The Conceptual Modeling System Based on Metagraph Approach

N.D. Todosiev, V.I. Yankovskiy, Y.E. Gapanyuk, A.M. Andreev

Bauman Moscow State Technical University, Moscow, Russia

Abstract. The article is devoted to an approach to building a conceptual modeling system, which includes text recognition in a conceptual structure and text generation based on a conceptual structure. The metagraph is used as a conceptual structure. The architecture of the conceptual modeling system is proposed. The metagraph model is considered as a data model for conceptual modeling. The main ideas of the work of the text parsing module and text generation module are considered.

Keywords: *complex Graph Structures, Metagraph, Conceptual Compression, Text Parsing, Text Generation.*

DOI: 10.14357/20790279230120

Introduction

Humanity has accumulated a huge number of text documents. The task of extracting the meaning of a text from a large number of documents is difficult for the user and may require significant time costs.

The task becomes more complicated when the user is a decision-maker and must distinguish the meaning of incoming texts and make decisions in a limited time. One of the common ways to "conceptually compress" text information is to use conceptual models. Such models include mindmap diagrams, concept maps, and, in part, ontological models. Research is currently underway to develop conceptual models presented in complex graph structures, such as hypergraphs, hypernetworks, and metagraphs. The use of complex graph structures provides a significant degree of "conceptual compression". The use of conceptual models in the decision-making task involves the sequential implementation of three enlarged steps:

1. Synthesis of a conceptual model based on a text description.
2. Conceptual modeling, as a result of which new conceptual models are formed.
3. Analysis of the results of modeling, decision-making, and the formation of reports based on the decisions made.

In this regard, the development of methods and algorithms for the synthesis of conceptual models based on a text description, implementing step 1, and generation of text reports based on the models obtained as a result of conceptual modeling in step 3 are urgent tasks without solving which it is impossible to implement conceptual modeling fully.

This article discusses both the architecture of the conceptual modeling system and the principles of implementing the main modules of the system.

1. The Architecture of a Conceptual Modeling System

The Architecture of a Conceptual Modeling System is represented in the Fig. 1.

The system under development contains three large modules:

1. The text parsing module.
2. The text generation module.
3. The metagraph concepts modeling module.

The operation of the system consists of nine main steps:

1. In "Step I", a source text document is read.
2. In "Step II", "the text parsing module" parses the document, extracts concepts and relationships, and creates a metagraph structure.
3. In "Step III", the generated metagraph structure is recorded into "the metagraph concepts storage".
4. In "Step IV", "the metagraph concepts modeling module" receives the sourceconcepts for modeling from "the metagraph concepts storage".
5. In "Step V", the conceptual modeling is performed. The source concepts inthe form of metagraph are translated to the destination concepts.
6. In "Step VI", the results of conceptual modeling are recorded into "themetagraph concepts storage".

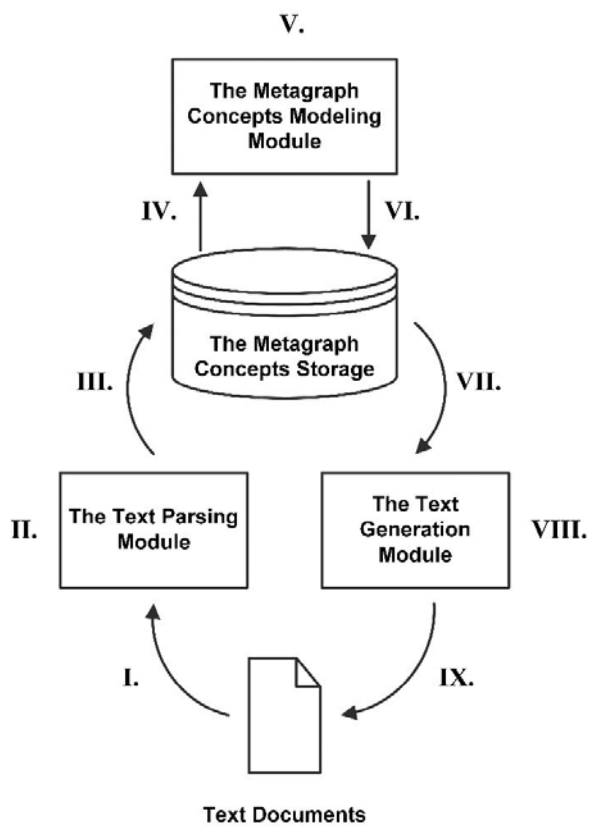

V.
The Metagraph Concepts Modeling Module
IV.
VI.
The Metagraph Concepts Storage
III.
VII.
II.
The Text Parsing Module
The Text Generation Module
VIII.
I.
IX.
Text Documents

Fig.1. The Architecture of a Conceptual Modeling System

7. In "Step VII", "the text generation module" receives the destination concepts from "the metagraph concepts storage".
8. In "Step VIII", "the text generation module" transforms destination concepts into text form.
9. In "Step IX," the output text document is generated.

The system architecture includes "the metagraph concepts storage". The detailed structure of the information stored in "the metagraph concepts storage" is the subject of a separate research. The main ideas of storing a metagraph model in relational, document-oriented, and graph databases are discussed in [15].

Various options for metagraph modeling are the subject of a separate study. In this article, we will consider in detail the principles of extracting metagraph structures from text and generating text based on the metagraph structure, which corresponds to modules "The text parsing module" and "The text generation module". We will also consider the basic principles that underlie "The metagraph concepts modeling module".

2. Using Metagraphs for Concepts Modeling

In this section, we discuss the use of a metagraph model as a data model for "The metagraph concepts modeling module". We will also compare flat concept maps (such as MindMaps and CMaps) with a metagraph model.

MindMaps. Probably, the most well-known approach to the representation of conceptual maps in practice is MindMap or "relationship diagram" [3].

The idea of constructing such a diagram is that the main topic ("topic") is depicted in the center of the sheet, in which hierarchical subtopics are nested. Various views can be used to visualize such a diagram. Hierarchically nested concepts can be displayed in the form of a tree, can be located on concentric circles, can be represented in the form of an Ishikawa diagram [6] (which is also known as a "cause-effect diagram"). The simplest software products support only one version of the representation of the relationship diagram (as a rule, this is the "classic" version, in which the concepts are located on concentric circles). More advanced software products (for example, XMind [1]) support several representation options and automatically convert a structure from one representation to another.

From the point of view of the data model, a relationship diagram is a flat graph whose vertices correspond to concepts and whose edges correspond to connections between concepts. Edges in this model are considered as non-directional and non-annotated (an edge cannot be assigned a label containing auxiliary data). It is the property of non-annotability that allows automatic transformations between views.

Advanced software products (for example, XMind) also allow you to create additional annotated edges (actually turning the link diagram model into a CMap model), but such edges are not subject to automatic transformations between views.

Concept maps (CMaps). The CMap approach was proposed by Professor Joseph D. Novak [10]. The CmapTools system is an automated tool developed based on Novak's theory, designed for the formalization of subject areas, and is also often used in practice as an automated learning tool.

From the point of view of the data model, a CMap (like a MindMap) is a flat graph whose vertices correspond to concepts, and whose edges correspond to connections between concepts. In contrast to the MindMap diagram, the edges in this model are considered as directed and annotated (an edge can be assigned a label containing auxiliary data). Unlike the MindMap diagram model, it is not possible to perform automatic transformations between views for the CMap model.

Advantages and Disadvantages of Flat Concept Maps. The main advantages of the existing approaches to the presentation of concept maps are:

1. The conceptual map is presented in a graphical

form, which allows the userto form a complete representation of the subject area.

2. This version of the presentation allows you to understand the hierarchicalrelationship between the concepts.
3. In the case of using a concept map as a training tool, it is possible to graduallyadd concepts and connections, which allows the student to form a holistic view of the subject area step by step.
4. The main disadvantages of the existing approaches to the presentation of concept maps are:
5. The MindMap diagram does not allow annotating edges, which seriouslylimits the expressive capabilities of the model. This problem is solved in the CMap approach.
6. Both the MindMap diagram and CMap use flat graphs to represent conceptual maps. This leads to the fact that a concept map with a size of even a few dozen concepts becomes almost unreadable. In particular, this happens due to a violation of Miller's law [13], since at the same time, the user is forced to work with a number of concepts that exceeds the number 7 ± 2.

To solve the second problem, the CMap approach uses the concept of nested nodes. But nested nodes allow you to combine ordinary nodes only once – hierarchical embedding of nested catches into each other is impossible in the CMap approach.

Thus, having analyzed the existing approaches to the representation of conceptual maps, we can conclude that the main problem of the existing approaches is using a flat graph as a model for the representation of a conceptual map. Next, we will consider the possibility of using complex graphs (metagraphs) as a model for conceptual maps.

The Metagraph Model for Concepts Modeling. The metagraph model is a kind of complex graph model. In this article, by metagraph we will understand the following: $MG = \langle V,MV,E \rangle$, where MG – metagraph; V – set of metagraph vertices; MV – set of metagraph metavertices; E – set of metagraph edges.

It should also be noted that in some versions of the metagraph model, there is such an element as a metaedge [14]. But in the proposed approach, metaedges are not used for conceptual modeling, so they are not considered in this article.

Metagraph vertex $v_i = \{atr_k\}, v_i \in V$, where atr_k – attribute. Metagraph edge $e_i = \langle v_S, v_E, \{atr_k\} \rangle, e_i \in E$, where v_S – source vertex (metavertex) of the edge; v_E – destination vertex (metavertex) of the edge; atr_k – attribute.

The metagraph fragment is defined as $MG_i = \{ev_j\}, ev_j \in (V \cup E \cup MV)$, where ev_j – an element that belongs to the union of vertices, edges and metavertices. The metagraph metavertex: $mv_i = \langle \{atr_k\}, MG_f \rangle, mv_i \in MV$, where mv_i – metagraph metavertex; atr_k – attribute, MG_f – metagraph fragment.

The main element of the metagraph model is the metavertex. From the general system theory point of view, metavertex is a special case of manifestation of emergence principle, which means that metavertex with its private attributes and connections became a whole that cannot be separated into its component parts. The example of metagraph representation is given in the Fig. 2.

The example contains three metavertices: mv_1, mv_2 and mv_3. Metavertex mv_1 contains vertices v_1, v_2, v_3 and connecting them edges e_1, e_2, e_3. Metavertex mv_2 contains vertices v_4, v_5 and connecting them edge e_6. Edges e_4, e_5 are examples of edges connecting vertices v_2–v_4 and v_3-v_5 are contained in different metavertices mv_1 and mv_2. Edge e_7 is an example of edge connecting metavertices mv_1 and mv_2. Edge e_8 is an example of edge connecting vertex v_2 and metavertex mv_2. Metavertex mv_3 contains metavertex mv_2, vertices v_2, v_3 and edge e_2 from metavertex mv_1 and also edges e_4,e_5,e_8 showing emergent nature of metagraph structure.

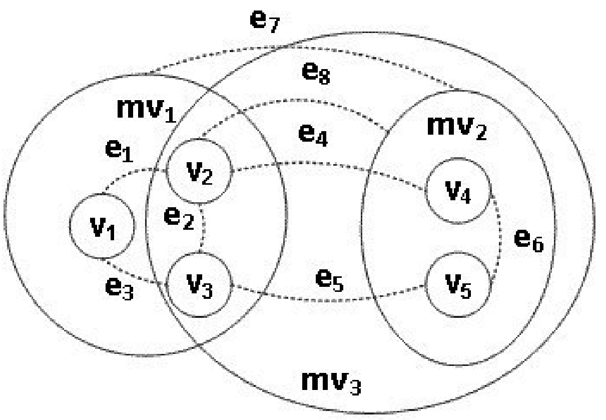


Fig.2. The example of metagraph representation

The metagraph model can be useful for describing conceptual maps. In this case, we can consider "simple" and "complex" concepts. At the same time, "simple" concepts are modeled using ordinary vertices, and "complex" ones are modeled using metavertices.

The use of metavertices to describe conceptual maps allows us to abandon the representation of a conceptual map in the form of a flat graph and switch to a holonic spatial description of a conceptual map in the form of a metagraph.

Using metagraph calculus [16], it is possible to carry out a formal transformation of the metagraph in the process of conceptual modeling.

The process of conceptual modeling is based on the sequential transformation of metagraphs using metagraph agents, which are described in [17]. As a result of modeling, an existing metagraph can either be modified or a new metagraph can be created.

In order to implement “the metagraph concepts storage”, the graph model can be transformed into a flat graph model as shown in [15]. Also, in accordance with [15], the database based on the flat graph model or document model or relational model can be used to store a metagraph model.

3. The Text Parsing Module

The module is based on the frame approach. For researching the frame semantic core transformation, the following frame concepts are considered:

– Zhabotinskaya’s frames[19]. The author of the article proposes frames of a simpler (lower) level, which are very close in meaning to syntactic constructions (cases, sentence members, etc.) Thus, it is not difficult to find a certain set of rules for converting syntax trees into Zhabotinskaya frames.
– FrameNet [2] database is a great tool providing many frame definitions, semantic roles, and word senses. Frames provide structure information. Frame instances provide data information. Entities in frame instances can be linked, showing that they are the same. Due to the nature of frames, entities inside some frame can also be frames. The resulting data representation is shown in the Fig. 3.
– FrameBank database is a thesaurus for Russian language [8]. Once there were just the Russian version of FrameNet [7], it were transformed and adapted for Russian, which has more morphological and semantic features than English.

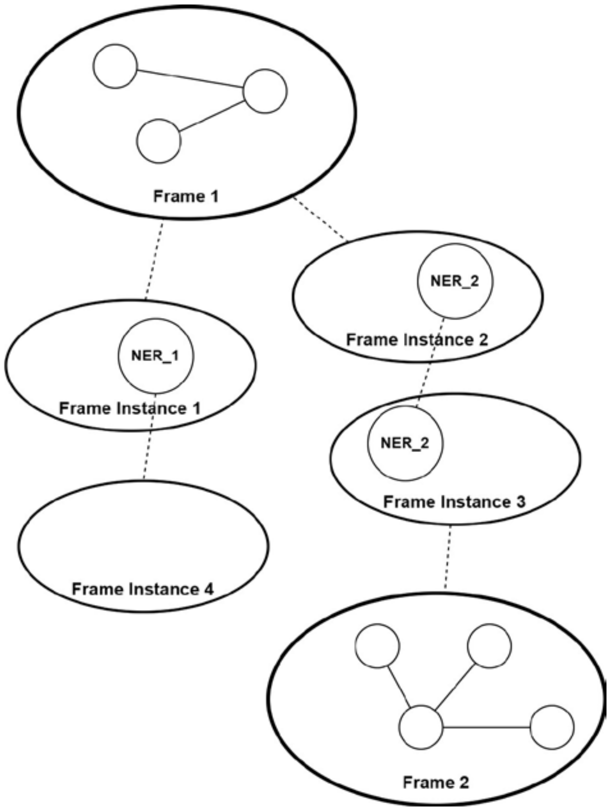


Fig.3. The Data Representation of the Frame Semantic Core

As a result, the FrameNet is considered more suitable because of its more flexible structure and acceptance worldwide. The Russian version of it will be used in further work. In this article, the English version is used.

The general idea of the graph generation algorithm is shown in the Fig. 4.

Initially, the source text is subjected to the resolution of reference links (coreference), which allows you to remove leaves that do not carry useful information, and also make the future graph more connected.

Then the text gets into the module for converting text to frames [12]. After that, the received frames are linked by target words. Thus, a graph is obtained, each node of which is a word. The frame attributes are turned into similar nodes. Links between the main node and the attribute node are marked with the appropriate tag (frame attribute).

Up to this point, we are working with a flat graph in which initially parsed low-level concepts correspond to words and phrases of the source text and relationships between these low-level concepts corresponds to the links both between the members of the sentence and between individual sentences.

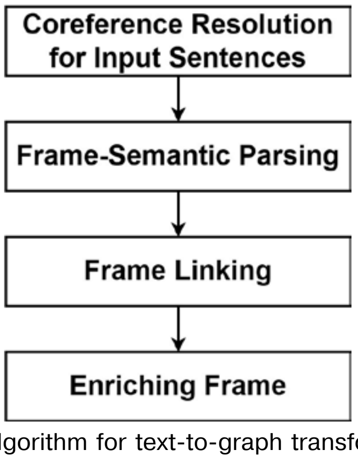


Fig.4. Algorithm for text-to-graph transformation

Further enrichment of frames leads to the formation of high-level concepts based on low-level concepts. But it is impossible within the framework of the flat graph structure, so we turn to the metagraph model.

Enriched frames can be considered as metavertices of a metagraph. This is due to the nested structure of the resulting frames – frames can be atomic or contain other frames. In this model, metavertices may be considered as compositions of low-level vertices.

It should be noted that items of "the metagraph concepts storage" only indirectly corresponds to sentences of the source text. The "the metagraph concepts storage" stores the high-level concepts of the metagraph model that were extracted and enriched based on the source text.

4. The Text Generation Module

The general idea of the text generation algorithm is shown in the Fig. 5.

First of all, the generating text system needs to know the purpose of generating text. This purpose will determine the path of the algorithm. The purpose of generating text is considered of these types of input information:

– User request.
– Meta information i.e., user language, user location, history of user request.

It should be noted that the purpose of generating text can be supplemented by another kind of information, such as another text input from the user explaining how the result text should be like.

This purpose is then analyzed for two main goals: find out which parts of the metagraph will be used to output the answer, and determine in which format output text will be generated. For simplistic purposes, it is supposed that analyzing input information is transforming this input into a semantic representation. That way, the purpose of generating text can be presented as semantic and manipulated as such.

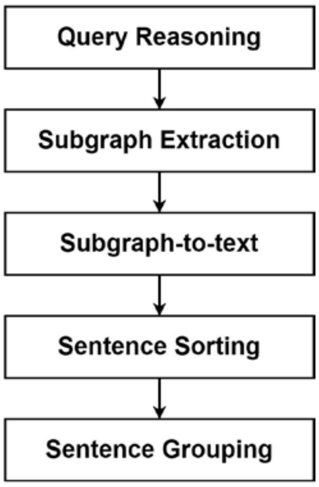


Fig.5. Algorithm for graph-to-text transformation

After analyzing the purpose of the generating text, the algorithm uses this semantic of purpose in semantic representation and highlights the individual parts of the metagraph.

These results are then compared with a graph to find parts of a metagraph. For the general case, it can be done in several approaches:

1. If supposed, that input frame and metagraph are using the same semanticcore, then this task becomes a subgraph isomorphism problem with NPcompleteness, which has too high computational costs. It has been solved recently [9], so this approach can be used for it;
2. In case when input frame and metagraph are not using same semantic core,then the graph comparison methods can be performed: the target frame is matched to search the relevant part of the metagraph;
3. Use unique approaches related to the features of the semantic core.

Highlighted parts of the metagraph will transform into text representation with syntactic and morphological transformations.

In the case of using frame semantic core, this algorithm became rather deterministic. Text of user query transform into semantic frames by the same algorithm, as described earlier, to transform the text into a semantic representation with one caveat: the interrogative words will be marked as "blank words". This helps to split the purpose of generating text into two categories:

1. Search purpose, i.e. the user with search query wants to find an answer inmetagraph. In this case, the model will be like the QnA system.
2. Descriptive purpose, i.e. the user doesn't have a certain question in the inputquery, rather the user wants to learn about something. The algorithm will subsequently focus on this category of purposes.

With the use of frame semantic core, using unique approaches related to the features of the semantic core is preferable because of the strict and descriptive structure of frames in FrameNet.

In the case of search, the algorithm will look for missing values of the vertex in the purpose frame. In the case of requests with an ambiguous answer (i.e. purpose were descriptive), a multi-frame response is obtained, where it is enough to request searches for certain vertices, and then search for all frames associated with this vertex, getting more than one frame. Then all the relevant parts of the metagraph will be picked out for the next step.

In both cases, the result is a subgraph of the metagraph representing the response, which is then converted to text. For text transformation, you can use both algorithmic approaches and deep neural networks (for example, T5 [11]).

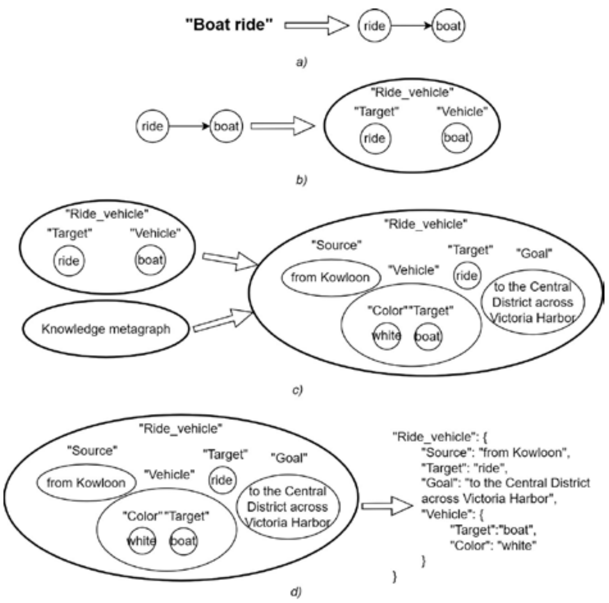
For syntactic transformation, the only thing that is needed is the following:

1. The resulting parts of the metagraph are sorted for generating text. This isdone by ATTOrderNet [4] or similar models [18].
2. The frame chain is split into parts that will represent future sentences. Itcan be done because semantic frames are well-structured, and each one of the frames makes one meaning. Sentences are generated based on the frame description of FrameNet. With the usage of Flesch reading ease scale for natural languages [5] it can be completed without complex logic: $FRE = 206{,}835 - 1{,}015 \times ASL - 84{,}6 \times ASW$; where ASL is an average number of words per sentence, and ASW is the average number of syllables per word. For each language, this formula slightly shifts in coefficients, but the variables stay the same. It is worth to be noted, that this method of analyzing the complexity of text tends to be a poor cause of used mean variables. But in the case of splitting frames into sentences, this approach is enough.
3. Supplementing a chain of frames with punctuation marks. In most cases,this will be a comma between frames, unless a period is included.

The presented algorithm can generate texts based on incoming text queries and constructed metagraph. In subsequent works, it is planned to refine the algorithm in the direction of generating texts in other languages, as well as generating texts with different styles.

5. Experiment

This system will be tested for the tasks of generating text based on data. A prototype was developed to test the performance of the system.



```
"Ride_vehicle": {
    "Source": "from Kowloon",
    "Target": "ride",
    "Goal": "to the Central District
    across Victoria Harbor",
    "Vehicle": {
        "Target":"boat",
        "Color": "white"
    }
}
```

Fig.6. Example of implementation of parsing and generating text

The system receives the user's input query in the form of a text string. According to the rules described in paragraph "The Text Parsing Module" a text string is converted into a graph. After that, the query subgraph is searched in the graph. In case of successful finding of the subgraph, the selection of vertices and links is called the response subgraph.

Next, the selected response subgraph is sent to the text generation module. T5 input must be represented as text, so we convert the graph to a text string according to the following rules:

1. Selected vertices of the response graph are translated into the response dictionary.
2. The root of the dictionary contains the main vertices.
3. The subtrees of the selected vertices turns into nested dictionaries.

Suppose we need to find certain information about a boat ride in the metagraph. In the Fig. 6 a), the input query transforms into syntax tree according to parse module. In the Fig. 6 b), the syntax tree parsed into metagraph that is called query metagraph. In the Fig. 6 c), shows the extraction of the response metagraph from the main metagraph by query metagraph. In the Fig. 6 d), the response metagraph is converted to a dictionary with nested vertex descriptions that is called response dictionary.

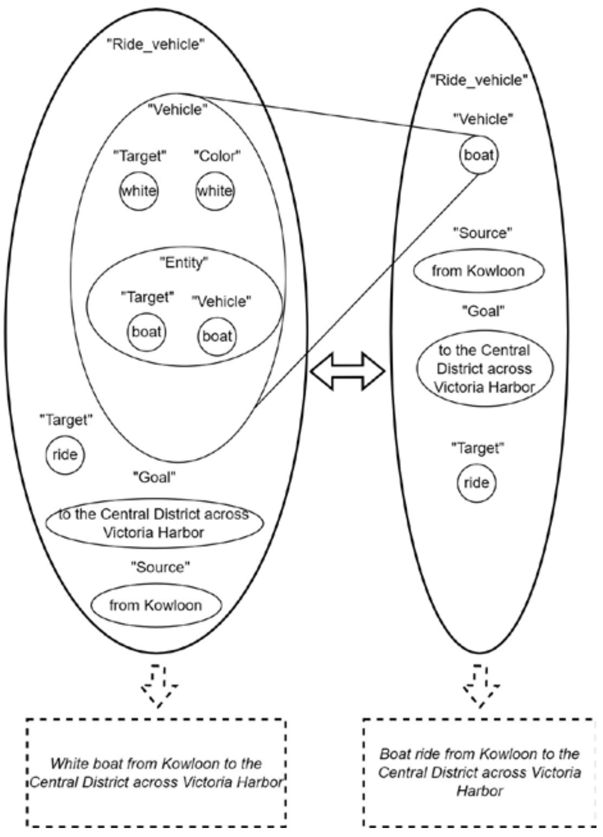

"Ride_vehicle"
"Vehicle"
"Target"
"Color"
white
white
"Entity"
"Target"
"Vehicle"
boat
boat
"Target"
ride
"Goal"
to the Central District across Victoria Harbor
"Source"
from Kowloon
"Ride_vehicle"
"Vehicle"
boat
"Source"
from Kowloon
"Goal"
to the Central District across Victoria Harbor
"Target"
ride
White boat from Kowloon to the Central District across Victoria Harbor
Boat ride from Kowloon to the Central District across Victoria Harbor

Fig.7. Example of Frame Refining and Sentence Representation

The T5 model [11] is used for transforming response dictionary (in text representation) into output text. It is worth noting that T5 can be fine-tuned to generate the desired style of speech, the desired detail of the answer. In this experiment, a FrameNet dataset [2] is used, consisting mainly of news messages. Detailing of answers varies by the number of selected frames.

There were about 5000 frames and their combinations in the original dataset. The dataset was divided into training and test samples in the ratio of 90% and 10%.

The response subgraph selection algorithm requires additional research. However, already now it is possible to vary the number of vertices and links of the response subgraph used to generate the response, get more or less detailed answers, control the subject of the response, the subject area of the response and many other factors of answer generation.

For example, in the Fig. 7 one can see how the graph is refined. So, by expanding the meaning of the word "boat" to "white boat", we change the output text.

Table 1

Result of the Experiments.

Metric Name	**Metric Value**
BLEU	0.501
METEOR	0.694
Cosine Similarity	0.813

Results of the test dataset validation are shown in Table 1. According to the definition of BLEU and METEOR metrics, result text theme is the same theme as referenced text theme. To confirm that the referenced and result texts are similar, cosine similarity is calculated.

The conducted experiment proves the idea of this system. The proposed system allows generating text based on data. It is worth noting, the FrameNet dataset has a relatively small number of sentences and frames. In the future, it is planned to increase the dataset with other sources, including texts converted into a graph.

6. Related Works and Discussion

The approaches proposed in the article relate to several areas of NLP, such as knowledge graph representation, natural language generation (NLG) and language modelling.

The article [22] proposes a text-enhanced knowledge graph representation model, named BCRL, which utilizes entity description and relation mention to enhance the knowledge representations of a triple. BCRL based on TransE [24] which is an energy-based model that produces knowledge base embeddings. It models relationships by interpreting them as translations operating on the lowdimensional embeddings of the entities.

Article [20] implements a system for generating the end of a story based on graphs. The implementation of the answer selection system is implemented as follows: several nodes of the graph are selected, each of which is weighted in some way, then these nodes fall into the answer. GPT-2 [25] and others were used as technologies.

The article [21] describes a language model based on the transformer architecture BERT [26], but with more details about the location of the token within the text, such as paragraph index, sentence index, and word index. The experimental results demonstrate that this proposed method works on both language models with relative position embeddings and pretrained language models with absolute position embeddings. The F1-score on datasets SQUAD1.1 and SQUAD2.0 [23] is 92.6 and 85.2 respectively.

Based on the results of the study of related works, we see the prospect of modern transformers, so in future work we will consider the option of combining metagraph concepts and transformers (e.g., BERT or GPT-2). Also, in future work it is planned to divide the system into three modules for a deeper comparison with other models.

Conclusions

The article proposes the conceptual modeling system based on a metagraph model that includes three main steps: synthesis of a conceptual model based on a text description; conceptual modeling, as a result of which new conceptual models are formed; analysis of the results of modeling, decision-making, and the formation of reports based on the decisions made.

The system architecture includes "the text parsing module", "the text generation module", and also "the metagraph concepts modeling module".

Having analyzed the existing approaches to the representation of conceptual maps, we can conclude that the main problem of the existing approaches is the use of a flat graph as a model for the representation of a conceptual map.

The "text parsing module" contains the transformation of the input text using various already existing techniques into a graph using frames. The resulting graph is enriched with additional data sources for a better description of the subject area.

The "Text generation module" is a straight forward algorithm in case of frame semantic core. The T5 model was used as a model for the experiment. Various query subgraphs are fed to the input, the output is a text.

The metagraph model can be useful for describing conceptual maps. In this case, we can consider "simple" and "complex" concepts. At the same time, "simple" concepts are modeled using ordinary vertices, and "complex" ones are modeled using metavertices. The use of metavertices to describe conceptual maps allows us to abandon the representation of a conceptual map in the form of a flat graph and switch to a holonic spatial description of a conceptual map in the form of a metagraph.

References

1. The XMind homepage. Available at: https://www.xmind.net/ (accessed August 30, 2022)
2. *Baker, C.F., C.J. Fillmore and J.B. Lowe.* 1998. The Berkeley FrameNet Project, In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1, Association for Computational Linguistics, pp: 86–90.
3. *Buzan, T.* 2018. Mind Map Mastery: The Complete Guide to Learning and Using the Most Powerful Thinking Tool in the Universe. Watkins Media.
4. *Cui, B., Y. Li, M. Chen and Z. Zhang*. 2018. Deep attentive sentence ordering network, In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, , pp: 4340–4349.
5. *Flesch, R.* 1948. A new readability yardstick. The Journal of Applied Psychology, 32(3): 221–233.
6. *Ishikawa, K.* 1986. Guide to Quality Control. Asian Productivity Organization.
7. *Lyashevskaya, O.N. and J.L. Kuznetsova*. 2009. Russian FrameNet: constructing a corpus--based dictionary of constructions [Russkij Frejmnet: k zadache sozdanija korpusnogo slovarja konstruktsij], In Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog"[Komp'juternaja Lingvistika I Intelleltual'nye Tehnologii: Po Materialam Ezhegodnoj Mezhdunarodnoj Konferentsii "Dialog"], , pp: 306–312.
8. *Lyashevskaya, O. and E. Kashkin*. 2015. FrameBank: A Database of Russian Lexical Constructions, In Analysis of Images, Social Networks and Texts, Springer International Publishing, pp: 350–360.
9. *McCreesh, C., P. Prosser and J. Trimble*. 2020. The Glasgow Subgraph Solver: Using Constraint Programming to Tackle Hard Subgraph Isomorphism Problem Variants. Graph Transformation, 316–324.
10. *Novak, J. and A.J. Cañas*. 2006. The Origins of the Concept Mapping Tool and the Continuing Evolution of the Tool. Information Visualization, 5(3): 175–184. https://doi.org/10.1057/palgrave.ivs.9500126
11. *Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P.J. Liu*. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv [cs.LG]. https://doi.org/10.48550/arxiv.1910.10683
12. *Swayamdipta, S., S. Thomson, C. Dyer and N.A. Smith*. 2017. Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold. arXiv [cs.CL]. https://doi.org/10.48550/arxiv.1706.09528
13. *Talvitie, V.* 2018. The Foundations of Psychoanalytic Theories.
14. *Gapanyuk, Y.* 2021. The development of the metagraph data and knowledge model. In Selected Contributions to the 10th International Conference on "Integrated Models and Soft Computing in Artificial Intelligence (IMSC-2021)". pp: 1–7
15. *Chernenkiy, V.M., Y.E. Gapanyuk, Y. Kaganov and I. Dunin*. 2018. Storing Metagraph Model in Relational, Document-Oriented, and Graph Databases. DAMDID/RCDL.
16. *Tarassov, V., Y. Kaganov and Y. Gapanyuk*. 2021. The Metagraph Model for Complex Networks: Definition, Calculus, and Granulation Issues, In Artificial Intelligence, Springer International Publishing, pp: 135–151. https://doi.org/10.23919/FRUCT48808.2020.9087470
17. *Chernenkiy, V., Y. Gapanyuk, A. Nardid and N. Todosiev*. 2020. The Implementation of Metagraph Agents Based on Functional Reactive Programming, In 2020 26th Conference of Open Innovations Association (FRUCT), pp: 1–8. https://doi.org/10.23919/FRUCT48808.2020.9087470
18. *Yin, Y., L. Song, J. Su, J. Zeng, C. Zhou and J. Luo*. 2019. Graph-based Neural Sentence Ordering. arXiv [cs.CL].
19. *Zhabotynska, S.A.* 2010. Principles of building conceptual models for thesaurus dictionaries. Cognition, Communication, Discourse, 1: 75–92.
20. *Ji, H., P. Ke, S. Huang, F. Wei, X. Zhu and M. Huang*. 2020. Language Generation with Multi-Hop Reasoning on Commonsense Knowledge Graph. arXiv [cs.CL]. https://doi.org/10.18653/v1/2020.emnlp-main.54
21. *Bai, H., P. Shi, J. Lin, Y. Xie, L. Tan, K. Xiong, W. Gao and M. Li*. 2021. Segatron: Segment-Aware Transformer for Language Modeling and Understanding. Proceedings of the AAAI Conference on Artificial Intelligence, 35(14): 12526–12534.
22. *Wu, G., W. Wu, L. Li, G. Zhao, D. Han and B. Qiao*. 2020. BCRL: Long Text Friendly Knowledge Graph Representation Learning, In The Semantic Web – ISWC 2020, Springer International Publishing, pp: 636–653.
23. *Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang*. 2016. SQuAD: 100,000+ Questions for Machine

Comprehension of Text. arXiv [cs.CL]. https://doi.org/10.18653/v1/D16-1264

24. *Bordes, A., N. Usunier and A. Garcia-Duran*. 2013. Translating embeddings for modeling multi-relational data. Advances in Neural Information Processing Systems.
25. *Radford, A., J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever*. 2019. Language models are unsupervised multitask learners. OpenAI Blog, 1(8): 9.
26. *Devlin, J., M.-W. Chang, K. Lee and K. Toutanova*. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv [cs.CL].

Todosiev N.D. Graduate student, Federal state budgetary institution of higher professional education «Bauman Moscow State Technical University», Moscow, Russia. Number of publications: 8. Research interests: information technology, natural language processing. E-mail: todosievnik@gmail.com.

Yankovsky V.I. Graduate student, Federal state budgetary institution of higher professional education «Bauman Moscow State Technical University», Moscow, Russia. Number of publications: 4. Research interests: information technology, natural language processing. E-mail: lucker1005000@gmail.com.

Gapanyuk Yu.E. Associate professor, Federal state budgetary institution of higher professional education «Bauman Moscow State Technical University», Moscow, Russia. Number of publications: about 100. Research interests: designing of automated systems, designing of hybrid intelligent information systems, complex graph models. E-mail: gapyu@bmstu.ru

Andreev A.M. Associate professor, Federal state budgetary institution of higher professional education «Bauman Moscow State Technical University», Moscow, Russia. Number of publications: about 100 (2 monographs). Research interests: designing of automated systems, designing of hybrid intelligent information systems, complex graph models. E-mail: arkandreev@gmail.com