DISTRIBUTIONAL SEMANTIC MODELING: A REVISED TECHNIQUE TO TRAIN TERM/WORD VECTOR SPACE MODELS APPLYING THE ONTOLOGY-RELATED APPROACH

O.V. Palagin, V.Yu. Velychko, K.S. Malakhov, O.S. Shchurov

We design a new technique for the distributional semantic modeling with a neural network-based approach to learn distributed term representations (or term embeddings) – term vector space models as a result, inspired by the recent ontology-related approach (using different types of contextual knowledge such as syntactic knowledge, terminological knowledge, semantic knowledge, etc.) to the identification of terms (term extraction) and relations between them (relation extraction) called semantic pre-processing technology – *SPT*. Our method relies on automatic term extraction from the natural language texts and subsequent formation of the problem-oriented or application-oriented (also deeply annotated) text corpora where the fundamental entity is the *term* (includes non-compositional and compositional terms). This gives us an opportunity to changeover from distributed word representations (or word embeddings) to distributed term representations (or term embeddings). The main practical result of our work is the development kit (set of toolkits represented as web service APIs and web application), which provides all necessary routines for the basic linguistic pre-processing and the semantic pre-processing of the natural language texts in Ukrainian for future training of term vector space models.

Key words: distributional semantics, vector space model, word embedding, term extraction, term embedding, ontology, ontology engineering.

В роботі запропоновано новий метод дистрибутивно-семантичного моделювання з елементами онтологічного інжинірингу (а саме, автоматичне добування термінів) для навчання передбачуваних моделей дистрибутивної семантики з використанням векторного представлення термінів – *term embeddings*. В основі запропонованого методу лежить нова технологія обчислювальної/математичної лінгвістики для обробки природномовних текстів, що отримала назву – технологія семантичного пре-процесингу текстів. Технологія семантичного пре-процесингу текстів базується на автоматичному синтактико-семантичному аналізі природномовних текстів, зокрема, автоматичному добуванні/виокремленні (їх ідентифікація, валідація та розмітка) термінів з подальшим формуванням проблемно-орієнтованих, глибоко анотованих текстових корпусів, в яких фундаментальною сутністю є термін (включаючи композиційні терміни). Це дає можливість перейти від розподіленого/векторного представлення слів до розподіленого/векторного представлення термінів. Практичним результатом роботи є розроблений набір інструментальних/програмних засобів (у вигляді веб-сервісів і веб-застосунку), який забезпечує виконання всіх необхідних процедур і функцій для реалізації технологій базової лінгвістичної попередньої обробки та семантичного пре-процесингу природномовних текстів українською мовою з подальшим навчанням дистрибутивно-семантичних моделей векторного представлення термінів.

Ключові слова: дистрибутивна семантика, векторна модель, векторне представлення слів, векторне представлення термінів, онтологія, онтологічний інжиніринг.

В работе предложен новый метод дистрибутивно-семантического моделирования с элементами онтологического инжиниринга (а именно, автоматическое извлечение терминов) для обучения предсказательных моделей дистрибутивной семантики с использованием векторного представления терминов – *term embeddings*. В основе предложенного метода лежит новая технология вычислительной/математической лингвистики для обработки естественно-языковых текстов, получившая название – технология семантического пре-процессинга текстов. Технология семантического пре-процессинга текстов основана на автоматическом синтактико-семантическом анализе естественно-языковых текстов, в частности, автоматическом извлечении терминов (их идентификация, валидация и разметка) с последующим формированием проблемно-ориентированных, глубоко аннотированных текстовых корпусов, в которых фундаментальной сущностью является термин (включая композиционные термины). Это даёт возможность перейти от распределенного/векторного представления слов к распределенному/векторному представлению терминов. Практическим результатом работы является разработанный набор инструментальных/программных средств (в виде веб-сервисов и веб-приложения), который обеспечивает выполнение всех необходимых процедур и функций для реализации технологий базовой лингвистической предобработки и семантического пре-процессинга естественно-языковых текстов на украинском языке с последующим обучением дистрибутивно-семантических моделей векторного представления терминов.

Ключевые слова: дистрибутивная семантика, векторная модель, векторное представление слов, векторное представление терминов, онтология, онтологический инжиниринг.

Introduction

Distributional semantic modeling (word embeddings) are now arguably the most popular way to computationally handle lexical semantics. The identification of terms (non-compositional and compositional) that are relevant to the domain is a vital first step in both the automated ontology development and natural language processing tasks. This task is known as term extraction. For ontology generation, terms are first found and then relations between them are extracted. In general, a term can be said to refer to a specific concept which is characteristic of a domain or sublanguage. We propose a new technique for the distributional semantic modeling applying the ontology-related approach. This technique will give us an opportunity to changeover from distributed word representations to distributed term representations. This transition will allow to generate more accurate semantic maps of different subject domains (also, of relations between input terms – it is useful to explore clusters and oppositions, or to test your hypotheses about them).

Background. The distributed numerical feature representations of words (word embeddings): word2vec, fastText, ELMo, Gensim

The distributed numerical feature representations of words (word embeddings) and word vector space models, as a result, are well established in the field of computational linguistics and have been here for decades (see [1, 2] for an extensive review). However, recently they received substantially growing attention. Learning word representations lies at the very foundation of many natural language processing (NLP) tasks because many NLP tasks rely on good feature representations for words that preserve their semantics as well as their context in a language. For example, the feature representation of the word *car* should be very different from *fox* as these words are rarely used in similar contexts, whereas the representations of *car* and *vehicle* should be very similar. In distributional semantics, words are usually represented as vectors in a multi-dimensional space of their contexts. Semantic similarity between two words is then calculated as a cosine similarity between their corresponding vectors; it takes values between -1 and 1 (usually only values above 0 are used in practical tasks). 0 value roughly means the words lack similar contexts, and thus their meanings are unrelated to each other. 1 value means that the words' contexts are identical, and thus their meaning is very similar. Word vector space models have been applied successfully to the following tasks: finding semantic similarity between words and multi-word expressions [3]; word clustering based on semantic similarity [4]; automatic creation of thesauri and bilingual dictionaries; lexical ambiguity resolution; expanding search requests using synonyms and associations; defining the topic of a document; document clustering for information retrieval [4]; data mining and named entities recognition [5]; creating semantic maps of different subject domains and word embeddings graphs [6]; paraphrasing; sentiment analysis [7]. Despite of fundamental differences in ontology-related and neural network-based approaches, vector space models can be used as a part of ontology engineering methodologies [8, 9] as well as a part of ontology engineering development kits [**Помилка! Джерело посилання не знайдено.**9–12].

Arguably the most important applications and tools of machine learning in text analysis now are *word2vec* [13, 14] and *fastText* [15] with its Continuous Skip-Gram (CSG) and Continuous Bag of Words (CBOW) algorithms proposed in [16–18], which allow fast training on huge amounts of raw or pre-processed linguistic data.

"You shall know a word by the company it keeps" – this statement, uttered by J.R. Firth [19] in 1957, lies at the very foundation of word2vec, as word2vec techniques use the context of a given word to learn its semantics. Word2vec is a groundbreaking approach that allows to learn the meaning of words without any human intervention. Also, word2vec learns numerical representations of words by looking at the words surrounding a given word. The magic of word2vec is in how it manages to capture the semantic representation of words in a vector. The papers, Efficient Estimation of Word Representations in Vector Space [16], Distributed Representations of Words and Phrases and their Compositionality [20], and Linguistic Regularities in Continuous Space Word Representations [21] lay the foundations for word2vec and describe their uses. There are two main algorithms to perform word2vec training, which are the CBOW and CSG models. The underlying architecture of these models is described in the original research paper, but both of these methods involve in understanding the context which we talked about before. The papers written by Mikolov and others provide further details of the training process, and since the code is public, it means we know what's going on under the hood.

FastText [15, 17, 22] is a library for efficient learning of word representations and sentence classification. It is written in C++ and supports multiprocessing during training. FastText allows to train supervised and unsupervised representations of words and sentences. These representations (embeddings) can be used for numerous applications from data compression, as features into additional models, for candidate selection, or as initializers for transfer learning. FastText supports training CBOW or CSG models using negative sampling, softmax or hierarchical softmax loss functions. The main difference from word2vec that fastText can achieve really good performance for word representations and sentence classification, especially in the case of rare words by making use of character-level information [17, 18, 22]. Each word is represented as a bag of character n-grams in addition to the word itself, so for example, for the word matter, with n = 3, the fastText representations for the character n-grams are <ma, mat, att, tte, ter, er>. <and> are added as boundary symbols to distinguish the n-gram of a word from a word itself, so for example, if the word mat is part of the vocabulary, it is represented as <mat>. This helps preserve the meaning of shorter words that may show up as n-grams of other words. Inherently, this also allows you to capture meaning for suffixes/prefixes.

The length of n-grams you use can be controlled by the -minn and -maxn flags for minimum and maximum number of characters to use respectively. These control the range of values to get n-grams for. The model is considered to be a bag of words model because aside of the sliding window of n-gram selection, there is no internal structure of a word that is taken into account for featurization, i.e. as long as the characters fall under the window, the order of the character n-grams does not matter. You can also turn n-gram embeddings completely off as well by setting them both to 0. This can be useful when the 'words' in your model aren't words for a particular language, and character level n-grams would not make sense. The most common use case is when you're putting in ids as your words. During the model update, fastText learns weights for each of the n-grams as well as the entire word token.

The context is important. One of the biggest breakthroughs in distributional semantic modeling is the Embeddings from Language Models (ELMo) representations [23] – developed in 2018 by AllenNLP [24], it goes beyond traditional embedding techniques. It uses a deep, bi-directional LSTM model to create contextualized word representations. Rather than a dictionary of words and their corresponding vectors, ELMo analyses words within the context that they are used. It is also character-based, allowing the model to form representations of out-of-vocabulary words. This, therefore, means that the way ELMo is used is quite different from word2vec or fastText. Rather than

having a dictionary 'look-up' of words and their corresponding vectors, ELMo instead creates vectors on-the-fly bypassing text through the deep learning model.

While the original C code [8] implementation of word2vec released by Google and Mikolov does an impressive job, recently, there is "state of the art" open-source Python library Gensim [25] for unsupervised topic modeling and natural language processing, using more efficient implementations of modern statistical machine learning algorithms [26–28]. Gensim includes streamed parallelized implementations of fastText, word2vec, and doc2vec algorithms. The primary features of Gensim are its memory-independent nature, multicore implementations of latent semantic analysis, latent Dirichlet allocation, random projections, hierarchical Dirichlet process HDP, as well as the ability to use latent semantic analysis LSA and latent Dirichlet allocation LDA on a cluster of computers. It also seamlessly plugs into the Python scientific computing ecosystem and can be extended with other vector space algorithms.

The conventional technique to train word vector space models

The review of the most common papers and books in the distributional semantics research area (in particular [2, 29–32]) allowed to create a typical technique for the distributional semantic modeling with a neural network-based approach to learn word embeddings (word vector space models of semantics as a result) represented in fig. 1. Let's briefly review it.
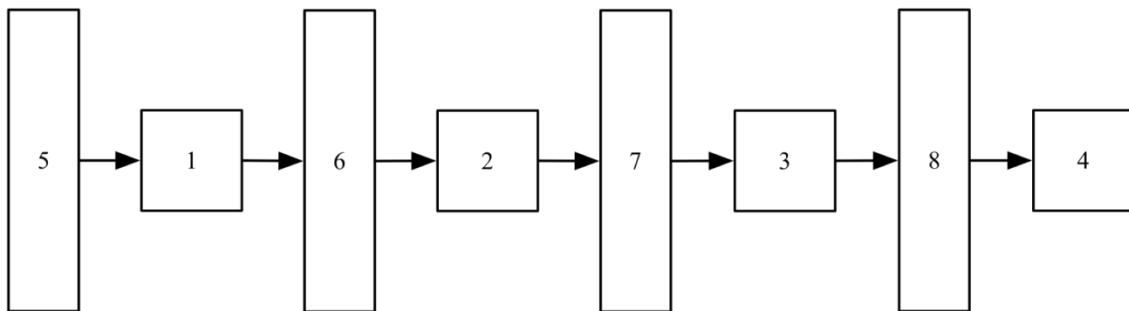

Fig. 1. A typical technique for the distributional semantic modeling with a neural network-based approach to learn word embeddings (word vector space models of semantics as a result)

On the practice, a typical technique for the distributional semantic modeling with a neural network-based approach to learn word embeddings (word vector space models of semantics as a result) consists of next components (includes technologies, pipelines and data entities/sources).

1. *Dataset (not annotated text corpus) construction/creation technology.*

The secret to getting word2vec, fastText or ELMo algorithms really working for you is to have lots and lots of text data in the relevant domain. For example, if your goal is to build a sentiment lexicon, then using a dataset from the medical domain or even Wikipedia may not be effective. Nevertheless, the most common way to construct the universal text corpus is to use a publicly available and sufficiently large Wikipedia (or other MediaWiki-based) database dump consists of Wikipedia article texts (also you can choose the language of the articles depending on your needs). The original Wikipedia dump [33] that can be downloaded is in XML format and the structure is quite complex. Thus we need to use an extractor tool to parse it. Python package Gensim provides easy to use application programming interface (API) *genism.corpora.wikicorpus* [25] for that purpose and uses multiprocessing internally to parallelize the work and process the dump more quickly. Dataset (not annotated text corpus) represented in the simple text format is the result of applying this technology.

2. *NLP pre-processing pipeline.*

An NLP pre-processing pipeline of Dataset (not annotated text corpus) typically consists of the following components [32].

Tokenization – is the task of splitting the input text into very simple units, called tokens, which generally correspond to words, numbers and symbols, and are typically separated by white space in English for example, Tokenization is a required step in almost any linguistic processing application, since more complex algorithms such as part of speech taggers mostly require tokens as their input, rather than using the raw text. Consequently, it is important to use a high-quality tokenizer, as errors are likely to affect the results of all NLP components in the pipeline.

Sentence splitting – or sentence detection is the task of separating the text into its constituent sentences. This typically involves determining whether punctuation (full stops, commas, exclamation marks, and question marks) denote the end of a sentence or something else (quoted speech, abbreviations, etc.).

Part of speech (POS) tagging – is concern with tagging words with their parts of speech (e.g., noun, verb, adjective). These basic linguistic categories are typically divided into quite fine-grained tags, distinguishing for instance between singular and plural nouns, and different tenses of verbs. For languages other than English, for example, Ukrainian, gender may also be included in the tag.

Elements of the morphological analysis (stemming, lemmatization, filtering out common stopwords) – essentially concern the identification and classification of the linguistic units of a word, typically breaking the word

down into its root form and an affix. Lemmatization will reduce vocabulary (for word2vec, fastText, ELMo) and increase text coherence.

Elements of the syntactic parsing (including chunking). Syntactic parsing is concerned with analyzing sentences to derive their syntactic structure according to a grammar. Essentially, parsing explains how different elements in a sentence are related to each other, such as how the subject and object of a verb are connected. Also base chunking (or shallow parsing) is required to get noun phrases and verb phrases for a simple compositional vocabulary (for word2vec, fastText, ELMo) creation. Note that there is a *gensim.models.phrases* module of the Python Gensim package, which lets you automatically detect phrases longer than one word. Using phrases, you can learn a model where words (entities) are actually multiword expressions, such as "the_national_academy" or "financial_crisis". An annotated text corpus is the result of applying this pipeline.

The most common way to implement NLP pre-processing pipeline is to use open-source linguistic pre-processing toolkits such as spaCy [34], NLTK [35], StanfordNLP [36]. spaCy is a free, open-source library for advanced NLP) in Python. It can be used to build information extraction or natural language understanding systems or to pre-process text for deep learning and computational linguistics purposes. Following are the features of spaCy (the Ukrainian language is not supported for now): Non-destructive tokenization; support for more than 21 natural languages; 6 statistical models for 5 languages; Pre-trained word vectors (word embeddings); POS tagging; named entity recognition (NER); labeled dependency parsing; syntax-driven sentence segmentation; built-in visualizers for syntax and NER; export to NumPy data arrays; efficient binary serialization; robust, rigorously evaluated accuracy, etc. spaCy is designed specifically for production use and helps you build applications that process and "understand" large volumes of text. On the other hand, NLTK' and StanfordNLP' primary focus is to give students and researchers a toolkit to play around with computational linguistics algorithms is not a production environment.

3. *Word vector space models (word embeddings) training technology.*

Gensim open-source Python library is the most suitable to implement the process of training the new word vector space models. Gensim package has the following API (modules) for this purpose [37]: *genism.models.word2vec* (this module implements the word2vec family of algorithms, using highly optimized C routines, data streaming, and pythonic interfaces; the word2vec algorithms include CSG and CBOW models, using either hierarchical softmax or negative sampling); *genism.models.fasttext* (this module allows training word embeddings from a training corpus with the additional ability to obtain word vectors for out-of-vocabulary words; this module contains a fast native C implementation of fastText with Python interfaces); *genism.models.doc2vec* (this module implements learning paragraph and document embeddings via the distributed memory and distributed bag of words models from [38]); *genism.models.keyedvectors* (this module implements word vectors and their similarity look-ups; since trained word vectors are independent from the way they were trained, they can be represented by a standalone structure, as implemented in this module). Another important thing about using CSG and CBOW algorithms are hyperparameters values. The detailed review and recommendations of hyperparameters represented in [39, 40].

4. *Word vector space model (word embeddings) processing (load/serialization and usage) technology.*

Deploying deep learning models in the production environment is challenging. There are different ways you can have a model deployed: loading model directly into application (this option essentially considers the model a part of the overall application and hence loads it within the application); calling an API (this option involves making an API and calling the API from your application, this can be done in several different ways, for example, via the Kubernetes open-source container orchestration system for automating application deployment); Serverless cloud-computing execution model (the cloud provider runs the server, and dynamically manages the allocation of machine resources, such as AWS Lambda function as a service solution [41]); custom representational state transfer REST API with Flask/Django Python packages from scratch (this option could possibly be combined with Docker and firefly [42] Python package as well).

5. *Sources of text documents and corpora.*

These are analog texts, the Internet, corpora of Wikipedia texts, electronic collections of text documents, databases, etc.

6. *Dataset (not annotated text corpus) entity.*

The most common datasets include an entire corpus of Wikipedia texts, the common crawl dataset [43], or the Google News Dataset or use the Dataset Search [44] toolkit from Google. Note if your future application is specific to a certain domain the dataset must be relevant.

7. *Annotated text corpus entity.*

Annotation consists of the application of a scheme to texts. Annotations may include structural markup, POS tagging, parsing, and numerous other representations.

8. *Word vector space model (word embeddings or distributional semantic model) entity.*

The revised technique to train term embeddings (term vector space models as a result) applying the ontology-related approach

In this section, we propose a new technique for the distributional semantic modeling applying the ontology-related approach. Our method relies on automatic term extraction from the natural language texts and subsequent formation of the problem-oriented or application-oriented (also deeply annotated) text corpora where the

fundamental entity is the term (includes non-compositional and compositional terms). This technique will give us an opportunity to changeover from distributed word representations (or word embeddings) to distributed term representations (or term embeddings). This transition will allow generating more accurate *semantic maps* of different subject domains (also, of relations between input terms – it is useful to explore clusters and oppositions or to test your hypotheses about them). The semantic map can be represented as a graph using, for example, Vec2graph [6, 45] – a Python library for visualizing word embeddings (term embeddings in our case) as dynamic and interactive graphs. Using the Vec2graph library coupled with term embeddings will not only improve accuracy in solving standard NLP tasks but also update the conventional concept of automated ontology development [32] (which comprises three related components: learning, population and refinement) despite the fundamental differences between the ontology-related approach and distributional semantic modeling techniques (that relies on top of the statistical semantics). Ontology learning (or generation) denotes the task of creating an ontology from scratch and mainly concerns the task of defining the concepts – *terms* and generating relevant relations between them. The ontology population consists of adding instances to an existing ontology structure (as created by the ontology learning task, for instance). Ontology refinement involves adding, deleting, or changing new terms, relations, and/or instances in an existing ontology. Ontology learning may also be used to denote all three tasks, in particular where the tasks of learning and population are performed via a single methodology. For all three components of ontology development, the starting point is typically a large corpus of unstructured text (which may be the entire web itself, or a set of domain-specific documents). For now, the automated ontology development is beyond the scope of this paper but will be concerned in future researches.

Fig. 2 shows the new technique to train distributed term representations (or term embeddings) – term vector space model as a result.
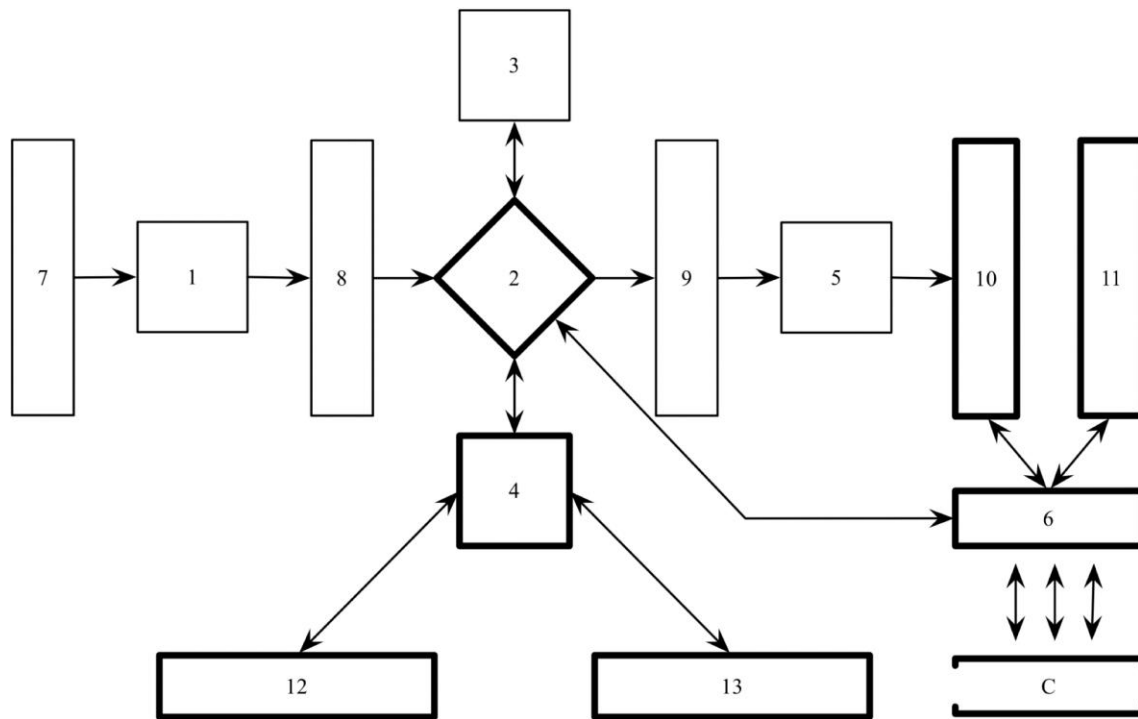

Fig. 2. The new technique to train distributed term representations (or term embeddings) – term vector space models as a result

The new consists of next components (includes technologies, pipelines and data entities/sources).

1. *Dataset* (not annotated text corpus) construction/creation technology. This component complies with the conventional technique.

2. The single-page *web application* for managing datasets (not annotated and deeply annotated text corpora) and separate text documents processing technologies.

3. The basic *NLP pre-processing pipeline.*

4. *Semantic pre-processing technology SPT.*

SPT inspired by the recent ontology-related approach (using different types of contextual knowledge such as syntactic knowledge, terminological knowledge, semantic knowledge, etc.) to the identification of terms (term extraction) and relations between them (relation extraction). For languages with advanced morphology, such as Ukrainian and Russian, inflexions and function words are the primary means of expressing syntax in a sentence. Semantic analysis of a sentence is capable of revealing some errors of syntactic structure. Thus, there is an inverse relationship between semantic and syntactic analysis, so it is advisable to combine these two stages of analysis and

execute them together in one analytical unit. SPT technology implemented as a web service (server-side web API interface consisting of several exposed endpoints and using JSON for request–response message system) on top of the functions of the "Konspekt" [46] utility for the analysis of the Ukrainian and Russian languages. Let us consider the algorithm of the STP analysis (syntactic and semantic analysis), which is implemented in the "Konspekt" utility. To find a connection between separate words, inflectional means are used to express semantic and syntactic relations. An indicator of the morphological dependency between words is inflection. Segments of phrases that encode the relations between content words, and consist of inflexions and function words are called syntactic determinants [47]. Since several syntactic relations can correspond to one syntactic determinant, *the concept of a correlator* [47] is introduced for the uniqueness of determining the relations between words. Correlators additionally include the grammatical attributes of words inside the phrase.

The component of the STP analysis (syntactic and semantic analysis) uses the following input data:

– result of the previous stages of text analysis (grapheme and morphological analysis);

– dictionary of stems (contains stems of words and their semantic attributes);

– list of all possible inflexions of words;

– dictionary of determinants (contains syntactic determinants and lists of correlators for each of them);

– dictionary of correlators (each correlator consists of the name of the relation and a list of pairs of semantic attributes of words between which this relation can exist).

Let us consider the core stages of the natural language SPT analysis (syntactic and semantic analysis) pipeline.

First stage. In each word from the sentence, its stem and inflectional component are defined with the help of the dictionary of stems and the list of inflexions. The classification of words in the sentence is implemented on the grammatical attributes respective stems in the dictionary. Possible ambiguities in the stemming process and determining its grammatical attributes are solved by analyzing the attributes of the words that a next in the sentence, and grammatical attributes of the word stem from the dictionary of stems.

Second stage. The syntagmas extraction in a sentence begins with a phrase that defines the core relation (the relation between subject and predicate). In the case when such a phrase cannot be extracted, the sentence is analyzed from left to right from the first content word. For the extracted phrase, a syntactic determinant is formed, which consists of function words and inflectional parts of content words of the phrase. If the generated determinant exists in the dictionary of determinants, a list of correlators is selected for it from the dictionary of determinants. In the correlator dictionary, a correlator is searched from the list of correlators selected in the previous stage, taking into account the grammatical attributes of the stems of words in a possible phrase. The correlator that is found determines the type of syntactic and semantic relations between words. The unambiguity of determining such a relation is ensured by the fact that for a particular determinant, the sets of pairs of grammatical attributes for the correlator from its list do not intersect. The defined content words of the sentence are added to a certain phrase by establishing a relation between the new word and one of the words of the processed part of the sentence. This creates a group of related words. It is important to select a word from the syntagma that will be associated with the following words. It should be either the word with the main relation, or the last word of the syntagma. In the case when it is impossible to determine a relation between the new word and the words of the phrase, a new syntagma is created. At the end of the analysis, all syntagmas are combined into one, which reflects the structure of relations between all the words of the sentence. The impossibility of determining a relation between syntagms in a sentence and their combination testifies either to a complex sentence, parts of which are connected (or implicitly linked) to each other, or to incorrect selection of syntagms. If there are several options for syntagma extraction, a return is made to the step of selecting the list of correlators and another option is selected for determining the syntactic and semantic relations. The best relation option is considered the option with the least number of unrelated syntagms in the sentence.

To build the conceptual structure of a natural language text, an automatic term extraction procedure is performed. The terms are considered nouns, abbreviations and noun phrases expressed by the scheme: a matched word + noun. In this model, the noun is the main word, and the matched word is dependent and can be expressed as an adjective or a noun [48]. If a noun is used as a matched word, then it is in the phrase in the genitive case. Phrases may also include prepositions and compositional conjunctions. The number of words in noun phrases for Russian texts ranges from two to fifteen words and averages three words.

An automatic terms extraction (includes compositional terms) procedure uses the results of syntactic and semantic analysis of the text. The terms extraction procedure consists of two main steps [49]. At the first step, there is a direct search in the text of words and phrases – candidates for terms. As one-word (non-compositional) terms, nouns and abbreviations are chosen. Compositional terms are formed using the types of relations between the words of a sentence defined at the previous stage of the text analysis, by gradually adding words to a one-word (non-compositional) term - a noun. For terms – noun phrases, the following basic types of relations between words that are part of the phrases are used: object relation, affiliation (between two nouns), defining relationship (between an adjective and a noun), uniformity words (between two nouns or two adjectives). When the adjective is included in the compositional term, the semantic attributes of the adjective is additionally taken into account. For the compositional term, which includes several nouns, terms of shorter length are automatically extracted and the relation between them is determined. A prerequisite for extracting the term is the correspondence of the relations between the words that are part of it, with certain types of relations.

5. *Term vector space models (term embeddings) training technology.*

This component complies with the conventional technique except that the vector space model is trained on the deeply annotated dataset, the fundamental entity of which is the terms (non-compositional and compositional terms).

6. *Term vector space models (term embeddings) processing/managing technology as a service.*

Term vector space models (term embeddings) processing/managing technology implemented as a server-side web API interface named DS-REST-API. The core functions of the DS-REST-API are: to load pre-trained model (Word2vec, fastText, etc.) and prepare it for inference; to calculate semantic similarity between pairs of terms; to find terms semantically closest to the query term (optionally with POS and frequency filters); to perform analogical inference: to find a term X which is related to the term Y in the same way as the term A is related to the term B; to find the center of term cluster formed by your positive terms.

7. *Sources of text documents and corpora.*

These are analog texts, the Internet, corpora of Wikipedia texts, electronic collections of text documents, databases, etc.

8. *Datasets* (not annotated text corpora) digital repository.

This component complies with the conventional technique.

9. *Deeply annotated datasets* (text corpora) digital repository.

This digital repository stores the deeply annotated datasets (the results of SPT processing over not annotated text corpora).

10. *Term vector space models* digital repository (contains new pre-trained models).

This is an internal digital repository for the new pre-trained word embeddings models.

11. *Sources of pre-trained distributional semantic models (word embeddings).*

These are external digital repositories of the pre-trained word embeddings models.

12. *Semantic dictionary and lexical WordNet database* (this component may also include distributional thesaurus).

This component is used to validate of the semantic relations between terms.

13. *Wikipedia* online encyclopedia.

This component is used to validate terms.

C *External clients* for web APIs of term vector space models processing/managing technology as a service.

Conclusion

We design a new technique for the distributional semantic modeling with a neural network-based approach to learn distributed term representations (or term embeddings) – term vector space models as a result, inspired by the recent ontology-related approach (using different types of contextual knowledge such as syntactic knowledge, terminological knowledge, semantic knowledge, etc.) to the identification of terms (term extraction) and relations between them (relation extraction) called semantic pre-processing technology – SPT. Our method relies on automatic term extraction from the natural language texts and subsequent formation of the problem-oriented or application-oriented (also deeply annotated) text corpora where the fundamental entity is the term (includes non-compositional and compositional terms). This gives us an opportunity to changeover from distributed word representations (or word embeddings) to distributed term representations (or term embeddings). This transition will allow to generate more accurate semantic maps of different subject domains (also, of relations between input terms – it is useful to explore clusters and oppositions, or to test your hypotheses about them). The semantic map can be represented as a graph using Vec2graph [45] – a Python library for visualizing word embeddings (term embeddings in our case) as dynamic and interactive graphs. The Vec2graph library coupled with term embeddings will not only improve accuracy in solving standard NLP tasks, but also update the conventional concept of automated ontology development. For example, we want to form a graph of the text document (the fundamental entity in this graph will be a non-compositional or compositional term). This text document may be included in the dataset on which the vector space model was trained or not included in it but it must be relevant to the domain of this model. After the SPT processing of the input text document the set of terms is extracted. Using the Vec2graph library [45] we visualize (in current vector space model) only terms which present in the input text document. The next step will be marking the relations between those terms partially based on the SPT processing results with future manual corrections by the knowledge engineer. Also, this gives us an opportunity to use pre-trained models which not based on term entities. Using pre-trained models (including contextualized ones) avoids the resource-intensive training process.

The main practical result of our work is the development kit (set of toolkits represented as web service APIs and web application), which provides all necessary routines for the basic linguistic pre-processing and the semantic pre-processing of the natural language texts in Ukrainian for future training of term vector space models.

In [50] we proposed the new class of Current Research Information Systems and related intelligent information technologies. This class supports the main stages of the scientific research and development lifecycle, starting with the semantic analysis of the information of arbitrary domain area and ending with the formation of constructive features of innovative proposals. It was called – Research and Development Workstation Environment – the comprehensive problem-oriented information systems for scientific research and development support. As part of the research and

development work of the Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine (Department of Microprocessor Technology) has developed and implemented the software system in its class. It was called – Personal Research Information System (PRIS) [50] – the RDWE class system for supporting research in the field of ontology engineering (the automated building of applied ontology in an arbitrary domain area as a main feature), scientific and technical creativity. The set of toolkits represented as web service APIs and web application are integrated in PRIS atomic web services ecosystem.

References

1. Turney P.D. & Pantel P. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research.* 2010. 37(1). P. 141–188.
2. Ganegedara T. 2018. Natural Language Processing with TensorFlow: Teach language to machines using Python's deep learning library. Packt Publishing Ltd.
3. Kutuzov A. & Andreev I.A. 2015. Texts in, meaning out: neural language models in semantic similarity task for Russian. In: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference "Dialogue"*. Moscow, May 27 – 30. Moscow: RGGU. Issue 14 (21).
4. Kutuzov A. 2014. Semantic clustering of Russian web search results: possibilities and problems. *In Russian Summer School in Information Retrieval*. Aug 18–22. Cham: Springer. P. 320–331.
5. Sienčnik S.K. Adapting word2vec to named entity recognition. In: *Proceedings of the 20th Nordic conference of computational linguistics*. Nodalida, May 11–13. Vilnius: Linköping University Electronic Press. 2015. N 109. P. 239–243.
6. Katricheva N., Yaskevich A., Lisitsina A., Zhordaniya T., Kutuzov A., Kuzmenko E. 2020. Vec2graph: A Python Library for Visualizing Word Embeddings as Graphs. In: van der Aalst W. et al. (eds) Analysis of Images, Social Networks and Texts. AIST 2019. *Communications in Computer and Information Science*. Vol. 1086. Springer, Cham.
7. Maas A.L., Daly R.E., Pham P.T., Huang D., NG, A.Y. and potts C. 2011. Learning word vectors for sentiment analysis. In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* volume 1. Association for Computational Linguistics. P. 142–150.
8. Palagin A.V., Petrenko N.G., Malakhov K.S. Technique for designing a domain ontology. *Computer means, networks and systems*. 2011. N 10. P. 5–12.
9. Palagin O.V., Petrenko M.G. and Kryvyi S.L. 2012. Ontolohichni metody ta zasoby obrobky predmetnykh znan. Publishing center of V. Dahl East Ukrainian National University.
10. Palagin A.V., Petrenko N.G., Velychko V.YU. and Malakhov K.S. 2014. Development of formal models, algorithms, procedures, engineering and functioning of the software system "Instrumental complex for ontological engineering purpose". In: *Proceedings of the 9th International Conference of Programming UkrPROG*. CEUR Workshop Proceedings 1843. Kyiv, Ukraine, May 20-22, 2014. [Online] Available from: http://ceur-ws.org/Vol-1843/221-232.pdf [Accessed: 03 February 2020].
11. Velychko V.YU., Malakhov K.S., Semenkov V.V., Strizhak A.E. Integrated Tools for Engineering Ontologies. *Information Models and Analyses*. 2014. N 4. P. 336–361.
12. Palagin A.V., Petrenko N.G., Velichko V.YU., Malakhov K.S. and Tikhonov YU.L. To the problem of "The Instrumental complex for ontological engineering purpose" software system design. *Problems in programming*. 2012. N 2-3. P. 289–298.
13. Mikolov T., Chen K., Corrado G.S. and Dean J.A., Google LLC. 2015. Computing numeric representations of words in a high-dimensional space. U.S. Patent 9,037,464.
14. Google Code Archive. Word2vec tool for computing continuous distributed representations of words. [Online] Available from: https://code.google.com/archive/p/word2vec [Accessed: 03 February 2020].
15. fastText. Library for efficient text classification and representation learning. [Online] Available from: https://fasttext.cc [Accessed: 03 February 2020].
16. Mikolov T., Chen K., Corrado G. and Dean J. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
17. Bojanowski P., Grave E., Joulin A. and Mikolov T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*. 2017. 5. P. 135–146.
18. Joulin A., Grave E., Bojanowski P., Douze M., Jégou H. and Mikolov T. 2016. Fasttext.zip: Compressing text classification models. arXiv preprint arXiv:1612.03651.
19. Encyclopædia Britannica. John R. Firth. [Online] Available from: https://www.britannica.com/biography/John-R-Firth [Accessed: 03 February 2020].
20. Mikolov T., Sutskever I., Chen K., Corrado G.S. and Dean J. Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems*. 2013. P. 3111–3119.
21. Mikolov T., Yih W.T. and Zweig G. Linguistic regularities in continuous space word representations. In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 2013. P. 746–751.
22. Joulin A., Grave E., Bojanowski P. and Mikolov T. 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.
23. Peters M.E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K. and Zettlemoyer L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.
24. AllenNLP an open-source NLP research library. ELMo. [Online] Available from: https://allennlp.org/elmo [Accessed: 03 February 2020].
25. Gensim: Topic modelling for humans. [Online] Available from: https://radimrehurek.com/gensim [Accessed: 03 February 2020].
26. Luo Q. and Xu W. Learning word vectors efficiently using shared representations and document representations. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI Press*. 2015. P. 4180–4181.
27. Luo Q., Xu W. and Guo J. A Study on the CBOW Model's Overfitting and Stability. In: *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning. Association for Computing Machinery*. 2014. P. 9–12.
28. Mnih A. and Kavukcuoglu K. Learning word embeddings efficiently with noise-contrastive estimation. In: *Advances in neural information processing systems. Curran Associates Inc*. 2013. P. 2265–2273.
29. Srinivasa-Desikan B. (2018). Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras. *Packt Publishing Ltd*.
30. Nikolenko S., Kadurin A., and Arkhangel'skaya E. (2018) Glubokoe obuchenie. Pogruzhenie v mir neironnykh setei (Deep Learning: An Immersion in the World of Neural Networks). St. Petersburg: Piter.
31. Goyal P., Pandey S., & Jain K. (2018). Deep learning for natural language processing. Deep Learning for Natural Language Processing: Creating Neural Networks with Python. Berkeley, CA]: Apress.

32. Maynard D., Bontcheva K., & Augenstein I. (2017). Natural language processing for the semantic web. *Synthesis Lectures on the Semantic Web: Theory and Technology*. Morgan & Claypool Publishers.
33. Wikimedia Downloads. [Online] Available from: https://dumps.wikimedia.org [Accessed: 03 February 2020].
34. spaCy. Industrial-strength Natural Language Processing in Python. [Online] Available from https://spacy.io [Accessed: 03 February 2020].
35. Natural Language Toolkit. NLTK 3.4.5. [Online] Available from https://www.nltk.org [Accessed: 03 February 2020].
36. StanfordNLP. Python NLP Library for Many Human Languages. [Online] Available from https://stanfordnlp.github.io/stanfordnlp [Accessed: 03 February 2020].
37. Gensim: API Reference. [Online] Available from https://radimrehurek.com/gensim/apiref.html [Accessed: 03 February 2020].
38. Le Q. and Mikolov T. Distributed representations of sentences and documents. In International conference on machine learning. 2014. P. 1188–1196.
39. Caselles-Dupré H., Lesaint F. and Royo-Letelier J. Word2vec applied to recommendation: Hyperparameters matter. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. 2018. P. 352–356.
40. Rong X. (2014). word2vec parameter learning explained. arXiv preprint arXiv:1411.2738.
41. AWS Machine Learning Blog. How to Deploy Deep Learning Models with AWS Lambda and Tensorflow. [Online] Available from https://aws.amazon.com/blogs/machine-learning/how-to-deploy-deep-learning-models-with-aws-lambda-and-tensorflow [Accessed: 03 February 2020].
42. Firefly. Firefly documentation. [Online] Available from https://rorodata.github.io/firefly [Accessed: 03 February 2020].
43. Common Crawl. [Online] Available from http://commoncrawl.org [Accessed: 03 February 2020].
44. Google Dataset Search. [Online] Available from https://datasetsearch.research.google.com [Accessed: 03 February 2020].
45. Vec2graph mini-library for producing graph visualizations from embedding models [Online] Available from: https://github.com/lizaku/vec2graph [Accessed: 03 February 2020].
46. Palagin A.V., Svitla S.JU., Petrenko M.G., Velychko V.JU. About one approach to analysis and understanding of the natural. *Computer means, networks and systems*. 2008. N 7. P. 128–137.
47. Gladun V.P. 1994. Processy formirovanija novyh znanij [Processes of formation of new knowledge]. Sofija: SD «Pedagog 6» – Sofia: ST «Teacher 6», 192 [in Russian].
48. Dobrov B., Loukachevitch N., Nevzorova O. 2003. The technology of new domains' ontologies development. *Proceedings of the X-th International Conference "Knowledge-Dialogue-Solution"* (KDS'2003). Varna, Bulgaria. 2003. P. 283–290.
49. Velychko V., Voloshin P., Svitla S. 2009. Avtomatizirovannoe sozdanie tezaurusa terminov predmetnoj oblasti dlja lokal'nyh poiskovyh sistem. International Book Series "Information Science & Computing". Book No: 15. Knowledge – Dialogue – Solution, Sofia, 2009. P. 24–31.
50. Palagin O.V., Velychko V.YU., Malakhov K.S., Shchurov O.S. (2018) Research and development workstation environment: the new class of current research information systems In: *Proceedings of the 11th International Conference of Programming UkrPROG 2018. CEUR Workshop Proceedings* 2139. Kyiv, Ukraine, May 22-24, 2018. [Online] Available from: http://ceur-ws.org/Vol-2139/255-269.pdf [Accessed: 03 February 2020].

Література

1. Turney P.D. & Pantel P. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*. 2010. 37(1). P. 141–188.
2. Ganegedara T. 2018. Natural Language Processing with TensorFlow: Teach language to machines using Python's deep learning library. Packt Publishing Ltd.
3. Kutuzov A. & Andreev I.A. 2015. Texts in, meaning out: neural language models in semantic similarity task for Russian. In: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference "Dialogue"*. Moscow, May 27 – 30. Moscow: RGGU. Issue 14 (21).
4. Kutuzov A. 2014. Semantic clustering of Russian web search results: possibilities and problems. *In Russian Summer School in Information Retrieval*. Aug 18–22. Cham: Springer. P. 320–331.
5. Sienčnik S.K. Adapting word2vec to named entity recognition. In: *Proceedings of the 20th Nordic conference of computational linguistics*. Nodalida, May 11–13. Vilnius: Linköping University Electronic Press. 2015. N 109. P. 239–243.
6. Katricheva N., Yaskevich A., Lisitsina A., Zhordaniya T., Kutuzov A., Kuzmenko E. 2020. Vec2graph: A Python Library for Visualizing Word Embeddings as Graphs. In: van der Aalst W. et al. (eds) Analysis of Images, Social Networks and Texts. AIST 2019. *Communications in Computer and Information Science*. Vol. 1086. Springer, Cham.
7. Maas A.L., Daly R.E., Pham P.T., Huang D., NG, A.Y. and potts C. 2011. Learning word vectors for sentiment analysis. In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* volume 1. Association for Computational Linguistics. P. 142–150.
8. Palagin A.V., Petrenko N.G., Malakhov K.S. Technique for designing a domain ontology. *Computer means, networks and systems*. 2011. N 10. P. 5–12.
9. Palagin O.V., Petrenko M.G. and Kryvyi S.L. 2012. Ontolohichni metody ta zasoby obrobky predmetnykh znan. Publishing center of V. Dahl East Ukrainian National University.
10. Palagin A.V., Petrenko N.G., Velychko V.YU. and Malakhov K.S. 2014. Development of formal models, algorithms, procedures, engineering and functioning of the software system "Instrumental complex for ontological engineering purpose". In: *Proceedings of the 9th International Conference of Programming UkrPROG*. CEUR Workshop Proceedings 1843. Kyiv, Ukraine, May 20-22, 2014. [Online] Available from: http://ceur-ws.org/Vol-1843/221-232.pdf [Accessed: 03 February 2020].
11. Velychko V.YU., Malakhov K.S., Semenkov V.V., Strizhak A.E. Integrated Tools for Engineering Ontologies. *Information Models and Analyses*. 2014. N 4. P. 336–361.
12. Palagin A.V., Petrenko N.G., Velichko V.YU., Malakhov K.S. and Tikhonov YU.L. To the problem of "The Instrumental complex for ontological engineering purpose" software system design. *Problems in programming*. 2012. N 2-3. P. 289–298.
13. Mikolov T., Chen K., Corrado G.S. and Dean J.A., Google LLC. 2015. Computing numeric representations of words in a high-dimensional space. U.S. Patent 9,037,464.
14. Google Code Archive. Word2vec tool for computing continuous distributed representations of words. [Online] Available from: https://code.google.com/archive/p/word2vec [Accessed: 03 February 2020].
15. fastText. Library for efficient text classification and representation learning. [Online] Available from: https://fasttext.cc [Accessed: 03 February 2020].
16. Mikolov T., Chen K., Corrado G. and Dean J. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

17. Bojanowski P., Grave E., Joulin A. and Mikolov T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*. 2017. 5. P. 135–146.

18. Joulin A., Grave E., Bojanowski P., Douze M., Jégou H. and Mikolov T. 2016. Fasttext.zip: Compressing text classification models. arXiv preprint arXiv:1612.03651.

19. Encyclopædia Britannica. John R. Firth. [Online] Available from: https://www.britannica.com/biography/John-R-Firth [Accessed: 03 February 2020].

20. Mikolov T., Sutskever I., Chen K., Corrado G.S. and Dean J. Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems*. 2013. P. 3111–3119.

21. Mikolov T., Yih W.T. and Zweig G. Linguistic regularities in continuous space word representations. In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 2013. P. 746–751.

22. Joulin A., Grave E., Bojanowski P. and Mikolov T. 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

23. Peters M.E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K. and Zettlemoyer L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

24. AllenNLP an open-source NLP research library. ELMo. [Online] Available from: https://allennlp.org/elmo [Accessed: 03 February 2020].

25. Gensim: Topic modelling for humans. [Online] Available from: https://radimrehurek.com/gensim [Accessed: 03 February 2020].

26. Luo Q. and Xu W. Learning word vectors efficiently using shared representations and document representations. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI Press*. 2015. P. 4180–4181.

27. Luo Q., Xu W. and Guo J. A Study on the CBOW Model's Overfitting and Stability. In: *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning. Association for Computing Machinery*. 2014. P. 9–12.

28. Mnih A. and Kavukcuoglu K. Learning word embeddings efficiently with noise-contrastive estimation. In: *Advances in neural information processing systems. Curran Associates Inc*. 2013. P. 2265–2273.

29. Srinivasa-Desikan B. (2018). Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras. *Packt Publishing Ltd*.

30. Nikolenko S., Kadurin A., and Arkhangel'skaya E. (2018) Glubokoe obuchenie. Pogruzhenie v mir neironnykh setei (Deep Learning: An Immersion in the World of Neural Networks). St. Petersburg: Piter.

31. Goyal P., Pandey S., & Jain K. (2018). Deep learning for natural language processing. Deep Learning for Natural Language Processing: Creating Neural Networks with Python. Berkeley, CA]: Apress.

32. Maynard D., Bontcheva K., & Augenstein I. (2017). Natural language processing for the semantic web. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers.

33. Wikimedia Downloads. [Online] Available from: https://dumps.wikimedia.org [Accessed: 03 February 2020].

34. spaCy. Industrial-strength Natural Language Processing in Python. [Online] Available from https://spacy.io [Accessed: 03 February 2020].

35. Natural Language Toolkit. NLTK 3.4.5. [Online] Available from https://www.nltk.org [Accessed: 03 February 2020].

36. StanfordNLP. Python NLP Library for Many Human Languages. [Online] Available from https://stanfordnlp.github.io/stanfordnlp [Accessed: 03 February 2020].

37. Gensim: API Reference. [Online] Available from https://radimrehurek.com/gensim/apiref.html [Accessed: 03 February 2020].

38. Le Q. and Mikolov T. Distributed representations of sentences and documents. In International conference on machine learning. 2014. P. 1188–1196.

39. Caselles-Dupré H., Lesaint F. and Royo-Letelier J. Word2vec applied to recommendation: Hyperparameters matter. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. 2018. P. 352–356.

40. Rong X. (2014). word2vec parameter learning explained. arXiv preprint arXiv:1411.2738.

41. AWS Machine Learning Blog. How to Deploy Deep Learning Models with AWS Lambda and Tensorflow. [Online] Available from https://aws.amazon.com/blogs/machine-learning/how-to-deploy-deep-learning-models-with-aws-lambda-and-tensorflow [Accessed: 03 February 2020].

42. Firefly. Firefly documentation. [Online] Available from https://rorodata.github.io/firefly [Accessed: 03 February 2020].

43. Common Crawl. [Online] Available from http://commoncrawl.org [Accessed: 03 February 2020].

44. Google Dataset Search. [Online] Available from https://datasetsearch.research.google.com [Accessed: 03 February 2020].

45. Vec2graph mini-library for producing graph visualizations from embedding models [Online] Available from: https://github.com/lizaku/vec2graph [Accessed: 03 February 2020].

46. Palagin A.V., Svitla S.JU., Petrenko M.G., Velychko V.JU. About one approach to analysis and understanding of the natural. *Computer means, networks and systems*. 2008. N 7. P. 128–137.

47. Gladun V.P. 1994. Processy formirovanija novyh znanij [Processes of formation of new knowledge]. Sofija: SD «Pedagog 6» – Sofia: ST «Teacher 6», 192 [in Russian].

48. Dobrov B., Loukachevitch N., Nevzorova O. 2003. The technology of new domains' ontologies development. *Proceedings of the X-th International Conference "Knowledge-Dialogue-Solution"* (KDS'2003). Varna, Bulgaria. 2003. P. 283–290.

49. Velychko V., Voloshin P., Svitla S. 2009. Avtomatizirovannoe sozdanie tezaurusa terminov predmetnoj oblasti dlja lokal'nyh poiskovyh sistem. International Book Series "Information Science & Computing". Book No: 15. Knowledge – Dialogue – Solution, Sofia, 2009. P. 24–31.

50. Palagin O.V., Velychko V.YU., Malakhov K.S., Shchurov O.S. (2018) Research and development workstation environment: the new class of current research information systems In: *Proceedings of the 11th International Conference of Programming UkrPROG 2018. CEUR Workshop Proceedings* 2139. Kyiv, Ukraine, May 22-24, 2018. [Online] Available from: http://ceur-ws.org/Vol-2139/255-269.pdf [Accessed: 03 February 2020].

About the authors:

Oleksandr Palagin,
Doctor of Sciences, Academician of National Academy of Sciences of Ukraine,
Deputy director of Glushkov Institute of Cybernetics,
head of department 205 at Glushkov Institute of Cybernetics.

Publications in Ukrainian journals – 290.
Publications in foreign journals – 45.
H-index: Google Scholar – 17,
Scopus – 3.
http://orcid.org/0000-0003-3223-1391,

Vitalii Velychko,
PhD, assistant professor, Senior researcher.
Publications in Ukrainian journals – 75.
Publications in foreign journals – 27.
H-index: Google Scholar – 11,
Scopus – 1.
http://orcid.org/0000-0002-7155-9202,

Kyrylo Malakhov,
Junior Research Fellow.
Publications in Ukrainian journals –32.
Publications in foreign journals – 3.
H-index: Google Scholar – 5.
http://orcid.org/0000-0003-3223-9844,

Oleksandr Shchurov,
1 category software engineer.
Publications in Ukrainian journals –6.
Publications in foreign journals – 1.
H-index: Google Scholar – 1.
http://orcid.org/0000-0002-0449-1295.

Authors' place of work:

V.M. Glushkov Institute of Cybernetics of National Academy of Sciences of Ukraine,
Akademician Glushkov Avenue, 40,
Kyiv, Ukraine, 03187.
Phone: (+38) (044) 526 3348.
E-mail: palagin_a@ukr.net