Specifics of construction industry terminology and methods of analysis of the natural language sentences similarity by using isomorphism of their structures

Alexander Sak
Moscow State University of Civil Engineering, Yaroslavskoe shosse, 26, Moscow, 129337, Russia
E-mail: sakan@mgsu.ru

Abstract This article discusses methods of automatic translation by analogy. Each dictionary entry in the online dictionary contains a certain number of examples highlighting the use of this lexeme in different contexts. The purpose of this study is to find an example from an online dictionary that most fully conveys the meaning of a word in a particular context. Based on the fact that natural language sentences can be represented as graphs, the author considers several methods based on set theory, graph theory and linear algebra to compare sentences in order to find the closest example in meaning. In addition to purely formal comparison methods based on syntax, the author presents methods for analyzing sentences based on the semantics of their components. The balance between syntactic and semantic structure when comparing sentences gives the best result for finding the correct meaning of the predicate, which ensures an adequate transfer of the idea of the sentence during translation.

1. Introduction.
In the construction field, document handling is getting increasingly important. Many documents are presented in foreign languages. There is an urgent need for a machine translator focused specifically on this area. Sentences consist of discrete units and can naturally be represented by graphs, the vertices of which consist of semantic units, united by syntactic links - the edges of the graph. This work is based on the principles of discrete mathematics, namely set theory and graph theory, which provide wide opportunities for semantic and syntactic analysis in translation [1]. Representation of the semantic structure of a sentence as a fundamental product allows us to build a multidimensional cube, the vertices of which are fundamental products corresponding to the lexemes of the sentence. To check the correctness of building a multidimensional cube, critical graphs are used [3]. Tree graphs formed from the original sentence as well as the online dictionary examples can be checked for isomorphism to find trees with similar connections between its vertices. To do this, one can use a comparison of the adjacent matrices diagonals of these graphs raised to a power depending on their diameter, which gives a formal analysis of these structures, and also use graph neural networks to analyze both syntactic structures and semantic features of graphs vertices or lexemes. These features can be obtained through the so-called graph kernels [5] – i.e. graphs kernels’ comparison by means of the frequency of their use.
2. Methods

2.1 Fundamental products

In our previous studies [1], [2], we represented the meaning of expressions, in particular modal verbs, by using set intersections, where each set is a collection of possible meanings of a given lexeme. For example, to put out a fire - \( E = A \cap B \cap C \), where \( A = \text{put} \); \( B = \text{out} \); \( C = \text{fire} \). Imagine that \( a \in A \), \( b \in B \) and \( R \) stands for the “is_Part_of” generic relation, which is often used in semantic networks. If \( a \) and \( b \) are types of some generic concept, then they are connected by the relation \( R \) and, accordingly, \((a,b) \in R \), then \((b,a) \in R\), i.e. they are symmetrical and retain the same sense of expression in both sentences. For example, a candle \( R \) a lantern, where the generic word is source of light. The presence of the same feature in the original lexeme and the example lexeme gives the complement of the set \( B^C \). Thus, we can come up with the predicate by using the full normal form of union of intersections.

A multidimensional cube can be used to embed vocabulary examples into a system to determine how far they are from the original sentence. When designing finite state machines, when transmitting data, when constructing optimal block codes, graphs with a special structure are used. Such graphs are subgraphs of a graph called an n-dimensional cube. The graph of an n-dimensional cube can be defined differently: the vertices are all different sequences of zeros and ones of length \( n \), and two vertices are connected by an edge if the corresponding sets differ in exactly one position (in one bit). Often, instead of digits consisting of zeros and ones, the vertices of an n-dimensional cube can be represented by fundamental products. This presentation is made to minimize the union of fundamental products.

![Figure 1. Organizing fundamental products by using a graph](image)

We can assign the literal \( A \) to a direct object, \( B \) to a preposition and \( C \) to an indirect object. For example, for the following sentence *I read an interesting story in a thick book*, we can define the following product as \( E = \text{story} \cap \text{in} \cap \text{book} = A \cap B \cap C \). Looking through examples in the dictionary, we compare the corresponding elements, and if they match the literals of the original sentence, we assign them the value of the variable, if they do not match, we assign them the value of the complement of the variable. We compare nouns in terms of their class, and prepositions from the point of view of their coincidences. We obtain examples as follows:

- *I read you as a quiet guy* \( E_1 = A^C \cap B^C \cap C^C \);
- *I can read between the lines* \( E_2 = A^C \cap B^C \cap C \);
- *Please read this manuscript for spelling errors* \( E_3 = A \cap B^C \cap C^C \);
- *I read an interesting article in today’s newspaper* \( E_4 = A \cap B \cap C \);
- *I think I have read of you in the papers* \( E_5 = A^C \cap B \cap C \);
- We have \( E = (A \cap B \cap C) \cup (A^C \cap B^C \cap C^C) \cup (A^C \cap B^C \cap C) \cup (A \cap B^C \cap C^C) \cup (A \cap B \cap C) \cup (A^C \cap B \cap C) \); We see that the first and fifth fundamental products completely coincide - obviously the meaning of the input sentence can be conveyed by this particular product. Nevertheless, full
coincidence does not always happen, and in our case we delete the 5th piece and see how close the remaining pieces are to the original.

\[ E = (A \cap B \cap C) \cup (A^C \cap B^C \cap C^C) \cup (A^C \cap B^C \cap C) \cup (A \cap B^C \cap C) \cup (A \cap B \cap C); \]

The polynomial contains 5 fundamental products, so the graph will have 5 vertices. Since the polynomial contains \( n = 3 \) variables, then \( n + 1 = 4 \) groups are possible when splitting the vertices.

**Figure 2.** Finding implicants on a multidimensional cube

We have the following implicants: \((B \cap C) \cup (A^C \cap B^C \cap C) \cup (B^C \cap C) \cup (A^C \cap C)\). Thus, we observe a significant simplification of fundamental products, which leads to reduce the number of the necessary features for determining the word’ belonging to a certain class of lexemes. This means that in case of a large number of literals, you can resort to this optimization to find and further compare the implicants, which will allow you to compare sentences from the point of view of coincidence of features, not only on the basis of a quantitative, but also a qualitative indicator. For this, an appropriate hierarchy of features can be constructed, for example, the union of the sets \( B \cap C \) may be preferable to the implicant \( A^C \cap B^C \) when choosing a candidate.

2.2 Critical graph.

When creating an \( n \)-cube with vertices corresponding to fundamental products expressing sentences, you need to check its integrity, i.e. whether its vertices really represent the correct sequences of zeros and ones, and whether the relationships between examples from the dictionary are displayed using the edges of the cube. To check this out, depending on the size of the \( n \)-cube, one can use various bipartite graphs that cannot be embedded into the \( n \)-cube formed by the original sentence and sentences from dictionary examples. These cube-embedded graphs are called critical graphs. A critical graph is a graph that is not a subgraph of an \( n \)-cube, but by removing any edge from it makes it a subgraph of an \( n \)-cube, i.e. a cubed graph [3]. By removing the edge \( e = \{v_i, v_j\} \) we mean removing the pair \( \{v_i, v_j\} \) and the vertices remain. A catalog of critical graphs would allow checking the integrity of an \( n \)-cube with a sequence of different sizes. In our case, the minimal bipartite graph that is not embeddable into an \( n \)-cube is the graph \( K_{2,3} \).
In this case, if the edge $e = \{A \cap B \cap C, A^C \cap B^C \cap C^C\}$ is removed, this graph becomes a subgraph of the graph in Fig. 1, i.e. a hypercube. Obviously, there cannot be an edge between vertices, in which the sequences of zeros and ones differ in more than one position.

2.3 Isomorphism

Another principle for comparing the structure of sentences (their graphs obtained by using parsing) is to check for isomorphism of these graphs. For example, molecules with a similar structure are believed to have similar functional features. Of course, small tree graphs consisting of 3-4 vertices can be compared by brute force, i.e. by directly matching the vertices, especially if each of them is to occupy a certain position and can easily be matched to another vertex at a certain position. For example, *preposition - A, direct object - B, indirect object - C.* to say, it is checked whether for each vertex $v_i$ which in the first graph is adjacent to the vertices labeled with numbers $(v_i, v_j, \ldots v_k)$, there is a vertex in the second graph that is adjacent to the vertices with the same numbers. Other methods are used to determine whether large graphs are adjacent, including the adjacency matrices of these graphs and their exponentiation based on the diameter of the graph. It is known that the relationship between the vertices of the graph can be reflected in the adjacency matrix. Despite the development of data processing techniques based on the use of abstract data types, such as lists, stacks, or queues, matrices still remain a convenient and often indispensable tool for solving many graph problems. The adjacency matrix of vertices $A=[a_{ij}]$ of a simple labeled graph (labeled vertices) $G = (V, E)$ with $p$ vertices is a square matrix of order $p$.

$$a_{ij} = \begin{cases} 1, & \text{if the vertex } v_i \text{ is incident to the edge } e_j \\ 0, & \text{otherwise} \end{cases}$$

Graphs $G_1$ and $G_2$ with $p$ vertices and adjacency matrices $A_1$ and $A_2$, respectively, are isomorphic if and only if the set of diagonal elements of the $k^{th}$ degree of the matrix $A_1^k$, ordered in ascending order, coincides with the set of diagonal elements of the $k^{th}$ degree of the matrix $A_2^k$, also ordered ascending for any $k=d(G_1), \ldots, p-1$, where $d(G_1)$ is the diameter of the graph $G_1$, equal to the diameter $d(G_2)$. The proof of necessity is extremely simple, because if graphs are isomorphic, then all their parameters, including the number of walks, are the same. However, although the algorithm is heuristic, there are many advantages of using it. It should also be added here that this method of diagonal comparison is very convenient because one can compare matrices of different dimensions, and when comparing subgraphs, one can search for matches on the sections of the corresponding adjacency matrices diagonals.

3. Results

Let's take the graphs of three sentences and compose the corresponding adjacency matrices:
A She designs a very nice room.
B The Americans designed skyscrapers in 19th century.
C They design him as a builder

Let us reduce the given proposal to the adjacency matrices of dimension 5x5

| Table 1. Adjacency matrix of sentence A |
|-----------------------------------------|
| designs | a | very | nice | room |
| designs | 0 | 0 | 0 | 0 | 1 |
| a | 0 | 0 | 0 | 0 | 1 |
| very | 0 | 0 | 0 | 1 | 0 |
| nice | 0 | 0 | 0 | 0 | 1 |
| room | 1 | 1 | 1 | 1 | 0 |

| Table 2. Adjacency matrix of sentence B |
|-----------------------------------------|
| designed | skyscrapers | in | 19th | century |
| designed | 0 | 1 | 1 | 0 | 0 |
| skyscrapers | 1 | 0 | 1 | 0 | 0 |
| in | 1 | 1 | 0 | 0 | 1 |
| 19th | 0 | 0 | 0 | 0 | 1 |
| century | 0 | 0 | 1 | 1 | 0 |

| Table 3. Adjacency matrix of sentence C |
|-----------------------------------------|
| design | him | as | a | builder |
| design | 0 | 1 | 1 | 0 | 0 |
| him | 1 | 0 | 1 | 0 | 0 |
| as | 1 | 0 | 0 | 0 | 1 |
| a | 0 | 0 | 0 | 0 | 1 |
| builder | 0 | 0 | 1 | 1 | 0 |

| Table 4. Sentence A adjacency matrix square |
|---------------------------------------------|
| 0 0 0 0 1 | 1 1 1 1 0 |
| 0 0 0 0 1 | 1 1 1 1 0 |
| A= 0 0 0 1 0 | A^2 = 0 0 0 0 1 |
| 0 0 0 1 0 | 1 1 1 1 0 |
| 1 1 1 1 0 | 0 0 0 1 3 |

| Table 5. Sentence B adjacency matrix square |
|---------------------------------------------|
| 0 1 1 0 0 | 2 0 0 0 1 |
| 1 0 1 0 0 | 0 1 1 0 0 |
| B= 1 1 0 0 1 | B^2 = 0 1 2 1 0 |
| 0 0 0 1 0 | 0 0 1 1 0 |
| 0 0 1 1 0 | 1 0 0 0 2 |
Table 6. Sentence C adjacency matrix square

|     | 0 | 1 | 0 | 0 |
|-----|---|---|---|---|
| 0   | 2 | 0 | 1 | 0 |
| 1   | 0 | 1 | 0 | 0 |
| C   | 1 | 0 | 0 | 0 |
| C^2 | 0 | 1 | 2 | 1 |

The diameter of these graphs is 2, and therefore it is sufficient to raise the matrices to the second power. All squared diagonal elements of the adjacency matrices are equal for B and C, therefore these graphs are isomorphic, and the graph A is not isomorphic to any of them. Indeed, from a formal grammatical point of view, sentences B and C are very similar, as indicated by the correspondences skyscrapers-him; in 19th century-as a builder; design in-design as. From the point of view of the meaning of the sentence, when conveying the meaning, the key role is played by lexemes which determine the semantic content of the graph vertices, the coincidence of prepositions and the belonging of nouns at a certain position to the same generic group. Therefore, comparison based on fundamental products by embedding them into n-cubes seems to be a more justified method from the point of view of conveying sentence meaning. Nevertheless, analysis methods based on the synthesis of linear algebra and graph theory are improving and this has led to the emergence of a new direction in artificial intelligence, known as the Graph Convolutional Network [4].

We return to our adjacency matrices A, B, and C. We also need the features of each vertex in the graph. We have five vertices, so we create two integer features each based on their index.

```matlab
for i=1:5
    X(i,1)=i-1; X(i,2)=i+1;
end
```

Multiplying, A by X in this form denotes the sum of the features of their neighbors. This means that the convolutional layer of the graph represents each layer as a collection of its environment. It turns out that the representation of the vertex does not include its own features. A representation is a collection of features of neighboring vertices, i.e. only vertices with a loop include their own features in this set. Vertices with a large value of degrees will have a greater value in the presentation of their features, and vice versa, the lower the degree at the vertex, the less the value of its features. This can lead to fading or explosive growth of gradients. It can also create problems when using stochastic descent algorithms, which are used to train networks and are sensitive to the range of values of each input feature. Thus, it is necessary to add the identity matrix I to the adjacency matrix.

```matlab
for i=1:5
    A(i,i)=1;
    B(i,i)=1;
    C(i,i)=1;
end
```

The feature representation can be normalized by the vertex degree by multiplying the adjacency matrix by the inverse degree matrix D. First, we find the degree matrix for each adjacency matrix:

```matlab
for i=1:5
    s=0;
    s1=0;
    s2=0;
    for j=1:5
        s=s+A(i,j);
        s1=s1+B(i,j);
    end
end
```

The feature representation can be normalized by the vertex degree by multiplying the adjacency matrix by the inverse degree matrix D. First, we find the degree matrix for each adjacency matrix:
Thus, the simplified propagation rule will be as follows: \( f(X, A) = D^{-1}AX \)
Matrix \( D^{-1}A \) looks as follows:

\[
\begin{bmatrix}
0.5, 0, 0, 0, 0.5 \\
0, 0.5, 0, 0, 0.5 \\
0, 0, 0, 0.5, 0.5 \\
0.2, 0.2, 0.2, 0.2, 0.2 \\
\end{bmatrix}
\]

Here, in each row, the values (weights) of each row of the adjacency matrix are divided by the degree of the vertex corresponding to the row. We apply the propagation rule to the transformed adjacency matrix. First of all, it is necessary to apply weights to the vertices: \( W = [1, -1; -1,1] \); In this case, the weights are applied arbitrarily.

\( D\_hat=inv(D); \)
\( D1\_hat=inv(D1); \)
\( D2\_hat=inv(D2); \)

And we apply the ReLu activation function to each product:

\[
\text{relu}(D\_hat*A*X*W) \\
\text{relu}(D1\_hat*B*X*W) \\
\text{relu}(D2\_hat*C*X*W)
\]

Имеем результат:

We obtain the result as follows:

| Table 7. Feature representations of A,B and C |
|---------------------------------------------|
| | [4 0] | [2 0] | [2 0] |
| [5 0] | [2 0] | [2 0] |
| A= | [5 0] | B= | [3.5 0] |
| [7 0] | [7 0] | C= | [4 0] |
| [4 0] | [6 0] | [6 0] |

To reduce the dimension of the output representation of features, you can reduce the dimension of the weight matrix \( W = [1; -1] \);
We obtain the result as follows:
Table 8. Minimized feature representations of A, B and C

|     | 4  | 2  | 2  |
|-----|----|----|----|
| A   | 5  | 2  | 2  |
| B   | 3.5| 2  | 2  |
| C   | 4  | 6  | 6  |

Thus, we obtained a hidden layer using an adjacency matrix, input features, weights and an activation function.

It can be concluded that with the same weights at the vertices of the three graphs, the output representation of features for matrices B and C is practically the same. Let’s leave the value of the features of vertices for A and B the same, and for C, we slightly change the values of features for 2 and 4 vertices.

\[
X_1 = [1, -1, 3, -3; 3, -3, -5, -5; 0, 0]; \\
X_2 = [1, -1, 3, -3; 3, -3, -5, -5; 0, 0]; \\
X_3 = [1, -1, 2, -2; 3, -3; -4, -4; 0, 0];
\]

We obtain the result as follows:

Table 9. Changing features in vertices of A, B and C

|     | 1   | [4,66] | [4] |
|-----|-----|--------|-----|
| A   | 3   | [4,66] | [4] |
| B   | 3.5 | [4,66] | [4] |
| C   | 2.6 | [4,66] | [4] |

We see that, despite the changes in the features of the two vertices in C, the values of the elements of C are generally closer to B than the relationship between A and B. This is explained by the similarity of the adjacency matrices for B and C, as well as the insignificant difference in features between the non-coinciding vertices.

And with the next set of features

\[
X_1 = [1, -1; 3, -3; 3, -3; -5, -5; 0, 0]; \\
X_2 = [1, -1; 3, -3; 3, -3; -5, -5; 0, 0]; \\
X_3 = [1, -1; 6, -6; 3, -3; -7, -7; 3, -3];
\]
We obtain the result as follows:

|       |       |       |
|-------|-------|-------|
|       | [1]   |       |
|       |       |       |
| [3]   |       |       |
| [4,66]| [6.66]|       |
| A=    | [3]   |       |
|       | [6.66]|       |
| B=    | [3.5] |       |
|       |       | [4.66]|
| C=    | [0]   | [3]   |
|       | [0]   |       |
| [2.8] | [2]   | [4]   |

In this case, despite the similarity of syntactic characteristics (adjacency matrices) in A and B, sentences A and B are semantically much closer to each other and, accordingly, the meaning of a verb from sentence A will be chosen to translate sentence B.

We’ll show what a practical application of this method looks like.

Obviously, the translations by Google translate holds sentences A and B much closer in meaning than B and C despite the similarity of their structures. Though in the third sentence Google makes a mistake when translating the verb. Given the context in the first sentence Google chooses the verb “оформлять”, which is not bad though in this context it would be better to use “проектировать”.

She designs a very large room.
The Americans designed skyscrapers in 19 century.
They design their son as a builder.

Google translate:
Она оформляет очень большую комнату.
Американцы проектировали небоскребы еще в 19 веке.
Они конструируют сына строителем.

Yandex translate uses one verb to handle all three sentences which is not good:
Yandex translate’s version:
Она проектирует очень большую комнату.
Американцы проектировали небоскребы в 19 веке.
Они проектируют своего сына как строителя

Our system’s translation:
Она проектирует большую комнату.
Американцы проектировали небоскребы в 19ом веке.
Они растят своего сына строителем.

If in the first two sentences our translation results coincide, in the third one our variant shows a better result

3.1 Graph kernels
An important question that still remains is to solve the problem of combining nouns and adjectives and their translation. If (a,b) ∈ R, then (b,a) ∈ R. In mathematics, this kind of connection is called “relations”. The relation R on the sets A and B is transitive if for each pairs (a,b) ∈ R и (b,c) ∈ R, that is, there is aRc relation. A relation R is not transitive if there are elements a, b,c ∈ A, such that (a,b) ∈ R и (b,c) ∈ R, but a pair (a, c) ∉ R.

If G is a labeled graph with adjacency matrix A, then the element \( a_{ij}^{(n)} \) of the matrix \( A^{n} \) where \( A^{n}=A\cdot A\ldots A \) is equal to the number of paths of length \( n \) from the vertex \( v_{i} \) to the vertex \( v_{j} \) [2]. It is known that the more paths connect two graph vertices, the more similar characteristics these...
vertices have to each other. This does not work if the graph contains cycles. But there are no cycles
in trees. For example, "a lightweight structure". Let’s denote a set of semantic features in the word
“lightweight”: a weight in boxing and other sports intermediate between featherweight and
welterweight; a person or thing that is lightly built or constructed; of thin material or build and
weighing less than average; containing little serious matter.

“Structure” has meanings as follows: the manner in which something is built; something that is
built; a building.

At the next stage of the analysis, we add 2 sets of Russian values. In total, we have a set A
(meanings of an English adjective), a set B (meanings of an English noun), a set C (options for
translating an English adjective), sets D (options for translating an English noun). As a shift, we use
the connection between English and Russian nouns. To establish a connection between Russian nouns
and adjectives, we can indicate whether there are examples of their compatibility from the explanatory
dictionary (for this, we look at the meanings of both the noun and the adjective), if there is a match, we
denote the connection as = 1, for a more thorough analysis when working with large corpora of texts,
we can calculate frequency of use and assign a certain probability of compatibility as weight. For very
long paths between vertices, a special function can be used to reduce the importance of such paths.
The length k can be discounted by multiplying it by \(\lambda^k/k!\). 0\(\leq\lambda\leq1\) at 0\(\leq\lambda\leq1\), and the similarity is
determined by the formula

\[
k(i,j) = \left[ \sum_{k} \left( \frac{\lambda^k}{k!} \right) A^k \right]_{ij} = [\exp(\lambda A)]_{ij}.
\]

Our paths between the vertices are not that long, and we can just count the paths between them. We have raised to second power the binary matrix using conventional matrix multiplication. If the matrix A represents the full set of all 1-paths between the vertices \(v_i\) and \(v_j\), where \(A_{ij}=0\), if such a path does not exist and \(A_{ij}=1\), if such a path exists. When raising to second power the adjacency matrix if \(A_{ij}^2=1\) if and only if there is a number k such that \(A_{ik}A_{kj}=1\) or, in other words, there is an edge from the vertex \(v_i\) to the vertex \(v_k\) and from the vertex \(v_k\) to the vertex \(v_j\). Therefore, there is a path of length 2 or 2-path from the vertex \(v_i\) to the vertex \(v_j\).

We use the usual matrix multiplication, and we can make sure that \(A_i^2\) is equal to the number of k
values such that \(A_{ik}\) and \(A_{kj}\) are both equal to 1, that’s why it is equal to the number of paths from the vertex \(v_i\) to the vertex \(v_j\) of length 2. When the matrix is raised to third power, if the vertex \(v_i\) is connected by an edge with the vertex \(v_k\), and the vertex \(v_k\) has a common edge with the vertex \(v_j\), which is connected by an edge to the vertex \(v_j\), i.e. \(A_{ik}A_{ks}A_{sj}=1\), then there is a path from the vertex \(v_i\) to the vertex \(v_j\) through two vertices. In case of a conventional, not Boolean, rising of a matrix to third power, the corresponding value will correspond to the number of paths of length 3 from the vertex \(v_i\) to \(v_j\). The number of 3-paths between vertices indicates the strength of semantic links between lexical elements. Thus, all the weights of the paths are strengthened even more, clearly showing the relationship between the vertices. We have multiplied the Boolean matrix like a normal matrix, but you can match a normal and a Boolean matrix with each other. You can weigh the real numbers of the matrix by introducing some shift factors \(\mu\) and scale \(\sigma>0\), and use the value as approximation accuracy \(\Delta(S, \mu, \sigma)=\sum_{i,j=1}^{n}(aij - \mu - \sigma bij)^2\). But in fact, the representation of relations using real numbers is very convenient, due to the multidimensionality of connections between synonymous vertices. After raising this adjacency matrix to third power, the largest number of connections, and therefore, more weight are acquired by the phrases: легкая конструкция (221), легковесная конструкция (205), облегченная конструкция (150), легкое строение (120), легкая постройка (20), невесомая структура (10). Obviously, “a lightweight structure” can be translated by the word combination “легкая конструкция”. This one is the unique translation offered by Google translate and Yandex translate, though our system opts for “легковесная конструкция” because it is also very widely used one. Our system relies on a wider range of word combinations to choose from which allows to achieve the best result by context.
4. Conclusion
If formal structures of sentences are different, even the same set of vertex features does not allow to establish a superficial analogy between their graphs. Comparison of the adjacency matrices by formal features was shown above by comparing the diagonals of the adjacency matrices raised to second power. As we could see, this comparison is based only on the formal correspondence of the vertices and the paths between them. This is clearly not enough when it comes to semantics. The method of comparing fundamental products is based on the semantic meaning of the key elements of sentences. The disadvantage of this method is the impossibility of an intermediate value of the correspondence of a given vertex to a certain set. It means that the vertex $a \in (A \cap AC)$, i.e. either the vertex is included or is not included in the same set with the vertex $b$, making the analysis rather rough. Also, comparing the correspondences of vertices in two graphs is a rather laborious task. Analysis based on graph convolutional networks makes it possible to combine both syntactic relations between vertices (connections between vertices, expressed by adjacency matrices + degree matrices), and semantic relations (numerical values of the features of each vertex depending on their semantic meaning, as well as the values of weights). Given research put on display by means of two presented examples the advantage of offered technology of translating specialized text. Now the task is a mechanism for comparing lexical units and providing their features with numerical values so that they reflect how large or small the semantic distance is between them. One of the solutions to this problem is to find a correspondence between Russian and English phrases using the so-called graph kernels, i.e. search for paths between words of phrases and their possible translations due to the frequency of their use in online dictionaries.

References
[1] Sak A.N. 2019 Machine translation system modelling by means of sentences comparison.//Moscow: Journal of Physics: Conference Series Volume 1425 December 2019 https://iopscience.iop.org/article/10.1088/1742-6596/1425/1/012171/pdf
[2] Sak A.N. Modeling a natural language processing system by comparing sentences// Moscow: REDS: Telecommunication devices and systems, No. 3-2020, p.35 http://media-publisher.ru/wp-content/uploads/REDS-3-2020.pdf
[3] Kazansky A.A. Discrete mathematics.// Moscow: Publisher I.V. Balabanov, 2008.-208 pp
[4] Tobias Skovgaard Jepsen. How to do Deep Learning on Graphs with Graphs Convolutional Networks https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780
[5] S.V.N.Vishwanathan Graph Kernels vishy@ stat.purdue.edu