semprop$
$end - of - loop$
$end - if$
$leftnumber := 0$
$nattr := 0$
$Matr[pos, prep] := leftprep$
$leftprep := nil$
$Linenoun := Matr[pos, locunit]$
Comment
Number of the row in *Arls* containing an initial value of a noun
End-of-comment
$Sort1 := Arls[linenoun, st1]$
$if\ Sort1 \neq sit$
Comment
Situation
End-of-comment
$then\ numbent := numbent + 1$
Comment
Quantity of entities mentioned in the looked-through part of a text
End-of-comment
$end - if$
$gramnumber := Number(Rc[pos, mcoord])$
$if\ gramnumber = 1$
$then\ var1 := Varstring('x', numbent)$
$end - if$
$if\ (gramnumber = 2)\ or\ (gramnumber = 3)$
$then\ var1 := Varstring('S', numbent)$
$end - if$
$Matr[pos, mark] := var1$
$Find - noun - left(pos, posleftnoun)$
$if\ posleftnoun = 0$
Comment
To the left from the position *pos* there are no nouns that may govern a noun in the position *pos*
End-of-comment
$then\ if\ verbpos = 0$
$then\ numb - free - dep := numb - free - dep + 1$
$K1 := numb - free - dep$
$pos - free - dep[k1] := pos$
$else\ posvb := verbpos$
$Semantic - relation - verbal - form(posvb, pos, Matr)$
$else$
Comment
In the case when $posleftnoun > 0$
End-of-comment

$Find-set-relations-noun1-noun2$
$(posleftnoun, Matr[pos, prep], pos, Matr, nreln1n2, arreln1n2)$
Comment
Possible semantic connections (and their quantity) between the considered noun in the position *pos* and the closest from the left noun in the position $postleftnoun$ are being found
End-of-comment
$if\ (nreln1n2 = 0)$
Comment
There is no semantic-syntactic governance by the preceding noun
End-of-comment
$then\ posvb := verbpos$
$if\ posvb > 0$
$then\ Semantic-relation-verbal-form(posvb, pos, Matr)$
$else$
Comment
In the case when $posvb = 0$
End-of-comment
$numb-free-dep := numb-free-dep + 1$
$K1 := numb-free-dep$
$pos-free-dep[k1] := pos$
$end-if$
$end-if$
Comment
The case when $nreln1n2 = 0$ is considered
End-of-comment
$if\ (nreln1n2 > 0)$
Comment
There is an opportunity of semantic-syntactic governance by the preceding noun
End-of-comment
$then\ posvb := verbpos$
$if\ posvb > 0$
$then\ Find-set-thematic-roles$
$(posvb, pos, class1, subclass1, Matr, nrelvbdep, arrelvbdep)$
$if\ (nrelvbdep = 0)$
Comment
There is no semantic connection with a verbal form
End-of-comment
$then\ if\ (nreln1n2 = 1)$
$then\ m2 := 1$
Comment
$m2$ – number of an element of the array $arreln1n2$ which provides the information about the connection between $posleftnoun$ and *pos* for $Matr$
End-of-comment
$else\ Select-relation-between-nouns$

$(posleftnoun, Mate[pos, prep], pos, nreln1n2, arreln1n2, m2)$
$end-if$

Adding the information about a connection between the text units in the positions $posleftnoun$ and pos to $Matr$; this information is taken from the position $m2$ of the array $arreln1n2$;

$end-if$

Comment
A case when $nrelvbdep = 0$
End-of-comment

$if\ (nrelvbdep > 0)$

Comment
A connection with a verb is possible
End-of-comment

$then\ if\ (nreln1n2 > 0)$

Comment
A connection with a preceding noun is also possible
End-of-comment

$then\ Select-governance-verb-noun$
$(posvb, Matr[pos, prep], posleftnoun, pos, res)$

Comment
$res = 1 \Rightarrow$ a connection with a verb;
$res = 2 \Rightarrow$ a connection with a noun in the position $posn1$
End-of-comment

$if\ (res = 1)$
$then\ if\ (nrelvbdep = 1)\ then\ m1 := 1$
$else\ Select-thematic-roles$
$(posvb, Matr[pos, prep], pos, nrelvbdep, arrelvbdep, m1)$
$end-if$

Comment
Recording an information about a connection between a verbal form in the position $posvb$ and a noun in the position pos to $Matr$; this information is taken from the row $m1$ of the array $arrelvbdep$
End-of-comment

$nmasters[pos] := nmasters[pos] + 1$

Comment
A new governing arrow to the position pos is found
End-of-comment

$d := nmasters[pos]$
$Matr[pos, posdir, d] := posvb$
$Matr[pos, reldir, d] := arrelvbdep[m1, role]$
$Matr[posvb, locunit] := arrelvbdep[m1, linevb]$
$Matr[posvb, nval] := 1$
$Matr[pos, locunit] := arrelvbdep[m1, linenoun]$
$Matr[pos, nval] := 1$
$end-if$
$if\ (res = 2)$

Comment

There is no connection with a verbal form but there is a connection with the noun in the position *posleftnoun*

End-of-comment

then if $(nreln1n2 = 1)$ *then* $m2 := 1$
else
$Select - relation - between - nouns$
$(posleftnoun, prep, pos, nreln1n2, arreln1n2, m2)$
$end - if$

Comment

Recording the information about a semantic connection between the nouns in the positions *posleftnoun* and *pos* to *Matr* taking into account the preposition *prep* (which could also be a blank preposition *nil*); this information is taken from the row $m2$ of the array $arreln1n2$

End-of-comment

$Matr[posleftnoun, locunit] := arreln1n2[m2, locn1]$
$Matr[posleftnoun, nval] := 1$
$Matr[pos, locunit] := arreln1n2[m2, locn2]$
$Matr[pos, nval] := 1$
$Matr[pos, posdir, 1] := posleftnoun$
$Matr[pos, reldir, 1] := arreln1n2[m2, role]$
$end - if$
$end - if$
$end - if$
$end - if$
$end - if$
$if\ Rc[pos + 1, subclass] = proper - noun$
then logname :=(words in the positions *pos* and $pos + 1$ may be connected with the same grammatical case) and (semantic units corresponding to these words in the array *Arls* have the same set of sorts in *Arls*);
$if\ logname = True\ then\ Processing - proper - noun(pos)$
$end - if$
$end - if$
$if\ Rc[pos + 1, subclass] = Name\ then\ Processing - name(pos)$
$end - if$
$leftprep := nil$
$leftnumber := 0$
$nattr := 0$

Set to nil the column *place* of the array *Attributes*;

Fill in with the string *nil* – the designation of the void string – the column *prop* of the array *Attributes*.

End

Comment

End of the algorithm "$Processing - noun$"

End-of-comment

Description of auxiliary algorithms
Description of the algorithm "$Find-noun-left''$
External specification (see above)
Algorithm
Begin

$posleftnoun := 0$
$p1 := pos$
$loop-until\ p1 := p1 - 1$
$classleft := Rc[p1, tclass]$
$if\ classleft = noun\ then\ posleftnoun := p1$
$end-if$
$exit-when\ (p1 = 1)\ or\ (posleftnoun > 0)\ or\ (classleft \in \{verb, participle, adverb, pronoun, construct, marker\})$
$end-of-loop$

end

Example 1. Let Q1 = "How many containers with Indian ceramics came from Novorossiysk?"

Let's transform the question Q1 into the following marked-up representation: "How many (1) containers (2) with (3) Indian (4) ceramics (5) came (6) from (7) Novorossiysk (8) ? (9)" Let $pos = 5$ (position of the word "ceramics"). Then after the termination of the algorithm, $posleftnoun = 2$ (the position of the word "containers").

Description of the algorithm "$Processing-proper-noun$"
External specification (see above)
Algorithm
Begin

$k1 := pos + 1$
$while\ Rc[k1, tclass] = proper-noun\ loop$
$m1 := Matr[k1, locunit]$

Comment
The first and the only row of the array *Arls* containing the information about the unit $Rc[k1, unit]$ is found
End-of-comment

$Matr[k1, posdir, 1] := pos$

Comment
A governing arrow is drawn from the element in the position *pos* to the element in the position $k1$
End-of-comment

$sem1 := Arls[m1, sem]$
$Matr[k1, reldir, 1] := sem1$
$k1 := k1 + 1$
$end-of-loop$
$pos := k1 - 1$

end

Example 2. Let Q2 = "How many articles by professor Igor Pavlovich Nosov were published in 2008?" Then, as a result of applying the algorithm "*Processing* – *proper* – *noun*" with the parameter $pos = 4$ (position of the word "professor"), the governing arrows will be drawn (by transforming MSSR $Matr$) from the position pos to the positions $pos + 1$, $pos + 2$, $pos + 3$ corresponding to the fragment "Igor Pavlovich Nosov."

9.8 Final Step of Developing an Algorithm Building a Matrix Semantic-Syntactic Representation of the Input Text

9.8.1 Description of the Head Module of the Algorithm

In order to facilitate the understanding of the head module of the algorithm building an MSSR of the input NL-text, its external specification (developed in the Sect. 9.2) is given below.

*External specification of the algorithm BuildMatr*1

Input:
Lingb – linguistic basis;
T – input text;
lang – string variable with the values *English*, *German*, *Russian*.
Output:
nt – integer – the quantity of elementary meaningful text units;
Rc – classifying representation of the input text;
Rm – morphological representation of the input text;
$Arls$ – two-dimensional array – projection of the lexico-semantic dictionary *Lsdic* on the input text T;
$Arvfr$ – two-dimensional array – projection of the dictionary of verbal – prepositional frames Vfr on the input text T;
$Arfrp$ – two-dimensional array – projection of the dictionary of prepositional semantic-syntactic frames Frp on the input text T;
$Matr$ – matrix semantic-syntactic representation (MSSR) of the input text;
numbqswd – the variable representing the quantity of interrogative words in a sentence;
$pos - free - dep$ – one-dimensional array of integers;
$posvb$, $numb - free - dep$ – integers;
$nmasters[1 : nt]$ – one-dimensional array representing the quantity of governing words for each text unit.

9.8.2 External Specifications of Auxiliary Algorithms

Specification of the algorithm

"*Build – compon – morphol – representation*"

Input:
Lingb – linguistic basis;
T – NL-text.
Output:
Rc – classifying representation of the text *T*;
nt – integer – the quantity of elementary meaningful text units in the input text *T*, i.e. the quantity of meaningful rows in the classifying representation *Rc*;
Rm – morphological representation of the text *T*.

Specification of the algorithm

"*Build – projection – lexico – semantic – dictionary*"

Input:
Rc, *nt*, *Rm*;
Lsdic – lexico-semantic dictionary.
Output:
Arls – two-dimensional array – projection of the dictionary *Lsdic* on the input text .

Specification of the algorithm

"*Build – projection – verbal – frames – dictionary*"

Input:
Rc, *nt*, *Rm*, *Arls*;
Vfr – dictionary of verbal – prepositional semantic-syntactic frames.
Output:
Arvfr – two-dimensional array – projection of the verbal-prepositional frame dictionary *Vfr* on the input text .

Specification of the algorithm

"*Build – projection – prepositional – frames – dictionary*"

Input:
Rc, *nt*, *Rm*, *Arls*;
Frp – dictionary of prepositional semantic-syntactic frames.

Output:

$Arfrp$ – two-dimensional array – projection of the dictionary of prepositional frames Frp on the input text .

9.8.3 Algorithm of Building an MSSR of the Input Text

Algorithm BuildMatr

Begin
Build − compon − morphol − representation
(T, Rc, nt, Rm)
Build − projection − lexico − semantic − dictionary
$(Rc, nt, Rm, Lsdic, Arls)$
Build − projection − verbal − frames − dictionary
$(Rc, nt, Rm, Arls, Vfr, Arvfr)$
Build − projection − prepositional − frames − dictionary
$(Rc, nt, Rm, Arls, Frp, Arfp)$
Forming − initial − values − of − data
Defining − form − of − text
$(nt, Rc, Rm, leftprep, mainpos, kindtext, pos)$
loop − until
$pos := pos + 1$
$class := Rc[pos, tclass]$
case class of
preposition : *Processing − preposition* $(Rc, pos, leftprep)$
adjective : *Processing − adjective* $(Rc, pos, nattr, Attributes)$
cardinal − numeral : *Processing − cardinal − numeral* $(Rc, pos, numb)$
noun : *Processing − noun*
$(Rc, Rm, pos, Arls, Arfrp, Matr, leftprep, numb, nattr, Attributes)$
verb : *Processing − verbal − form*
$(Rc, Rm, pos, Arls, Rqs, Arvfr, Matr, leftprep)$
conjunction : *Empty operator*
construct : *Processing − construct*
name : *Processing − name*
marker : *Empty operator*
end − case − of
exit − when $(pos = nt)$
end

Thus, in this and previous sections of this chapter the algorithm *BuildMatr*1 has been developed. This algorithm finds (a) semantic relationships between the units of NL-text, (b) specific meanings of verbal forms and nouns from the text. This information is represented in the string-numerical matrix *Matr.*

The texts processed by the algorithm may express the statements (the descriptions of various situations), questions of many kinds, and commands. The texts may contain the verbs (in infinitive form, indicative or imperative mood), nouns, adjectives, numerical values of the parameters (constructs), cardinal numerals, digital representations of the numbers, interrogative words (being pronouns or adverbs), and the expressions in inverted commas or apostrophes serving as the names of various objects.

The algorithm *BuldMatr*1 is original and is oriented at directly discovering semantic relationships in NL-texts. It should be underlined that the algorithm *BuldMatr*1 is multilinguial: it carries out a semantic–syntactic analysis of texts from the sublanguages of English, German, and Russian languages being of practical interest.