Analysis of Natural Language Sentences by Methods of the Theory of Graphs and the Theory of Sets

Andrey Alyoshintsev
Moscow Technical University of communication and informatics
Moscow, Russia
alyoshintsev@mail.ru,

Alexander Sak
Moscow State University of Civil Emgineering
Moscow, Russia
sak_inter@mail.ru

***Abstract*—Natural language sentence can be represented by means of graphs, where words, groups of pixels or variants of decisions are used as vertices, and as edges is the relationship between words in a sentence, elements of images or decisions. In most sentences, the relations of subordination and linear order are related. The representation of the syntactic structure of the sentence in the form of a subordination tree is used in generative grammars of the language and in the algorithms of syntactic analysis. The tree is built, starting from the distribution of lexical units of the sentence by parts of speech, and then there is a transfer from the subordination tree to the tree of the components. A binary search tree is a kind of data structure that corresponds to the representation of the sentence in the form of a tree of components. When a tree graph is built it's possible to proceed to the analysis of a lexical expression by comparing the intersection of sets representing it with the dictionary expressions in order to reveal the maximum coincidence between them.**

INTRODUCTION

Graphs are one of the most important areas of the theory of computing systems. This is an abstract concept through which you can describe a variety of real phenomena such as the organization of transport systems, human relationships, and representation of the data structure. In linguistics, graph theory solves many problems associated with the representation of formal relations between the components of a sentence.

A binary decision tree is a data structure in the form of a binary tree, each node of which is bound to the decision choice function. The decision choice function is applied to an unknown feature vector and determines which child node of the current node should be processed further – the left one or the right one. A similar algorithm is observed when constructing a tree of components when analyzing sentences.

A correct description of the sentence by means of the graph allows to use to a certain extent, certain semantic links between lexemes apart from the syntactic ones.

I. GENERAL IDEA OF THE TEXT AND SPEECH SYNTHESIS WHEN CREATING SYSTEMS OF ARTIFICIAL I NTELLIGENCE (AI)

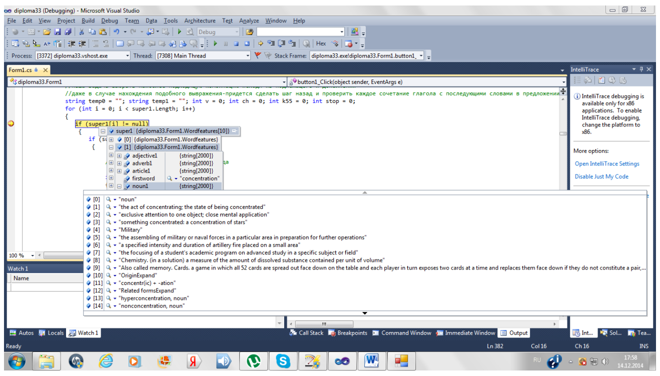
Artificial Intelligence (AI) is an area of science and technology, focused on the creation of software and hardware for solving intellectual problems [7]. Such tasks include interpretation and synthesis of natural language texts, interpretation and synthesis of speech, control of robots, and analysis of visual information and so on. Development of knowledge-based systems prevails in the development of intelligent systems, which construction is the main direction of artificial intelligence. Knowledge-based or expert systems develop in a fundamental direction in terms of models and methods for processing natural language (NL) towards:

1) creation of methods, models and algorithms for Semantic analysis and interpretation of the NL;
2) processing of continuous texts;
3) processing of speech acts.

In the applied aspect, the Natural Language Systems (NL-systems) also occupy a leading place in terms of research, such as speech NL-systems, fused text processing systems and others.

I. INTELLIGENT DIALOG SYSTEMS (IDS)

At present, the subsystems for processing text information are increasingly included in the complex data processing and analysis systems [4].


diploma33 (Debugging) - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Process: [3372] diploma33.vshost.exe
Thread: [7308] Main Thread
Stack Frame: diploma33.exe!diploma33.Form1.button1_
Form1.cs
diploma33.Form1
button1_Click(object sender, EventArgs e)
IntelliTrace
Watch 1
Name
Call Stack Breakpoints Command Window Immediate Window Output
Ln 382 Col 16 Ch 16 INS

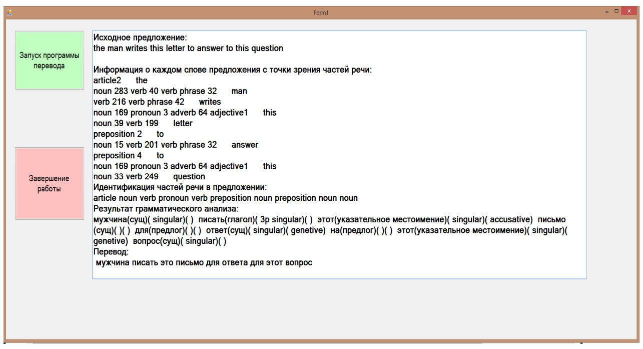
```
//даже в случае нахождения подобного выражения-придется сделать шаг назад и проверить каждое сочетание глагола с последующими словами в предложении
string temp0 = ""; string temp1 = ""; int v = 0; int ch = 0; int k55 = 0; int stop = 0;
for (int i = 0; i < super1.Length; i++)
{
    if (super1[i] != null)
    {
        if (su
        {
```

```
super1  {diploma33.Form1.Wordfeatures[10]}
  [0] {diploma33.Form1.Wordfeatures}
  [1] {diploma33.Form1.Wordfeatures}
    adjective1   {string[2000]}
    adverb1      {string[2000]}
    article1     {string[2000]}
    firstword    "concentration"
    noun1        {string[2000]}
[0]  "noun"
[1]  "the act of concentrating; the state of being concentrated"
[2]  "exclusive attention to one object; close mental application"
[3]  "something concentrated: a concentration of stars"
[4]  "Military"
[5]  "the assembling of military or naval forces in a particular area in preparation for further operations"
[6]  "a specified intensity and duration of artillery fire placed on a small area"
[7]  "the focusing of a student's academic program on advanced study in a specific subject or field"
[8]  "Chemistry. (in a solution) a measure of the amount of dissolved substance contained per unit of volume"
[9]  "Also called memory. Cards. a game in which all 52 cards are spread out face down on the table and each player in turn exposes two cards at a time and replaces them face down if they do not constitute a pair,...
[10] "OriginExpand"
[11] "concentr(ic) + -ation"
[12] "Related formsExpand"
[13] "hyperconcentration, noun"
[14] "nonconcentration, noun"
```

IntelliTrace debugging is available only for x86 applications. To enable IntelliTrace debugging, change the platform to x86.

Fig.1 Purpose of our program variables

If such subsystems are designed to work with data in several languages, they must tackle the task of automatically translating from one language to another.



```
Исходное предложение:
the man writes this letter to answer to this question

Информация о каждом слове предложения с точки зрения частей речи:
article2      the
noun 283 verb 40 verb phrase 32      man
verb 216 verb phrase 42      writes
noun 169 pronoun 3 adverb 64 adjective1      this
noun 39 verb 199      letter
preposition 2      to
noun 15 verb 201 verb phrase 32      answer
preposition 4      to
noun 169 pronoun 3 adverb 64 adjective1      this
noun 33 verb 249      question
Идентификация частей речи в предложении:
article noun verb pronoun verb preposition noun preposition noun noun
Результат грамматического анализа:
мужчина(сущ)( singular)( )  писать(глагол)( 3p singular)( )  этот(указательное местоимение)( singular)( accusative)  письмо
(сущ)( )( )  для(предлог)( )( )  ответ(сущ)( singular)( genetive)  на(предлог)( )( )  этот(указательное местоимение)( singular)(
genetive)  вопрос(сущ)( singular)( )
Перевод:
 мужчина писать это письмо для ответа для этот вопрос
```

Fig.2 Form1interface with the implementation of a working example of translation.

II. The main knowledge representating models in intelligent systems

Once knowledge composition and structure are determined, a representation model which is the most adequate and effective in given area is selected. Examples of such models can be:

1) *Deductive mode*l. The problem to be solved is written down in the way of some formal system statements (for example, in the calculation of first-order predicates).

2) *Inductive model*. There is a mechanism which is used in order to obtain general conclusions from a set of particular statements which can be either probabilistic or logical, depending on the specifics of the phenomenon studied.

3) Model of pseudo-physical logic. Zadé linguistic variables or order scales are used as propositional variables. They constitute a class of deductive formal systems.

4) *Models of functional networks*. Reflecting some decomposition of a certain computational or informational procedure, where the arcs show the functional connection between the parts the decomposition results in (for example, a program block diagram, etc.)

5) *Script models*. Homogeneous networks where the relation of a non-rigorous order acts as the only relation, which semantics may be different (for example, all possible sequences of events are a network schedule, etc.)

6) *Models of semantic networks*. An oriented graph with marked arcs and states (i.e., the vertices of the network can have different interpretations, and arc-relations belong to different types: logical, linguistic, set-theoretic, and quantified.)

7) *Frame model.* Formalized model for displaying an abstract image or situation: $F = \{<I, v_j\ g_j\ [p_j], \ldots, v_k\ g_k\ [p_k]>$,

where I is the name of the frame; v_j - the name of the j-th slot; g_j is the value of the j-th slot; p_j - procedures attached to slot j.
The values of slots can be the names of other frames, providing a link between them.

There are prototype frames stored in the Knowledge Base (KB) and instance frames that are created on the basis of prototype frames to display specific situations based on the incoming data.

TABLE I. Analysis of an example of a frame prototype for the situation "Taking an exam at the university"

Taking an exam at the university	
An examined one	(a student, a postgraduate student, an applicant, a group of students)
Examiners	(a lecturer, a lecturer assistant, a commission)
A subject/ discipline	(name of a subject/discipline)
Results	(a mark, obtained points)
Place / time	(exams schedule)

The example shows an important property of frames, consisting on the fact that the removal of any actant from this description leads to the loss of the properties determining the essence of this "passing an exam at the university" situation.
8) *Production model*. The model is based on rules that allow you to represent knowledge in form of sentences like: IF {<condition>} THAT {<action>} [ELSE {<action>}]

III. Expert knowledge

Expert knowledge will be information about possible actions taken to shift from one projected situation to another, as well as a sequence of actions to transform the project, which a designer or a design system is recommended to perform in order to achieve a given goal [11] . The production and frame method of knowledge representation is used in the construction of intellectual systems, that is, systems based on various aspects of the formalization of the concept of "knowledge" and logical inference.

When building intelligent systems, 4 groups of methods for representing knowledge became classical.

1) *A logical represen*tation based on the first-order predicates logic uses, as a rule, the means of the PROLOGUE language and numerous extensions of this language.

2) *Network representation*, in which a set of knowledge is represented as a graph, the vertices of which are objects of the domain, and the arcs are different relations between objects. Apart from the term "network representation", the term "semantic network" is also often used. Further in work we will describe it in more detailed way.

3) *A hierarchical representation* based on a hierarchy of concepts related to each other by means of inheritance bonds. Hierarchical views include frame views, scripts, etc.

4) *Production representation*, in which knowledge is encoded by sets of elementary actions, which application can lead to a solution of the problem.

IV. SEMANTIC NETWORKS

Semantic networks are an important model of knowledge representation. The concept is introduced to represent semantic links between words. Semantic networks are not a homogeneous class of representation schemes. A common feature of semantic networks is the similarity of a formal notation (a directed graph with marked vertices and edges) and the main principle consists on elements of knowledge stored adjacent if they are semantically related.

By a semantic network, we mean a directed graph with labeled vertices and arcs, in which the vertices correspond to specific objects, and the connecting arcs reflect relationships between them.

The problem of finding a solution in a semantic network knowledge base is reduced to the problem of finding a fragment of a network corresponding to a certain subnet having to do with the question posed.

There are several classifications of semantic networks:

1) by the number of types of relations (homogeneous ones are categorized by a single type of relationship; heterogeneous ones are categorized by various types of relations);

2) by types of relations (binary ones where relations connect two objects; *n-ary* ones connecting more than two concepts).

Semantic networks relations can be divided as follows:

1) *linguistic ones*, which include relations such as “object”, “agent”, “condition”, “place”, “tool”, “goal”, “time”, etc .;

2) *attributive ones*, which include the shape, size, color, etc .;

3) *characterization of verbs*, i.e. gender, tense, tilt, pledge, number;

4) *logical ones*, ensuring the execution of operations for calculating statements (disjunction, conjunction, implication, denial);

5) *quantified ones,* using quantifiers of community and existence;

6) *set-theoretic ones*, including the concepts of "element of the set", "subset", "superset" and others.

The basis of the semantic network is events, attributes, sets of features and procedures.

Events are judgments, facts, results of observations, recommendations. They can be represented by word combinations and numbers, grouped thematically or functionally into sections, divided into characterized ones and characterizing ones (events-features: for example, “it is raining” for the event “rainy weather”).

An attribute is a characterizing event that has several meanings (for example, “weather” is an attribute of “season”). Several features can be combined into a complex characterizing an event to a greater extent than a single feature. A procedure is a specific component of network that performs conversion of information. It allows you to calculate the values of some attributes on the basis of others, operating with both numbers and symbols

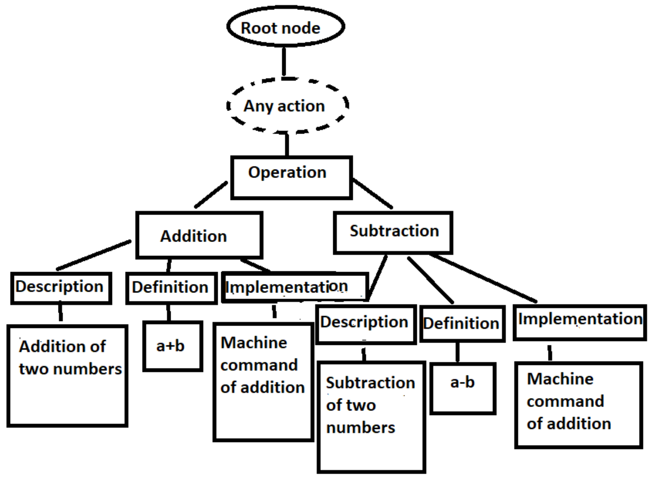


Fig.3 A semantic network describing simplest arithmetic operations

Fig. 3 presents an example of a semantic network that describes the simplest arithmetic operations. The expressive power of semantic networks is somewhat weaker than that of the logic of predicates. However, compared with logic of predicates, semantic networks take advantage of placing all well-known information about this or that concept around a corresponding vertex. A predicate is able to take over some participants of the situation expressed (verbalized) by given predicate and binary links between the predicate and each of its actors, i.e. participants involved in the situation, represent linguistic relationship (valence in Russian and deep cases in English linguistic tradition).

Knowledge that information about these properties is kept in certain place of a dictionary article as a verb regime model eases NL processing. The semantic network indicates, as a rule, three types of main vertices:

vertex situations (states, processes, etc.), expressed by predicates;

vertex concepts (abstract and physical);

vertex-characteristics (optional).

Semantic network uses the following types of relationships:

1) *set-theoretic relations* ("element-set", "part-whole", "set-subset", etc.);

2) *logical relationships* (AND, OR, NOT);

3) *quantified relations* (□, □);

4) *linguistic relations* (binary named relations between a predicate, reflecting a specific situation in the problem area and situation actors, that is, "roles" or participants.

More often, the following deep cases are used to define linguistic relationships:

- agent (A) is an animated initiator of action;
- counterparty (C) is a force which the action is directed against;
- object (O) is a thing that is an action target;
- addressee (D) is a person who is done good or damage as a result of this action;
- tool (T) is an inanimate object or force causing given action or given state;
- result (R) is a thing that occurs as a result of actions, etc.

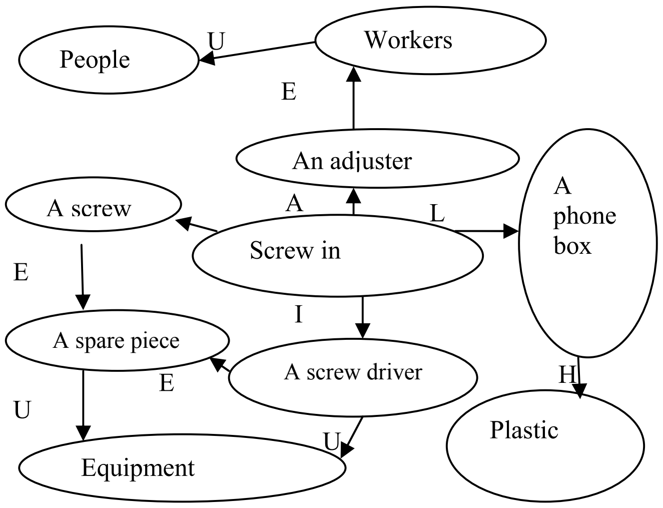


Fig.4. An example of a fragment of a semantic network for the “screwing” situation

The Fig. 4 shows an example of a fragment of a semantic network describing the “screwing” situation, for which the following types of relationships are used:

A corresponds to an animated initiator of the action;
O corresponds to the object of action;
I corresponds to an instrument of action;
L corresponds to a place of action;
E corresponds to an element of the set;
U is a subset of a set;
H corresponds to a "made of" characteristic.

There are simple semantic networks (their vertices do not have their own structure) and hierarchical networks (their vertices have some structure). The difference between hierarchical semantic networks is the possibility of dividing the network into subnets (subspaces) and establishing relationships not only between the vertices, but between the subspaces as well.

Semantic networks are convenient to use in systems of understanding NL-texts and have found application, in natural language processing systems, in question-answer systems, as well as in systems of artificial vision. The latest semantic networks are used to store knowledge of the structure, form, and properties of physical objects. In the field of natural language processing by means of semantic networks, special emphasis is made on: semantic knowledge, knowledge of the world, episodic knowledge (that is, knowledge of space-time events and states). The main advantage of this model is in accordance with modern ideas about the organization of long-term human memory. The disadvantage of the model is the complexity of finding the output on the semantic network [8].

V. Web technologies and semantic search in SEO and LSI texts

There are many interesting possibilities of using NLL systems with web technology. There are three main types of web-based NL systems:

1) *NL search information systems* (Retrieval systems);
2) *NL information extraction systems* (Information Extrieval systems);
3) *NL understanding systems* (Text / Message Understanding systems).

IR type systems provide document search by information request in documentary data bases.

Systems of type IE, unlike IR systems, allow not only to find relevant documents, but also to extract units of information that meet the information needs of users.

In systems like TMU, the understanding of text means:

1) understanding a text means to translate it into another language so that it is correctly interpreted by external "observers";
2) a text is understood correctly if the answers to the questions on it are evaluated by external “observers” as correct.

The problem of extracting information over the data (if you have access to the database scheme) has been solved (for small PDOs) at a practically applicable level.

Latent semantic indexing (LSI) is an indexing method, enabling the Yandex and Google search robots to pay attention to the general meaning of the text as a whole, and not only to the uniqueness and richness of key words. LSI copywriting is writing texts based on hidden semantic indexing technology. That is, such texts in which it is important not the presence of the keyword, but the content.

The difference between traditional SEO copywriting and LSI copywriting consists on:

SEO copywriting

- Writing text based on the list of keywords (with their obligatory entry into tags, headings, first paragraphs).
- Work with keyword density in the text.
- Work with technical uniqueness.

LSI Copywriting

- Writing text based on a list of keywords (with an emphasis on meaning, and not on the entry of these keys).
- Adding words related to basic queries on the text.
- Working out the utility and semantic uniqueness of the material.

Stages of LSI development

Latent-semantic analysis was patented in 1988. Search engines have begun to implement LSI since 2013.

In 2011, Google began to use the Panda search algorithm. His goal is to deal with poor quality texts. Panda assesses the user interaction with the site as well as the level of human involvement in the study of a particular page.

When forming the opinion of the person about the page, the main role is played by the text. Therefore, with the introduction of the Panda, hanging out process got rid of a heap of sites with poor-quality content - publications that were created not for people, but for entering keywords.

In 2013, Google launched the Hummingbird algorithm. Search queries were processed not only by keys, but also by meaning. In 2016, the search process was improved, a RankBrain ranking signal was introduced, a kind of artificial intelligence that should be able to understand not the whole meaning of each word or even the general content, but the essence of the whole phrase.

The innovation is connected with the fact that people began to enter concise inquiries with keys into the search bar, and also began to use "natural" queries - phrases from colloquial speech, sometimes long and complex.

A similar situation is observed in Yandex. In November 2016, Yandex launched the Palekh algorithm. As a result, the interest in writing LSI texts has increased dramatically.

Search engines increasingly pay attention to content, rather than technical indicators (presence of keys, number of entries, etc.). Now it's time to talk not only about the "relevance of benchmark words", but also about the "relevance of meanings". This is the latest trend in semantic search [9].

VI. Graphs

When translating from English into Russian at the first stage of the sentence analysis, special attention is paid to the problem of polysemy / homonymy of lexical units in the sentence. In English, a set of words can have one meaning, but belong to different parts of speech like "to go" which is a verb and "to have a go" is a noun, or be expressed with one part of speech, but can have many unrelated meanings like, for example, the verb "to draw. For this reason, the problem of determining to which part of the speech each lexeme belongs in the sentence [1] is solved.

Each lexeme gets a set of parts of speech that the given word can match in the sentence. Every possible sentence, i.e. a combination of parts of speech, can be regarded as a path in a graph whose vertices are a complete set of possible parts of speech corresponding to a given word [2]. Each possible part of speech corresponding to the i-th word is connected by an edge with every possible part of speech corresponding to (i + 1) -th word. Using the statistical data of linguistic environment, for example, a noun is less likely to join other nouns, if they do not perform the function of an adjective, which in turn must be reflected in the dictionary, you can assign a certain weight to each part of speech corresponding to the word. The cheapest path through this graph will be the best interpretation of the sentence in terms of determining its words belonging to parts of speech.

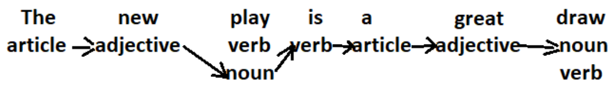


Fig. 5. Determining parts of speech by using a weighted graph

The determining of parts of speech of the sentence is the first step to interpret it and to find the subordination tree, i.e. hierarchical structure of the sentence.

Several linguistic schools developed several completely different ways of describing the syntactic structure of a sentence. We are interested in two representations of the syntactic structure that can be expressed by means of graphs: a) a description of the sentence structure through the system of its word combinations, i.e. through the relations of subordination and b) a description of the sentence structure through a hierarchy of its immediate components.

The description of the sentence structure by using the subordination tree means dealing with the word-combination. By word-combination we mean a syntactically related group of sentence words consisting of two words, and its components can be separated in the sentence by other words. In the word-combination we distinguish the main and a subordinate word, thus setting the subordination tree. The subordination tree can be considered as an adequate representation of the syntactic structure of the sentence, if the sentence doesn't have any formal or semantic links not reflected in the tree. Such a representation is not always met and is called an ideal tree [3]. If the subordination tree is not considered to be ideally hierarchical, connections and dependencies not reflected in the tree must be specified separately. The representation of the syntactic structure of a sentence in the form of a subordination tree is used in building up generating grammars and in algorithms for syntactic analysis. It is convenient that the tree graph reflects not only the relation of subordination, but also the order relation corresponding to the linear arrangement of words in the sentence, which are connected in a certain way. This connection is called projectivity. Properties of the subordination tree: no two branches intersect each other and no branch derived from some point *a* intersects perpendiculars dropped from the top points.

As an example of a subordination tree, let's take the previous example with already determined parts of speech.

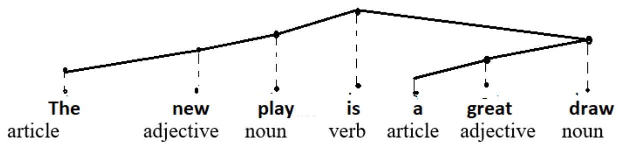


Fig.6. A subordination tree

The syntactic position of the identified parts of speech will allow us to build up a subordination tree, depending on, for example, the position of the verb with respect to names or nominal groups, as in our case, where one nominal group consisting of an article, an adjective and a noun stands before a verb, and the other comprised of similar components is placed after a verb. In accordance with the linear structure of the sentence, the first group should be considered as a subject, and the second one as the predicate of the sentence.

The concept of a component is the basic concept of descriptive linguistics. The concept of a component is closely related to the notion of immediate components. Components of a sentence are joined from smaller groups to larger ones. Connection of the subject group and the predicate group is the last stage of building a sentence. The sentence is the maximum component, and the word is the minimal one. The group of the subject and the group of the predicate are immediate components.

If these groups consist of more than one word, it means that they also include their immediate components. The immediate components are similar in their properties to strongly connected components of a graph. Although strongly connected components correspond to cycles, and there are no cycles in the tree, nevertheless, they have one common feature: if an element of one component (for example, a word) is included into another component, then one of them is completely integrated in the other one. It turns out that in the tree of components, although there is a one-way connection from a subordinating node to a subordinated one, there is also an inverse informal semantic connection that limits the compatibility of the dependent element and the determining one. It is most convenient to represent the tree of components in the form of a binary system. The binary tree provides a quick search for nodes, easy tree traversal, simple insertion and removal of an element.

Now it's important to make a transition from the subordination tree to the binary components tree. In order to pass from each sentence subordination tree on to its component system, it is necessary for each branch in the bushes with a fork to be assigned a certain number (weight), expressing the degree of proximity of this dependent element with respect to the determining element. If the branch with the number *n* is included in this segment, then all branches with weights less or equal to *n* must also be included in this segment. Assignment of numerical values to branches can be carried out, taking into account the belonging of each lexeme to a certain part of speech and its position in the sentence.

In our example, the verb-predicate of the sentence is the source vertex, and all the components belong to the same bush. Weighted edges are traced to the right and to the left from the axis coming from the top of the bush, which weights increment and no number is repeated twice, which makes it possible to obtain a binary system of components for a given tree. The following figure shows the binary tree of the components in their normal form.

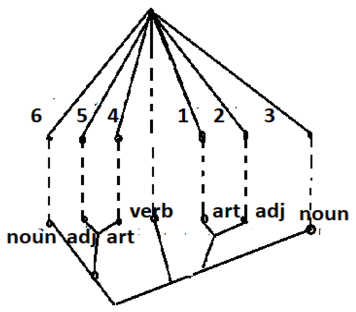


Fig. 7. Transition from a subordination tree with weighted edges to the component tree

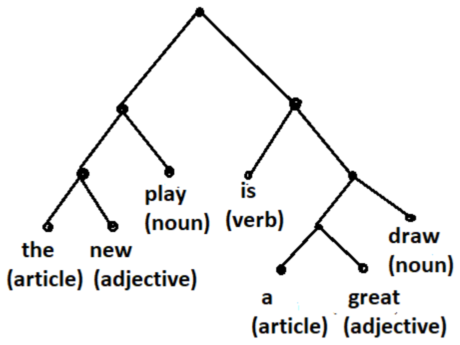


Fig. 8. A components tree

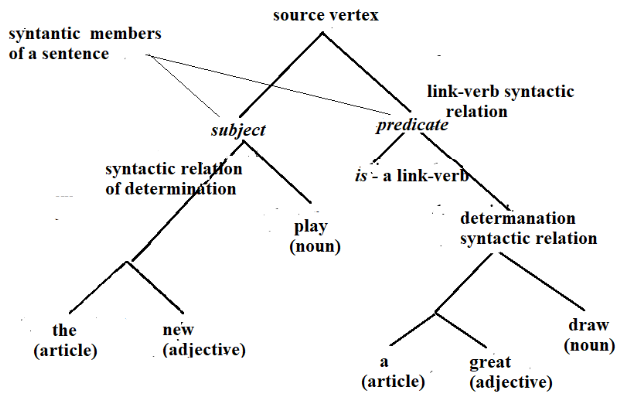

source vertex
syntantic members of a sentence
link-verb syntactic relation
subject
predicate
syntactic relation of determination
is - a link-verb
play (noun)
determanation syntactic relation
the (article)
new (adjective)
a (article)
great (adjective)
draw (noun)

Fig. 9. Building syntactic relations in a sentence

Sometimes, to optimize the graph, it's necessary to mark out the main component. The main component is usually allocated by double lines in higher-level component.

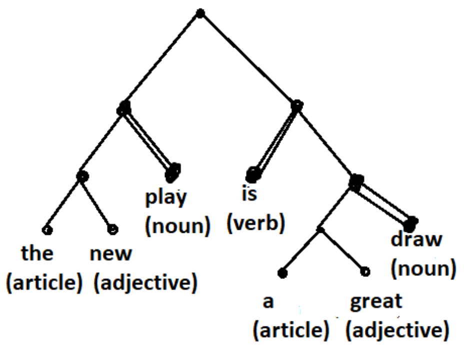


Fig. 10. Selecting main elements in the components tree

The main elements in the constituent parts of the sentence form a proposition: *play is draw*, which is not part of this sentence.

Representation of a sentence with a binary tree is good because in modern programming languages data is represented by linked lists that easily turn into binary trees and vice versa. Another advantage of the binary tree is a proven algorithm for comparing binary trees (a powerful tool for comparing the syntactic structures of two sentences).

Let's compare the sentence with an arithmetic expression. An arithmetic expression is represented by operators and numbers corresponding to the terminal elements. The problem is to represent the arithmetic expression in this way by leading it to such a form, so that it can be compared with another arithmetic expression based on a formal surface structure (without recourse to calculations).

At the first stage, we use the algorithm to express an expression using a binary tree. First, we give each operator a certain weight:

Algorithm 1 Assigning weights to sentence components

```
          for (int m=0;m<11;m++)
        {
          mas = mmm[i1, m];
          if (mas == "*")
             rlinq.AddLast(new point(mas, 4, s));
          else
             if (mas == "/")
                rlinq.AddLast(new point(mas, 3, s));
             else
                if (mas == "+")
                   rlinq.AddLast(new point(mas, 5, s));
                else
                   if (mas == "-")
                   rlinq.AddLast(new point(mas, 5, s));
                   else
                     if (mas!="")
                   rlinq.AddLast(new point(mas, 1, s));
          s++;
```

We define the operator with the greatest weight and its place in the expression. We split the expression into the left and right sides with respect to the operator with the maximum value. The terminal elements are assigned minimum values. We have a representation class consisting of a vertex (the maximal element) and a part of the expression corresponding to the left and right parts, if any, that are stored in a linked list that operates on these elements.

Algorithm 2 Building up a binary tree

```
    while (!stop)
        {
          sak = vvv.First;
          left.Clear();
          right.Clear();
          xxx.Clear();
          for (int i = 0; i < 20; i++)
             if (sak.Value.GetX(i) != null)
                xxx.AddLast(sak.Value.GetX(i));
          digit(xxx); f = 0;
          current = xxx.First;
          s = xxx.First.Value.GetNumber();
          xxx.RemoveFirst();
          node = xxx.First;
          f = node.Value.GetNumber();
          if (xxx.Count() == 1)
             left.AddLast(node.Value);
          else
             while (node != null)
             {
                if (f < s)
                   left.AddLast(node.Value);
                if (f > s)
                   right.AddLast(node.Value);
                f++; node = node.Next;
             }
          str = "";
          if (current.Value.GetSymbol() == "*")
             str = "Mult";
          if (current.Value.GetSymbol() == "/")
             str = "Div";
          if (current.Value.GetSymbol() == "+")
             str = "Add";
          if (current.Value.GetSymbol() == "-")
             str = "Subst";
 if
Char.IsDigit(Convert.ToChar(current.Value.GetSymbol()))
== true)
             str = "Numb";
          for (int i = 0; i < 5; i++)
          {
             if (matrix[0, i] == str)
             {
      str=str+ Convert.ToString(Convert.ToInt32(matrix[1,
i]) + 1);
          matrix[1,i]=
Convert.ToString(Convert.ToInt32(matrix[1, i]) + 1);
                break;
             }
          }
          if     (left.Count()     >     0     &&
str.IndexOf("Numb")==-1)
        vvv.AddLast(new              representation(str,
current.Value.GetSymbol(), left));
          if     (right.Count()     >     0     &&
str.IndexOf("Numb")==-1)
        vvv.AddLast(new              representation(str,
current.Value.GetSymbol(), right));
       ggg.AddLast(new          tree(sak.Value.GetPoint(),
sak.Value.GetSymbole(), str, current.Value.GetSymbol()));
          nedor.AddLast(new                  image(str,
current.Value.GetSymbol()));

          vvv.Remove(sak);

          if (vvv.Count() == 0)
             break;
        }
```

Operators are replaced by vertices with appropriate notation.

Links between vertices are displayed using contiguity lists. For greater clarity, expressions 1) 2*3+4*3-4/5; 2) 2*3+4/3+4/5; 3) 3*4+3*4-5/3; 4) 2+3+4+3; are given shapes as follows:

1)
Mult1 Add Subst1
Numb1 Mult1 Numb2
Mult2 Subst1 Div1
Numb3 Mult2 Numb4
Numb5 Div1 Numb6
2)
Mult1 Add Add1
Numb1 Mult1 Numb2
Div1 Add1 Div2
Numb3 Div1 Numb4
Numb5 Div2 Numb6
3)
Mult1 Add Subst1
Numb1 Mult1 Numb2
Mult2 Subst1 Div1
Numb3 Mult2 Numb4
Numb5 Div1 Numb6
4)
Numb1 Add Add1
Numb2 Add1 Add2
Numb3 Add2 Numb4

Even if the operators coincide or the leaves of the graph coincide, each of them is assigned a separate index, depending on the order of appearance in the graph.

At the next stage, optimization is performed to more clearly visualize the tree to simplify comparison with the other binary graph.

There are two approaches. It seems that the search for a backbone tree using the Kruskal algorithm could be suitable for this task. This algorithm looks for the cheapest edges connecting the connected components of the graph, i.e. cycles. The algorithm operates on weighted graphs, determining in the first stage whether both endpoints of the candidate edge are in the same connected component. In the case of a positive test result, such an edge is discarded, because by adding it it would create a loop in the future tree. If the endpoints are in different components, the edge receives and connects the two components into one. The structure of data storage is important. It is advisable to first group the vertices into connected components, and then search for the cheapest edges between the components.

In the arithmetic expression used as the model of the sentence, there are no cycles, therefore, there are no connected components, and there are only vertices with different degrees of their edges. In the approach to optimization, we consider the search for a vertex cover of a graph. The vertex cover does not coincide with the independent set, because a vertex cover can contain vertices belonging to one edge.

We get the following expression optimizations:

1) Mult1 Subst1 Mult2 Div1
2) Mult1 Add1 Div1 Div2
3) Mult1 Subst1 Mult2 Div1
4) Add1 Add Add2

As the method of comparing expressions, we use the remainder of sets intersection. The greatest coincidence is observed in the first and third expressions. The same approach can be used to compare the syntactic structure of sentences.

VII. Sets

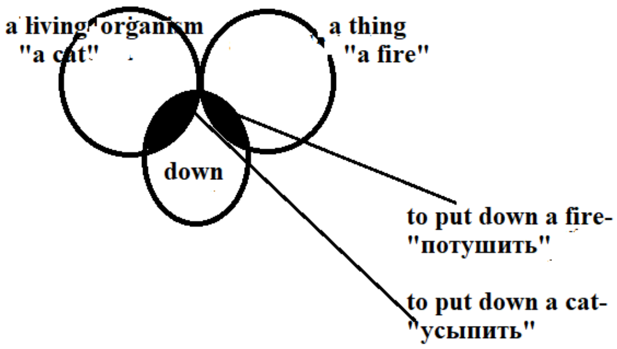


Fig. 11 Representation of a lexical construction by means of the intersection of sets

Turning to the proposition, one can figure out the meaning of the semantic construction as the intersection of the sets and its constituent elements. In semantics, this corresponds to the concept of intensional, in contrast to the extensional, where all elements of the set are considered. Let's compare the expressions: (keep ∩ down ∩ a living organism); (keep ∩ down∩ a material thing)
To check the definition of this expression as a whole, we can use the union of intersections E=(keep ∩ down ∩ a living organism)U(keep ∩ down∩ a material thing)=(keep ∩ down∩(a thing U a living organism)=(keep ∩ down ∩ u), where (a material thing U a living organism)=**u**, because if *a material thing*=a , then a person= a^c, a a U a^c=**u.** Hence, one concept corresponds to the set, and a broader one - the generic concept corresponds to the complement of the set, that is, one can figure out this union of intersections as follows:

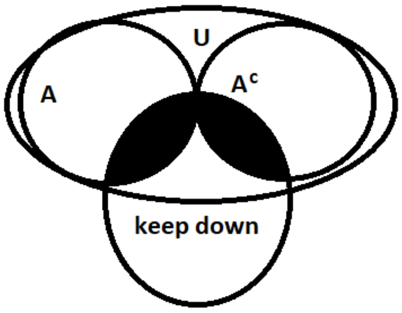


Fig.12 Intersection of the set and its complement

An English expression can be expressed in Russian by different verbs, although with similar features. The representation of the values of lexical constructions using the intersection of sets seems to be the right approach to formalize the meaning of an expression by means of variables connected with the help of a union, intersection, and addition operations. When comparing our original sentence with a few examples, if each of them contains all the variables appearing in the original expression, then there is no need to do transformations, and if there are some variables that are not in the original one, it will be necessary to bring the original expression to full normal form in order to size it up with other ones.

There is an expression “put it away”. Let’s represent it as an implicant: (put ∩ away ∩ it).

The explanatory dictionary contains the following implicants: (put ∩away ∩ something) = 1) put something into the proper place; 2) ship-move away; (put ∩ away ∩ a person)=1) get rid of someone; 2) put someone in prison or asylum; (put ∩away ∩some ideas)=leave behind;

Therefore, it is necessary to bring the variable *it* to an accordance with the implicants in the dictionary. Using it as an abstract variable that does not imply an *it - something* relation, we must embrace all the elements of a given set. Let’s represent this variable as an implicant ($C \cup C^C$), i.e. (a thing U a person), thus we obtain the product E = (put ∩ away)∩(a thing U a person)=(put ∩ away ∩ a thing) U (put ∩ away ∩ a person), is to say, this work covers all vocabulary meanings, which is good. There are no exceptions left uncovered. *It* is usually a demonstrative pronoun indicating an inanimate object, an animal, an event or fact. Thus, our expression is E = (put ∩ away ∩ a thing); will match the expression E_1 = (put ∩ away ∩ something).

Conclusion

When analyzing a sentence, two methods can be applied: 1) building the component tree for a general representation of the syntactic structure of the sentence and optimizing the resulting tree to minimize its elements; 2) representing the lexical structure using intersections; telling the difference of one expression from another. After checking the completeness of the set, a correspondence is established between the variable of the initial expression and the variable of dictionary expressions.

That in general will have a beneficial impact on the scientific background with further improvement, modernization and optimization of the text information processing subsystem.

References

[1] Sak A.N. Identificacia chlenov predlojenia kak osnovnaia zadacha mashinnogo perevoda. [Identification of the sentence members as the main task of machine translation]. Moscow .: Scientific Review Ed.14, 2015 (in Russian).

[2] Steven S. Skiena Algoritmy. Rucovodstvo po razrabotke[The algorithm Design Manual]. St.Petersburg, BHV, 2011(in Russian).

[3] Paducheva E. V. Statyi raznyh let [Articles of different periods]. Moscow, 2009, Studia philologica, 20p.

[4] Alyoshintsev A. V., Bessonova E.V., Sak A. N. Modelirovanie systemy machinnogo perevoda technicheskih tekstov v telecommunicacionnoy sfere s ispolzovaniem objectno-orientirovannogo yazika programmirovania C# [Modeling of the machine translation system for technical texts of the telecommunication field using the object-oriented programming language C #] T-Comm, Telecommunications and transport 2017.Vol 11. №10. pp. 66-73(in Russian)

[5] Russel Stuart Iskustvennyi intellect. Sovremennyi podhod.[Artificial intelligence. Modern approach]. Moscow/St. Petersburg/Kiev, 2006

[6] Alyoshintsev A. V., Jolodkov P.A. Obzor sovremennij tejnologiy ispolzuemij pri postroenii zdaniy [Overview of contemporary technologies used for smart-house design] T-Comm, Telecommunications and transport 2009. № S2. pp. 159-163.

[7] Rybina G.V. Osnovy postroenya intellectualnih system. [Basics of building intelligent systems]. - Moscow : Finance and Statistics; INFRA-M, 2010.-432pp.

[8] Malysheva E.N. Ekspertnuye systemi.[Expert systems]: Studies support– Kemerovo: Kemerov. state University of Culture and Arts, 2010. - 86 p.

[9] Shamina I.S. SEO copyrighting 2.0 Kak napisat’ teksti v eru semanticheskogo poiska. [SEO-copywriting 2.0 How to write texts in the era of semantic search]. - M.: Infra-Engineering, 2018. - 260 p.

[10] Alyoshintsev A. V., Sak A. N. Modelirovaniye informacionnih system computernogo zreniya dlia machinnogo perevoda metodamy teorii grafov. [Modeling computer vision information systems for machine translation using graph theory methods] // T-Comm: Telecommunications and Transport, Vol. 12, No. 10, 2018, P. 58-63

[11] Bibilo P. N.Logicheskoye proektirovaniye discretnih usrtroystv s ispol’zovaniem productionno-freimovoy modeli predstavleniya znanii. [Logical design of discrete devices using the production-and-frame model of knowledge representation / P.N. Bibilo, V.I. Romanov. - Minsk: Belarus. Navuka, 2011. - 279 p.