Conceptual basics of semantic comparison texts in social networks

Olena Chebanyuk [1,2], Vitalii Velychko [2], Olexander Palahin [2] and Sergiy Gnatyuk [1]

[1] *National Aviation University, Lubomyr Huzar Ave., 1, Kyiv, 03058, Ukraine*
[2] *V. M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Ave. Glushkov, Kyiv, 03187, Ukraine*

Abstract

A well-known direction to define similarities of texts is based on considering matching fragments of texts (also called plagiarism) allows for defining narrow aspects of text similarities. But, the fact, that texts have the same semantics and can use, for example, synonyms or a set of other words to express the same idea left in the other hand of the many working software systems, that are aimed to define plagiarism. The paper is devoted to the representation of conceptual basics explaining how to define whether different texts are equal. Such conceptual basics are grounded on the idea of extracting facts from texts and using ontologies and growing pyramidal networks to match whether texts are equal.

Keywords

Growing pyramidal network, ontology, semantic cooperation, text, natural language processing, graph

1. Introduction

The growing values of text information require principally new approaches for information processing. One of the important conditions is to implement new practical approaches allowing to process of big values of textual information. On the other hand, social networks (Figure 1) are the sources of a very big amount of information that can be duplicated in different tweets, posts, pages, and boards.

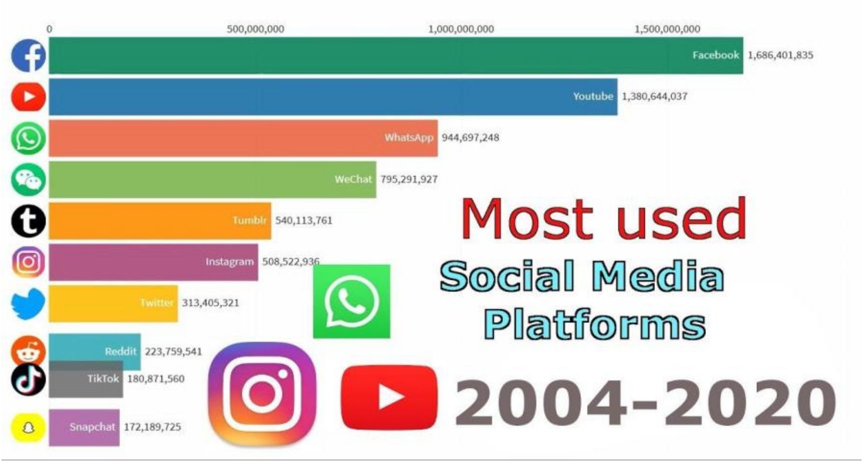

Figure 1: Most used social media

CMiGIN 2022: 2nd International Conference on Conflict Management in Global Information Networks, November 30, 2022, Kyiv, Ukraine
EMAIL: chebanyuk.elena@gmail.com (O. Chebanyuk); aduisukr@gmail.com (V. Velychko); palagin_a@ukr.net (O. Palahin); s.gnatyuk@nau.edu.ua (S. Gnatyuk)
ORCID: 0000-0002-9873-6010 (O. Chebanyuk); 0000-0002-7155-9202 (V. Velychko); 0000-0003-3223-1391 (O. Palahin); 0000-0003-4992-0564 (S. Gnatyuk)

Different users can be customers for defining simulates of textual information with different aims. For example, a researcher using social networks such as ResearchGate or ScienceDirect can define the best papers contain the best explanation of facts towards their interests. Tourist can find the best hotel and places to see according to their requires. The developer can optimize the procedure of documentation search. Many other practical tasks and practical approaches can be solved in this direction. The owner of the mailing list can define spam. Also, there are many other applications for this task.

2. Review of systems processing data from social networks

Consider working systems, allowing to processing of textual information from social networks. IBM Watson Services-based systems [1]. IBM provides a complex of systems supporting cloud technologies allowing analyzing initial information with the aim of understanding and supporting conversation by means of artificial intelligence.

One of the tasks of such a system is to classify a user request according to topic. Using such a system one can classify news, and spam or assume that text is touching on some specific topic (for example information about disaster). The procedures below must be followed in order to train the classifier before using the Natural Language Classifier service in your application: assemble the training set of data; develop and train the classifier; Inquire about the trained classifier; assess the outcomes; and update the data [1].

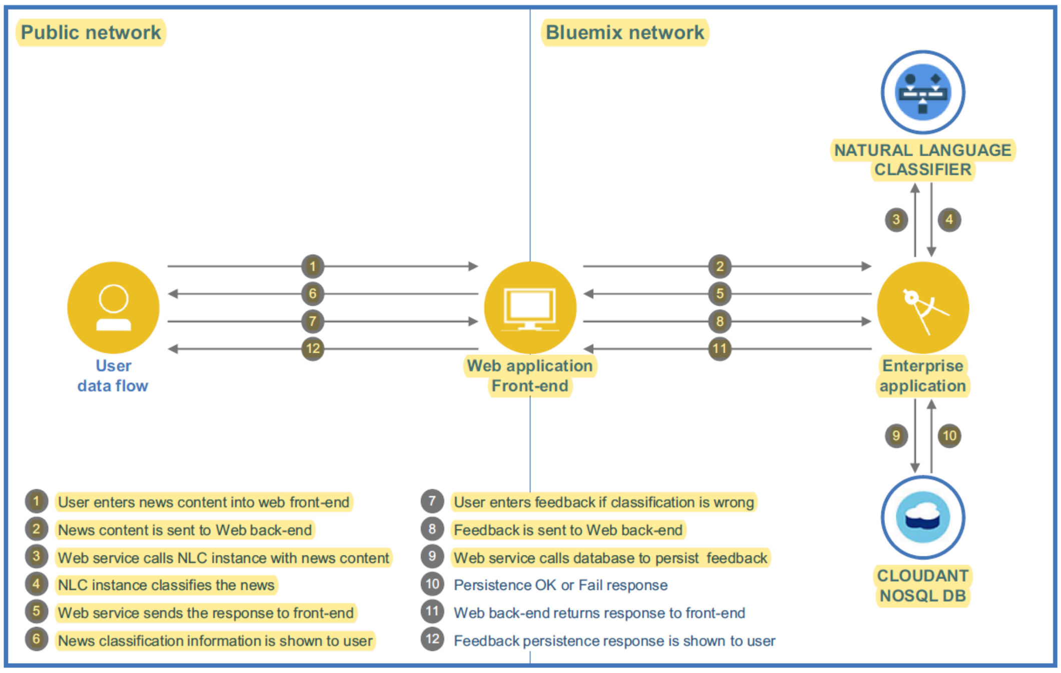
Public network
Bluemix network
NATURAL LANGUAGE CLASSIFIER
User data flow
Web application Front-end
Enterprise application
CLOUDANT NOSQL DB
1 User enters news content into web front-end
2 News content is sent to Web back-end
3 Web service calls NLC instance with news content
4 NLC instance classifies the news
5 Web service sends the response to front-end
6 News classification information is shown to user
7 User enters feedback if classification is wrong
8 Feedback is sent to Web back-end
9 Web service calls database to persist feedback
10 Persistence OK or Fail response
11 Web back-end returns response to front-end
12 Feedback persistence response is shown to user

Figure 2: News classification schema [2]

The diagram's steps are listed as follows:

1. The user inputs news information as text into the web interface and selects the Classify Text button to have it categorized.
2. The program (enterprise back end) receives the news text for processing.
3. To determine which category of news is the greatest match for the supplied text, the business application (web service) contacts the Watson Natural Language Classifier service.
4. The enterprise application receives the answer from the Natural Language Classifier service.
5. The web application front-end receives the answer from the business application.
6. The user may examine the response thanks to the web interface, which also controls the data information and carries out certain front-end processing.

7. The user feedback request is forwarded from the web application front end to the web application back end.
8. To persist the input, the back-end of the web application contacts the IBM Cloudant® noSQL DB service.
9. The database responds to the enterprise application with the results of the insert operation.
10. The web front-end receives the answer from the business application.
11. The user interface shows the user the findings.

Take into account other instances when such a classifier has been used while working with spam systems.

1. Using a web browser, the user navigates to the SPAM Classifier program [1, 3].
2. The input page is shown by the application controller.
3. The user completes and submits the input form by entering the letter topic or content [4].
4. When determining whether a message is spam or not, the application controller scans the topic or content of the mail and asks the Natural Language Classifier.
5. The application controller gives the user a web response that includes the classification outcome from the Watson Natural Language Classifier service.
6. The user offers input by indicating whether or not they agree with the categorization outcome [5, 6].
7. To update the training data, the application controller records user feedback in a data store.

The training data for the classifier [1] must include a minimum of five records (rows), a maximum of 20 thousand records, and a maximum of 3 thousand classes. The maximum total character count for a text value is 1024.

Take language categorization as an example in an application that supports a Q and A procedure [1]. A web interface, application logic, the Watson Natural Language Classifier service, and the Node.js run time make up the Healthcare Q and A application [1]. A categorization service is orchestrated by the application logic. The category to which the given query belongs is classified by the Watson Natural Language Classifier service. The Express framework serves as the integration mechanism between the Watson Natural Language Classifier service and the Node.js runtime.

1. Through the online interface, a user asks a question about healthcare.
2. A web interface posts the query that was sent to the program.
3. The application integration platform, Node.js Express, receives the query via application logic.
4. After receiving the query, Node.js forwards it to Watson's Natural Language Classifier service for classification.
5. Watson Natural Language Classifier service provides a response with the top class according to the inquiry category.
6. Node.js responds to the application logic with a Natural Language Classifier answer.
7. The web interface page to be presented is determined by application logic based on the query category returned from the Watson Natural Language Classifier service.
8. The web interface shows the page with the response to the query.

The following stage is to depict the structure of models that process text data and to define interaction pipelines with these models. To do this, think of deep DeepQA.The following elements make up the Minimum DeepQA pipeline, which takes a question as input and outputs a response along with a corresponding confidence score [7]:

- *Question analysis* - To do this, break down the question into its component components of speech and determine the many functions that each word and phrase in the sentence serve. The purpose of question analysis, which is primarily a sort of natural language processing issue, is to develop a thorough understanding of the question in terms of the entities involved, the relationships, the potential categories of solutions, etc.
- *Primary search* - Finding a group of potential sources that originate from both structured and unstructured data and contain the potential answers is the aim of the main search.
- *Hypothesis generation* - This stage aims to provide potential answers to the query from the document collection that the primary search's results have produced. Candidate answers or hypotheses are terms for solutions. Quantity now takes precedence over precision. In order to ensure

that the right response gets included, it is crucial to produce a huge number (thousands) of viable replies such that recall is near 100%.

- *Hypothesis and evidence scoring* - DeepQA uses a variety of techniques to find supporting information for each potential response. There is some proof to be found. Textual excerpts from potential responses include more evidence. DeepQA initiates a vast number of analytics once the information has been gathered, attempting to support and defend every hypothesis from every angle.
- *Final merging and ranking* - DeepQA's final step is to score each response according to its level of confidence, which is determined by combining and analyzing many sources of information. This is accomplished by utilizing machine learning strategies and historical data that provide hints and their related solutions.

Consider another category of applications, namely the process of translation. Represent technology stack for interaction with IBM BlueMix cloud [7]:

- *BlueMix;*
- *Java development environment;*
- *Node.js development environment;*
- *Flow-based development tool Node-RED;*
- *Browser application;*
- *Language classifier;*
- *Language* translator API.

Consider the pipeline of the language-translation process based on the represented technology stack [7].

1. Processing a text input (browser application)
2. Analysis of the text input source language (Java development environment)
3. Defining language (classifier)
4. Creation of the Language translator service instance
5. Authentication to Language Translator API
6. Translation of the inputted text (language translator API)
7. Output the result of processing in the form of the translated text
8. Output the result in the form of the source language and confidence score

3. Conclusion from the review

Described technologies, methods, and practical approaches represent a pipeline of performing some tasks related to text processing operations. However, known industry systems do not open for cloud developer algorithms and approaches to text processing [8, 9] and give no explanation of technologies involved in the realization of software modules. For example, a relation database has limited possibilities to proceed with big values of data effectively and within defined limits of time. We need to change the database engine to work with graph databases and involve natural language processing operations to save data quickly and effectively. Other aspects of function also are closed. That is why the necessity to develop flexible methods, that support the possibility to perform different operations of text processing is open. Using analytical methods with open mathematical apparatus allows the user to develop their software models, and to defend themselves to any software license policy. In addition, one can enrich such a module with different functionality that responds to peculiarities and specific tasks.

4. Task

To propose conceptual basics of text processing that are composed of analytical representation of tasks and approaches to text processing description. Provide the possibility to proceed with large amounts of data. Satisfy the criteria of quick assess to text data.

5. Theoretical Foundation of Text Processing

Trie is a tree for storing strings that has one node for each prefix that is often used. Additional leaf nodes are used to store the strings. A trie can be compared to an m-ary tree, where m is the total number of alphabetic letters [10].

An m-way branch is used to follow the proper path in the trie, starting at the root, in order to do a search by looking at the key one character at a time. For each letter of the alphabet, a possible child exists at each node in the multi-way trie (Fig. 3). The three words BE, BED, and BACCALAUREATE are contained in the multi-way trie shown below [11].

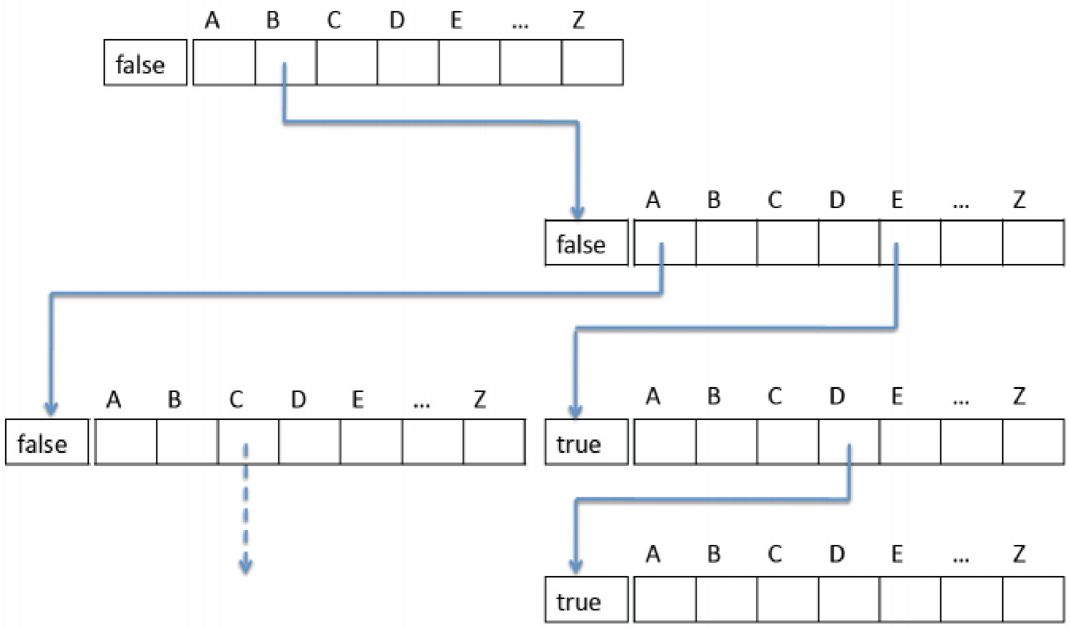

Figure 3: Example of multi-way trie [7]

Because they expressly presume that the keys are a sequence of values across a certain (finite) alphabet rather than a single indivisible item, tries differ from other data structures. The processing of variable-length keys is thus a situation where attempts excel. Because similar prefixes of words are concatenated, attempts, when done properly, can also enable compression of the set represented because words with the same prefix follow the same search route in the trie [12]. Trie [10] had utilized a list of the top 31 English terms to demonstrate his point [11].

A	FOR	IN	THE
AND	FROM	IS	THIS
ARE	HAD	IT	TO
AS	HAVE	NOT	WAS
AT	HE	OF	WHICH
BE	HER	ON	WITH
BUT	HIS	OR	YOU
BY	I	THAT	

Figure 4: The 31 most common English words [13]

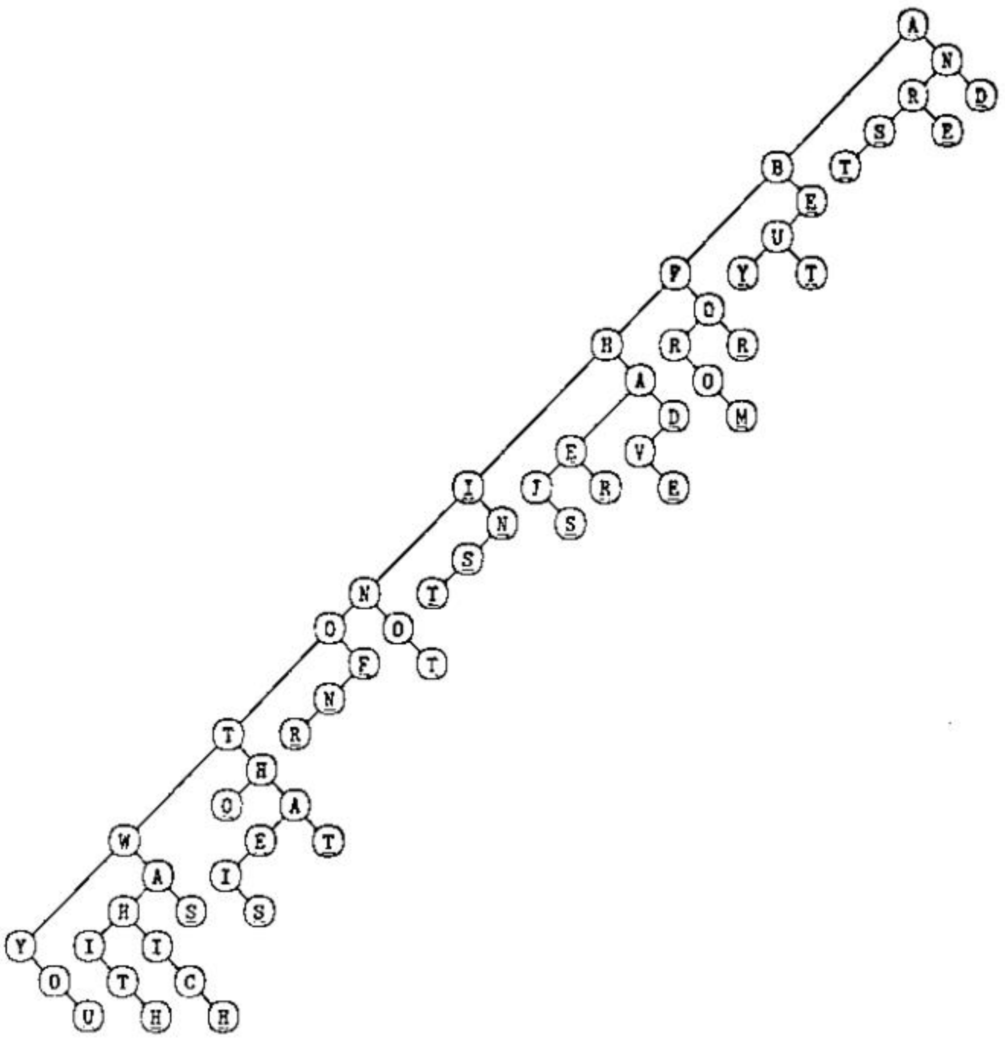
Figure 5: Linked trie for the 31 most common English words [13].

6. Proposed Conceptual Basics for Semantic Texts Comparison

Consider a general concept of knowledge processing that is presented in natural language texts. Represent it in a notation of UML sequence diagrams.

1. Preliminary analysis of the natural language text is carried out using the defined rules of facts extracted from the text. The knowledge formed from the information about the entities and relationships between them is extracted from the text (messages 1.1 and 1.2 on Fig. 6. The purpose of this step is to obtain preliminary information for an approximate estimation of the relevance of the subject of the text to the themes of the classifier stored in the ontology repository.
2. The correspondence of the subject of the text to the classifier themes is set. This operation is provided by means of matching subjects of the repository of ontologies (message 2 on Fig. 6) to text entities. The ontology classifier is presented in the form of a tree, the branches of which are the themes and subtopics of the ontology repository.
3. If the subject of the text corresponds to the subject of the classifier, then the ontology of the text is built using the knowledge that was extracted from the text earlier (messages 3.1 and 3.2 on Fig. 6). The ontology of the text is stored in the structure of a "growing pyramidal network". (Message 3.3 on fig. 6).
4. The ontology, which is transformed into a growing pyramidal network (message 4.1 on fig. 6), is stored in the repository of ontologies (message 4.2 on fig. 6). However, it can be stored as a

separate repository unit or combined with other ontologies, or certain components can be used to refine existing ontologies. Note that this preserves the traces of tracing the components of the ontology with the source text.

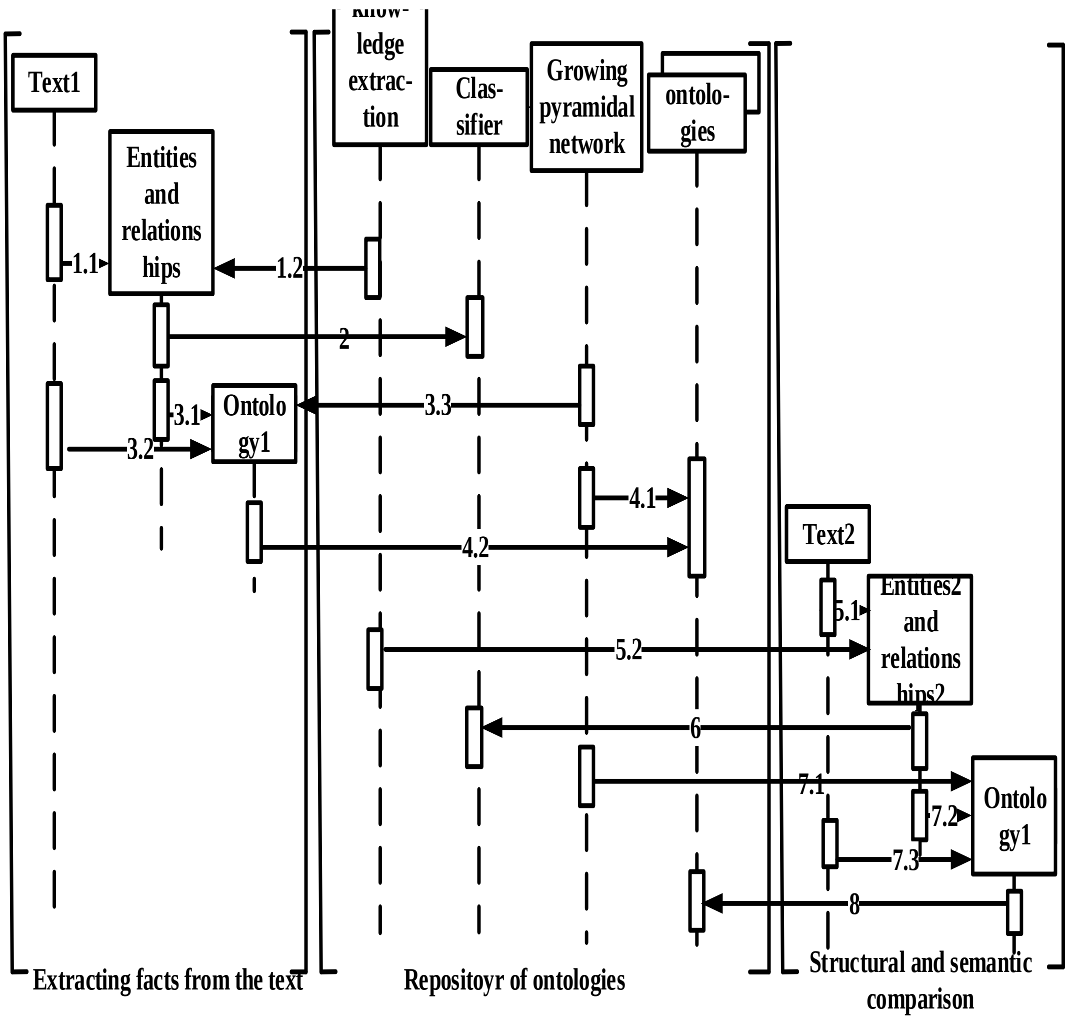
Text1
Entities and relations hips
ledge extrac-tion
Clas-sifier
Growing pyramidal network
ontolo-gies
Ontolo gy1
1.1
1.2
2
3.1
3.2
3.3
4.1
4.2
Text2
Entities2 and relations hips2
5.1
5.2
6
7.1
7.2
7.3
Ontolo gy1
8
Extracting facts from the text
Repositoyr of ontologies
Structural and semantic comparison

Figure 6: UML sequence diagram of natural language text processing

5. Another natural language text (denoted as Text2 in Fig. 6), which is the subject of processing, is considered. Information about the entities and connections (messages 5.1 and 5.2 in Fig. 6) is extracted from it using the rules of extracting knowledge from natural language texts.
6. It is determined whether the acquired knowledge corresponds to the subject of the ontology repository, by means of comparing the knowledge of the text with ones in the classifier of the ontology repository (message 6 on Fig. 6).
7. An ontology of the second text is formed, which is stored in the form of a growing pyramidal network (messages 7.1, 7.2, and 7.3 in Fig. 6).
8. In the future, possible word processing operations are performed, which are based on operations of comparing the ontologies of the repository and the processed text (obtaining answers to questions on the text, structural comparison of texts, search for the closest texts by facts, etc.).

The formal representation of the task allows to prepare of a repository of ontologies

Denote a tree of ontology repository by the next

$$W = \{w_1, w_2, ..., w_n\},$$

where n – a number of the processes in problem domain.

Denote the nested processes by the following:

$$w_1 = \{w_{1,1}, w_{1,2}, ..., w_{m,n_1}\}$$

In this example process is represented as a root process.

Initial information for composing the ontology repository consists from the triple of the next sources of information that can be transformed into text using modern ways and approaches to get text information from the audio and video content is denoted by the following: *T* – texts in natural languages; *A* – information from the audio resources; *V* – information from video resources, that is extracted from audio tracks.

Denote a triple of source for ontologies designing for the process of ontology tree as the number *i, i=1, ..., n,* where *n* – is a number of problem domain processes as $input_i$. Then

$$input_i = \{T_i, A_i, V_i\}.$$

If some type of initial information is absent, corresponding $input_i$ left empty.

The lower index defines the number of problem domain processes, for example T_i – natural language text corresponding to process number *i* of the tree.

A full set of processes, containing all arrays of initial information from sources *T*,*A*, and *V* is denoted by the following:

$$INPUT = \{input_1, input_2, ..., input_n\}.$$

Represent ontology of repository corresponding to process number *i* of repository tree by the following:

$$R_i = \{L_i^K, GPN^R\}, \qquad (1)$$

where L^K – tracing links between growing pyramidal networks of repository and $input_i$; GPN^K – Growing Pyramidal Network of Repository.

Higher index R (Repository) defines the entities before performing of procedures of its filling, for example, L^K, GPN^K .

A set of growing Pyramidal Networks of ontology repository is represented in the following way:

Denote a set of tracing links between ontologies and triple $input_i$ as □ . Other words $NL_i^R : input_i$ □ GPN_i^A.

Higher index A (Add) defines ontologies and Growing Pyramidal Networks that are designed in the current operation of repository filling. (For example, GPN_i^A, L_i^A).

Denotation that starts from the letter N (New) defines the content of growing pyramidal networks after the procedure of repository ontologies filling. For example ($NL_i^R, NGPN_i^R$).

The initial information for repository of ontologies filling is represented in the following way:

$$\begin{cases} W = \{w_1, w_2, ..., w_n\} \\ INPUT = \{input_1, input_2, ..., input_n\} \\ input_i = \{T_i, A_i, V_i\}, i = 1, ..., n \\ R = \{R_1, R_2, ..., R_n\} \\ R_i = \{L_i^R, GPN_i^R\}, i = 1, ..., n \end{cases} \qquad (2)$$

Denote an operation of filling unique information as $\cup$, for example adding to repository only unique information to Growing pyramidal network is denoted by the following: $NGPN_i^R = GPN_i^R \cup GPN_i^D$.

Denote the resulting information after repository of ontologies filling by the next:

$$\begin{cases} W = \{w_1, w_2, ..., w_n\} \\ INPUT = \{input_1, input_2, ..., input_n\}, input_i = \{T_i, A_i, V_i\}, i = 1, ..., n \\ NR = \{NR_1, NR_2, ..., NR_n\}, NR_i = \{NL_i^R, NGPN_i^R\}, i = 1, ..., n \\ NGPN_i^R = GPN_i^R \cup GPN_i^A, i = 1, ..., n \\ NL_i^R = L_i^R \cup L_i^A \ , NL_i^R : NLGPN_i^R \square \quad NLO_i^R \square \quad \{T_i^A, A^A{}_i, V^A{}_i\}, i = 1, ..., n \end{cases} \tag{3}$$
.

The task of filling the repository of ontologies is solved in the following formulation: from the tuple of source information *INPUT*, which contains different types of sources of source information, sorted by processes of the ontology repository tree, to prepare a set of growing pyramidal networks containing trace connections with

$$NL_i^R : input_i \square \quad \{T_i^A, A^A{}_i, V^A{}_i\}, i = 1, ..., n \ ,$$

(where n is the number of tree processes), including in the repository only such metamodels that do not contain duplicate information with existing ones.

7. Conclusion

The paper contains the result of research aimed to propose a conceptual basics of semantic text comparison based on facts, extracted from a text. It is proposed to use the RDF framework to support non-relational databases, provide quick access to data, and a quick procedure for storing and reading the information. The procedure of ontology designing allows us to perform the procedure of searching facts in text effectively as well as find answers to questions with a given level of accuracy. Growing Pyramidal Networks provide quick access to any entities and facts, allowing to adoption of technologies of data processing to big data. Proposed conceptual basics allow to design of a flexible architecture and software modules removing limitations of existing technologies of data processing.

8. References

[1] M. Manhaes, et al. Building Cognitive Applications with IBM Watson Services: Volume 4 Natural Language Classifier, IBM Redbooks, 2017. https://www.redbooks.ibm.com/abstracts/sg248391.html.

[2] IBM Classification Module: Make It Work for You. http://www.redbooks.ibm.com/redbooks/pdfs/sg247707.pdf.

[3] A. Ahraminezhad, M. Mojarad, H. Arfaeinia, An Intelligent Ensemble Classification Method for Spam Diagnosis in Social Networks, International Journal of Intelligent Systems and Applications 14(1) (2022) 24-31. doi: 10.5815/ijisa.2022.01.02.

[4] I. Alakbarova, Determining the interests of Social Network Users, International Journal of Education and Management Engineering 13(4) (2023) 1-8, 2023. doi:10.5815/ijeme.2023.04.01.

[5] T. Almutiri, F. Nadeem, Markov Models Applications in Natural Language Processing: A Survey, International Journal of Information Technology and Computer Science 14(2) (2022) 1-16. doi: 10.5815/ijitcs.2022.02.01.

[6] I. Korobiichuk, S. Fedushko, A. Juś, Y. Syerov, Methods of determining information support of web community user personal data verification system, Advances in Intelligent Systems and Computing, volume 550, 2017, Springer, Cham, pp 144–150. doi: https://doi.org/10.1007/978-3-319-54042-9_13.

[7] M. N. Omri, F. Fkih, Dynamic Editing Distance-based Extracting Relevant Information Approach from Social Networks, International Journal of Computer Network and Information Security 14(6) (2022) 1-13. doi:10.5815/ijcnis.2022.06.01

[8] A. Rahman, A. Nayem, M. Amjad, S. Siddik, How do Machine Learning Algorithms Effectively Classify Toxic Comments? An Empirical Analysis, International Journal of

Intelligent Systems and Applications 15(4) (2023) 1-14, 2023. DOI:10.5815/ijisa.2023.04.01.

[9] G. Alfio, et al. Building Cognitive Applications with IBM Watson Services, Getting Started, IBM Redbooks, volume 1, 2017.

[10] K. Markov, et al. Natural Language Addressing, IBS ISC, Kyiv, Madrid, Sofia, 2015.

[11] F. Pfenning, Lecture Notes on Tries, in: Principles of Imperative Computation, 2012. http://www.cs.cmu.edu/~fp/courses/15122-f12/lectures/21-tries.

[12] S. Sahni, Tries, in: Handbook of data structures and applications, D. P. Mehta, S. Sahni (Eds.), Chapman & Hall/CRC computer & information science, 2005.

[13] F. Liang, Word Hyphenation by Computer, PhD thesis, Department of Computer Science, Stanford University, Stanford, California 94305, 1983. http://www.tug.org/docs/liang/liang-thesis.pdf.