УДК 004.822
DOI: 10.25559/SITITO.15.201904.932-944

Knowledge Graph Essentials and Key Technologies

V. S. Gurin[1], E. V. Kostrov[1], Yu. Yu. Gavrilenko[2], D. F. Saada[2], E. A. Ilyushin[2*], I. V. Chizhov[2]
[1] Saint Petersburg State University, Saint Petersburg, Russia
7 Universitetskaya Emb., St Petersburg 199034, Russia
[2] Lomonosov Moscow State University, Moscow, Russia
1, Leninskie gory, Moscow 119991, Russia
[*] eugene.ilyushin@gmail.com

Abstract

In recent decades, the amount of information that humankind has accumulated has increased tremendously. People cannot analyze it effectively using simple algorithms, and data structures due to these approaches do not understand the semantics of the data. Thus, there is a need for such a data structure that would store a massive number of entities that would be accessible and easy to understand for machines, and moreover, saves the semantics. One efficient kind of structure is a knowledge graph, which quite recently appeared and became the subject of research for the last few years due to its interesting and sophisticated architecture. The peak of knowledge graph interest came at the time when Google introduced their Knowledge Graph technology in 2012, and since then, it has become apparent what this concept of a knowledge graph is. However, it is still unclear how to use this technology in practice due to the small number of existing information on this theme. In this paper, we introduce and review all steps of knowledge graph implementation. In addition to that, this article includes information about problems that need to be solved for having its instance of the knowledge graph; also, we consider machine learning embedding methods to analyze knowledge graph structure, practical steps for KG usage, and so on.

Keywords: Knowledge graph, Embedding, Semantic knowledge graph, Anthology.

For citation: Gurin V.S., Kostrov E.V., Gavrilenko Yu.Yu., Saada D.F., Ilyushin E.A., Chizhov I.V. Knowledge Graph Essentials and Key Technologies. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2019; 15(4):932-944. DOI: 10.25559/SITITO.15.201904.932-944

Представление знаний в виде графа: основные технологии и подходы

В. С. Гурин[1], Е. В. Костров[1], Ю. Ю. Гавриленко[2], Д. Ф. Саада[2], Е. А. Ильюшин[2*], И. В. Чижов[2]
[1] Санкт-Петербургский государственный университет, г. Санкт-Петербург, Россия
199034, Россия, г. Санкт-Петербург, Университетская наб., д. 7
[2] Московский государственный университет имени М.В. Ломоносова, г. Москва, Россия
119991, Россия, г. Москва, ГСП-1, Ленинские горы, д. 1
[*] eugene.ilyushin@gmail.com

Аннотация

В последние десятилетия объем накопленной человечеством информации увеличился невероятно. Люди не могут эффективно анализировать такой объем с помощью традиционных алгоритмов и структур данных из-за того, что они не позволяют использовать семантические связи. Таким образом, назрела необходимость в таком представлении информации, которое бы позволяло бы с одной стороны хранить огромное количество объектов и связей между ними, а с другой предоставляло высокоскоростной доступ к хранящимся данным, и, кроме того, сохраняло семантику. Одной из самых эффективных структур данных, позволяющей решать задачи подобного класса, является граф знаний, который относительно недавно появился и стал предметом исследований в последние годы. Пик интереса к графу знаний пришелся на то время, когда Google представил свою реализацию в 2012 году и стал использовать в своей поисковой машине, что значительно улучшило качество поиска. Однако до сих пор неясно, как воспользоваться данной технологией на практике из-за небольшого количества имеющейся информации по этой теме.
В этой статье мы рассматриваем все этапы реализации графа знаний, а также проблемы, с которыми возможно придется столкнуться при создании собственного экземпляра данной абстракции. Помимо этого, мы рассмотрим методы создания векторного представления информации для ее эффективного хранения в графе, а также практические шаги по его использованию.

Ключевые слова: граф знаний, семантический граф, онтологии.

Для цитирования: *Гурин В. С., Костров Е. В., Гавриленко Ю. Ю., Саада Д. Ф., Ильюшин Е. А., Чижов И. В.* Представление знаний в виде графа: основные технологии и подходы // Современные информационные технологии и ИТ-образование. 2019. Т. 15, № 4. С. 932-944. DOI: 10.25559/SITITO.15.201904.932-944

Introduction

A knowledge graph is often defined as a semantic graph consisting of nodes (vertices) and edges, where nodes represent concepts referring to the general categories of objects and edges represent the semantic relationships between entities. Some of the observed nodes can be connected to each other and form structured knowledge that can help to use, manage, and understand the information it contains. Knowledge graphs are primarly constructed from knowledge bases (KB), which refers to an intelligent database for managing, storing, and retrieving complex structured and unstructured information. Knowledge bases gather it from a free text on web pages, databases, audio, and video content. If Knowledge Graph is more about graphical structure, Knowledge Base focuses on data storage in the database. However, in most cases, they have the same meaning and similar approaches, though, in this article, we will use these two ideas as one general concept.

Due to the properties of the graph and its smart construction, it can be used to solve various problems, such as inferring conceptual meanings of user's web queries, building recommendations for users or other applications. Also, KGs can be served as a storage of structured knowledge which supports a large number of applications related to big data analytic

Nowadays there are lots of services, and sources are represented as knowledge graph structure or knowledge base: DBpedia [18], Freebase [14], YAGO [13], Wikidata , and WikiNet, Google KG [1], Facebook KG [14], Linkedin, but it is still unclear how to create and build this kind of structure and also what kind of technologies are inside. Knowledge graph creation could be separated into four main steps. Firstly, extracting useful information from different kind of sources to build a well- organized knowledge structure. Secondly, defining entities and relationships between them that represent knowledge graph using Nature Language Processing. The third step includes data integration. The final step includes graph structure analysis, link prediction cases, and using embedding methods based on machine learning and deep learning techniques.

In this paper, we will provide a review of all these steps, which includes semantic extraction methods, building knowledge base technique, machine learning embedding methods, and also some technical implementations. We hope our experience in building Knowledge Graph will be helpful for researches and enthusiasts in KG understanding and implementation.

Semantic Extraction

Entity Extraction

In this section, entity extraction methods from different sources will be observed. As we know, one of the most informative sources is Web pages that contain an enormously huge amount of structured and unstructured information for knowledge graph construction. The most well-known such kind of source is Wikipedia, which is used for a significant number of high-quality information extraction systems. There are lots of methods for extracting contents from Wikipedia pages that were proposed [13], [16], and which has become useful instruments in KG use cases.

In addition to that, lots of other high-quality web-resources include well- organized knowledge for entity collection. As for semistructured data resources, like template-based form, Wrapper Methods is one of the most often used solutions [20], [30]. This proposed methods automatically produce wrappers from a set of Web pages with high similarity.

There is also another effective method, where data is considered as hierarchical tree [44]. It is called Hierarchical Conditional Random Field, and it allows to optimize not only nodes detection but also attribute labeling. The main idea of this method is to calculate the maximum posterior probability value of y with given features X, then y^* will be computed by Eq.(1)

$$y^* = \arg\max p(y \mid X) \quad (1)$$

According to unstructured data, entity extraction can be implemented with named entity recognition (NER). It tries to detect and classify information elements from text and shows effectiveness in KG creation from unstructured data like texts. Some of Named Entity Recognition techniques are using machine learning methods and trained as predictive models for classification words into different entity types or the tags, which means the absence of belonging to any category.

Some papers [49] propose methods that use a chunk tagger which was based on Hidden Markov Model.

Other researchers provide their experience in using conditional random field (CRF) to train a sequential NE labeler [9].

In addition to that, [45] combines a K-nearest neighbors (KNN) classifier with a linear CRF model to perform NER for tweets.

Also, there is a paper that proposed Transfer Joint Embedding (TJE) method [32], based on transfer learning, for cross-domain classification of multiple classes.

[35], Prokofyev employs external sources (e.g., DBLP, Wikipedia, etc.) to improve the effectiveness of NER.

However, on top of all these approaches are systems that can read the Web effectively and provide relevant results in a short time. One of the latest methods is Never-Ending Language Learning. NELL [28] is a never-ending system that learns to read the Web. To extract triples in NELL, bootstrap constraints are used to learn new constraints. NELL has been running non-stop since January 2010, each day extracting more beliefs from the Web, then retraining itself to improve its competence. The result so far is a Knowledge Base (KB) with approximately 120mn interconnected beliefs, along with millions of learned phrasings, morphological features, and Web page structures NELL now uses to extract beliefs from the Web. NELL is also now learning to reason over its extracted knowledge to infer new beliefs that it has not yet read, and it is now able to propose extensions to its initial manually-provided ontology.

ReVerb [5] and OLLIE [38] are open information systems that extract a triple from a sentence by using syntactic and lexical patterns. Although these approaches successfully extract triples from unstructured text, they still do not consider entity mapping. As a result, the ambiguity of an extracted entity might occur.

Entity relations building

One of the graph features is that one entity can have different meanings in different sources, that is, relate to different parts of the text. KG goal is to try to correlate such textual mentions of entities with their nodes in the graph. For Example, the word "tesla" can associate with the scientist Nicola Tesla, and at the same time, it can mean the name of the company of Ilon Mask, called "Tesla." This task of linking entities is one of the most crucial tasks in building KG. The solution to this problem would help us to connect text information with a structure of a graph, which in turn contributes to the development of knowledge in the graph and its scaling.

To be more precise with the linking entities, if we found set X of

entities in textual source and set Y of entities in Knowledge Graph, what we want is to build a function.

$$f : x_i \to y_j, \text{for } x_i \in X, \textit{and } y_j \in Y \quad (2)$$

Then immediately, the question arises, what if we do not have this entity in the graph? According to the [40] it should be returned as NIL for the unlikable mention.

Several methods can cope with this nontrivial task. Some of these proposed approaches [40] use the context of mentions and features that are extracted from descriptive text associated with the entities, such as bag-of-words, entity popularity, etc. Unfortunately, these handcrafted features are useless, over-specified, and cannot accurately represent not only the mention but also semantic meanings of an entity. In this regard, methods using neural networks and applying word embedding have become popular in solving the problem of entity linking. The reason for this was their ability to capture distributed representation, which is necessary for this kind of task.

As a solution, Hoffart's model could be considered, which is a graph-based approach that finds a dense subgraph of entries in a document to address entity linking. This model is based on using a neural network for deriving entities representations and mention context to apply them entity linking.

Another approach for entity relations building is Yamada's model that jointly maps words and entities into the same continuous vector space and applies the embeddings to learn features for EL.

In addition to that, an effective method to cope with entity linking has been recently proposed. It is called a Deep Semantic Match Model (DSMM) [24] which helps to align a textual mention to the referent entity in a knowledge graph applying bidirectional Long Short Term Memory Network (BiLSTM) with multigranularities. This approach measures the match scores from two aspects: surface from the match, where a char-LSTM was applied to capture local representation, and semantic match, where a similar word-LSTM and TransE, a knowledge representation method, were used to learn the global representation of mention and entity respectively.

Given a mention x, the sentence s it occurs and Knowledge Graph denoted by a set of triples e_1, r, e_2, where $e_1, e_2 \in E$ are entities and $r \in R$ is a relation, DSMM aims to find the most relevant entity in candidate entities set $E_x \in E$. The peculiarity of this model is that it uses bidirectional LSTM that consists of two LSTMs with opposite directions to capture the context information on both sides better. Hence, at time step t, the hidden h_t can be referred to as the element-wise sum of the forward and backward pass.

The results of experiments on CoNLL benchmark dataset show that proposed DSMM outperforms previous state-of-the-art models.

Relation Extraction

KG is composed of many triples like $< subject, predicate, object >$, where the predicate is a semantic relationship between subject and object. TransE is first proposed by [3] to encode triples into a continuous low-dimensional space, which based on the translation $s + p \approx o$. Many follow-up works like TransH [43], [2], and TransR [29], proposed advanced methods of translation by introducing different embedding spaces.

Some recent works attempt to learn text and KG triples, including [15] and [7]. These models tend to strengthen the representation of entities and relationships for KG tasks, but not for text representation.

In order to properly construct a knowledge graph, between entities, the relationships must be defined in the correct way that can also be taken from text sources. This is what the techniques of extracting relations is capable of. One of the shared tasks in this field is to extract binary relations between entities. The following sentence can be used as an example: "Mark Cuban owns Dallas Mavericks", from which we can get the triple $< MarkCuban, own, DallasMavericks >$.

We can formulate relation extraction as follows: consider the sentence

$$s_1 s_2 \ldots o_1 \ldots s_i \ldots o_2 \ldots s_{n-1} s_n$$

Where o_1 and o_2 are two named objects and s_i relates to other words and predefined relation r, the learning algorithm tries to learn a function f:

$$f(S) = \begin{cases} +1, \text{ if } o_1 \text{ and } o_2 \text{ are related by relation } R, \\ 0, \text{ otherwise.} \end{cases} \quad (3)$$

In this equation f is a binary classifier, that can be not only Naive Bayes but also Perceptron and others. Extracted sentence features defined as S, but it also could be a structured sentence representation.

There are two approaches to relation extraction: feature-based and kernelbased method. According to the feature-based approach, it is mainly based on NLP techniques including part-of-speech tagging, syntax parsing, named entity recognition.

Based on the knowledge of previous works, there are several sets of features used for Relation Extraction: word features (headwords of entities, words or bigrams on the left, number of words separating the two entities, etc.), entity features (types of named entities (e.g., person, location) and their concatenation, mention levels (e.g., name, nominal, or pronoun), etc.), parse features (syntactic chunk sequence, path between the two entities in a parse tree, etc.)[17]

As for kernel-based methods, one of the examples is the string kernel [23]. The main idea of this kind of kernel includes comparison of textual information by the substring that they contain. In one of the other previous works [21], the sum of the similarity of left, middle, and right contexts is considered as a kernel. However, the main disadvantage of these approaches that they require a significant amount of data, that is labeled by a human, which takes a massive amount of time and manual resources for large-scale Web relation extraction.

Instead of feature-based and kernel-based approaches, few extraction relation methods, that is based on using textual patterns, were proposed [12].

Recent articles focus on deep learning networks due to error propagation during feature generating. More complicated models were proposed to learn deeper semantic features, like PCNN [42] and attention pooling CNN [33], graph LSTMs [2].

In addition to that approaches, a state-of-art model, called LFDS, was proposed.

First, we pre-train representations for entities and relations based on the translation law $s + p \approx o$ defined by typical KG embedding models such as TransE. Second, for each sentence in the train sets, we replace the entity mentions with the types of the entities in the KG. An attention mechanism is then applied to calculate the importance of words concerning the sentence pattern. Third, we train the sentence encoder by the margin loss between o, s, and sentence embeddings. Note we do not use the noisy relation labels to train the model. Finally, for prediction, we calculate the test sentences embedding, then compare the sentence embedding with all relation embeddings learned by TransE, and choose the closest relation as our predicted result.

However, other effective methods can cope with relation extraction task. They are based on word-embedding idea. Word-embedding can be defined by NLP methods where words or phrases represented as vectors in a low-dimensional space. Usually, for generating such maps deep learning, topic modeling, matrix factorization, and other techniques are used [3]

Neural-based representation learning methods encode KGs with low-dimensional vector representations of both entities and relations, which can be further used to extract unknown related facts. In addition to that, a weakly-supervised approach for extracting relations from textual sources was proposed. It is trained to use information from the text and from existing knowledge jointly [4].

Relation mentions, entities, and relations in this method are all embedded into a common low-dimensional vector space. A ranking-based embedding framework is used to train the model. For a triple (s, p, o), the model learns the plausibility of the triple by generalizing from the KG.

Another work that helps in solving the relation extraction task is based on features representation only as low-dimensional embeddings, which dramatically reduced the number of parameters [31].

Besides binary relations, there are also higher-order relations, which can be defined as $\langle o_1, o_2 \ldots o_n \rangle$, where o_i are entities with respect to the certain relation. This complex relationship can be constructed by binary relations unification according to Pereira F. work [48] Binary relations are first extracted using a classifier. Entities that have relations are linked in an entity graph, higher-order relations can be extracted by finding the maximum cliques in the graph.

Co-reference resolution

Before integrating triples into a knowledge graph, we need to receive components from coreference resolution, which aim is to detect co-referring chains of entities in unstructured text and then to group them. Since some texts include different kinds of entities expressions, pronouns, and abbreviations that refer to one entity, components of coreference resolution can be considered as essential parts in knowledge extraction. An entity and its various expressions can be grouped in such a way that the actions of identical entities in different expressions can be taken into account. One of the frameworks that can cope with this task are coreference solver from Stanford Core NLP [26] and AllenNLP[1].

CoreNLP uses a pipeline of tokenizer, part-of-speech tagger, named entity recognizer, syntactic parser, and coreference solver to annotate the unstructured text. In addition to coreference annotation, we store the named entity classification created by the pipeline. The named entity classes are used to filter named entity links having a conflicting ontology classification.

Triples Integration

The Triple Integration component aims to generate text triples by using outputs from the Entity Mapping component, the Coreference Resolution component, and the Triple Extraction component.

In the Triple Extraction component, we can extract relation triples from an unstructured text; however, entity mapping and coreference resolution among the entities of such triples are not performed. As a result, ambiguity in the triple occurs, and interlinking to entities in the KG is not established. Consequently, the transformation of a relation triple that conforms to the standard of KB is required. Therefore, to deal with such problems, the results from the three components are integrated and transformed by the following processes.

First, identical entities are grouped by using co-referring chains from the Coreference Resolution component. Second, a representative for the group of coreferring entities is selected by the voting algorithm. Because entities in the same group might have various representations, the majority of excluding pronouns in the group is chosen as the group representative. Third, all entities belonging to the group in the relation triples are replaced by the representative of its group. Fourth, the relation of this triple is straightforwardly transformed into a predicate by assigning a new URI. Finally, if an object of a relation triple is not an entity, it is left as literal. After performing these processes, text triples are extracted from unstructured text.

The Triple Integration component generates the text triple, e.g., $< DBpedia : MarkCuban, ex : own, DBpedia : DallasMavericks >$. However, the predicate of the triple, ex: own, is still not mapped to any predicate in the KG. However, the method proposed by [19] can handle this task. As shown in this papers, our ex: own could be mapped to DBpedia relation (for example $Property$), and the triple $< DBpedia : MarkCuban, DBpedia : Property, DBpedia : DallasMavericks >$ is created as a triple for the generated KG.

Embedding methods for link prediction

Knowledge graphs are being used in the field of machine learning for various applications, including question and answering, link prediction, fact-checking, entity disambiguation, etc. For many of these applications, a problem to find the missing relationships in the graph is essential to ensure, completeness, correctness, and quality. This involves the task of entity prediction and relationship prediction. A knowledge graph is a collection of entities and relationships between them in the form of RDF style triples (s, p, o) where s represents a head entity, o being the tail entity and p the relationship between the head and the tail entity. In this part, we will provide an overview of some methods that help to complete and populate Knowledge Graph.

Adversarial Network Embedding

In this paper [6], the strengths of generative adversarial networks in capturing latent features were provided, and its contribution to learning stable and robust graph representations was investigated. Specifically, an Adversarial Network Embedding (ANE) framework was proposed, which leverages the adversarial learning principle to regularize the representation learning. It consists of two components, i.e., a structure-preserving component and an adversarial learning component. The former component aims to capture network structural properties, while the latter contributes to learning robust representations by matching the posterior distribution of the latent representations to given priors.

Network embedding is a challenging research problem because of the high dimensionality, sparsity, and non-linearity of the graph data.

Though existing methods are effective in structure-preserving with different carefully designed objectives, they suffer from a lack of additional constraints for enhancing the robustness of the learned representations.

[1] AllenNLP - Demo [Electronic resource]. Available at: https://demo.allennlp.org/coreference-resolution (accessed 12.08.2019). (In Eng.)

struc2vec

Structural identity is a concept of symmetry in which network nodes are identified according to the network structure and their relationship to other nodes. This work presents *struc2vec* [36], a novel and flexible framework for learning latent representations for the structural identity of nodes. *struc2vec* uses a hierarchy to measure node similarity at different scales and constructs a multilayer graph to encode structural similarities and generate a structural context for nodes.

The labels of the nodes are not necessary, but their relations to other nodes (edges) are essential. The most common practical approaches to determine the structural identity of nodes are based on distances or recursions. While such approaches have advantages and disadvantages, an alternative methodology was provided, one based on unsupervised learning of representations for the structural identity of nodes. It is worth noting why recent approaches for learning node representations such as DeepWalk and *node2vec* succeed in classification tasks but tend to fail in structural equivalence tasks. The critical point is that many node features in most real networks exhibit strong homophily. Neighbors of nodes with a given feature are more likely to have the same feature.

Results indicate that while DeepWalk and *node2vec* fail to capture the notion of structural identity, *struc2vec* excels on this task - even when the first network is subject to intense random noise (random edge removal). It was also shown that *struc2vec* is superior in a classification task where node labels depend more on structural identity (i.e., air track networks with labels representing airport activity).

TransE [46]

Entity prediction is, given a s, p of a triple predict o and relationship prediction is, given s, p predict o the type of relationship between them. TransE [1] is an energy-based model that learns continuous, the low-dimensional embedding of entities and relationships by minimizing a margin-based pair-wise ranking loss. The idea behind TransE is, the relationship p is modeled as a translation between the head and the tail entity in the same low-dimensional plane. Therefore if a relationship (s, p, o) exists in the knowledge graph, the learnt embedding of $(s + p)$ should be closer to the embedding of o meaning $(s + p) \approx o$. If there is a negative or false triple such as (s', p, o') then the distance between the embedding of $(s' + p)$ and o' should be larger. The energy function $E(s, p, o)$ is defined to be the L_1 or L_2 distance between the $s + p$ (translated head entity) and o (tail entity). Now let us see how such embeddings can be learned from a training set S containing a set of positive (s, p, o) triples. Lets say the entity embedding and relationship embedding to be learnt is e and l, full list of entities E and relationships L, the training set $S \in (s, p, o)$ containing all positive triples, the first step is to initialise e and l with uniform embedding vectors for each entity in E and each relationship in L. Before we can go into the training phase, we also need negative samples in order to optimize a pair-wise ranking loss function. This is done by corrupting positive triples by either removing the head or tail and replacing it with a random entity from E. We call this the corrupted triples set $S' \in (s', p, o')$. Optimization happens in mini-batches. Therefore a mini-batch S_b is sampled from S of batch size b and for each positive triple in S_b, a negative triple is sampled from the corrupted triples set S'. Optimization happens now using stochastic gradient descent for each positive and negative triple pair in the mini-batch set, minimizing a margin-based ranking loss function given by,

$$L = \sum \max\left[0, \gamma + d(s + p, o) \mid d(s' + p, o')\right] \quad (4)$$

over the possible (s, p, o) embeddings. At each gradient step towards the minimum, the embeddings are updated with constant learning rate.

LINE [39]

This paper studies the problem of embedding extensive information networks into low-dimensional vector spaces, which is useful in many tasks such as visualization, node classification, and link prediction. In this paper, a novel network embedding method called the "LINE" was proposed, which is suitable for arbitrary types of information networks: undirected, directed, and/or weighted.

The method optimizes a carefully designed objective function that preserves both the local and global network structures. An edge-sampling algorithm is proposed that addresses the limitation of the classical stochastic gradient descent and improves both the effectiveness and the efficiency of the inference. Empirical experiments prove the effectiveness of the LINE on a variety of real-world information networks, including language networks, social networks, and citation networks. The algorithm is very efficient, which can learn the embedding of a network with millions of vertices and billions of edges in a few hours on a typical single machine.

This model makes use of latent variables and is capable of learning interpretable latent representations for undirected graphs. In addition to that, it uses a convolutional graph network (GCN) encoder and a simple inner product decoder.

The proposed model achieves competitive results on a link prediction task in citation networks. In contrast to most existing models for unsupervised learning on graph-structured data and link prediction, it can naturally incorporate node features, which significantly improves predictive performance on several benchmark datasets.

node2vec

This paper [10] propose *node2vec*, an algorithmic framework for learning continuous feature representations for nodes in networks. In *node2vec*, we learn a mapping of nodes to a low-dimensional space of features that maximizes the likelihood of preserving network neighborhoods of nodes. We define a flexible notion of a node's network neighborhood and design a biased random walk procedure, which efficiently explores diverse neighborhoods. The proposed algorithm generalizes prior work which is based on rigid notions of network neighborhoods, and we argue that the added flexibility in exploring neighborhoods is the key to learning richer representations.

Proposed *node2vec* is a semi-supervised algorithm for scalable feature learning in networks. Custom graph-based objective function was optimized using SGD. Intuitively, this approach returns feature representations that maximize the likelihood of preserving network neighborhoods of nodes in a *d*-dimensional feature space. A 2nd order random walk approach was used to generate (sample) network neighborhoods for nodes. A vital contribution is in defining a flexible notion of a node's network neighborhood. By choosing an appropriate notion of a neighborhood, *node2vec* can learn representations that organize nodes based on their network roles and/or communities they are belong. It was achieved by developing a family of biased random walks, which efficiently explore diverse neighborhoods of a given node. The resulting algorithm is flexible, giving control over the search space through tunable parameters.

Edge features. The *node2vec* algorithm provides a semi-supervised

method to learn rich feature representations for nodes in a network. However, we are often interested in prediction tasks involving pairs of nodes instead of individual nodes. For instance, in link prediction, we predict whether a link exists between two nodes in a network. Since proposed random walks are naturally based on the connectivity structure between nodes in the underlying network, they were extended to pairs of nodes using a bootstrapping approach over the feature representations of the individual node.

Logical queries

Learning low-dimensional embeddings of knowledge graphs is a powerful approach used to predict unobserved or missing edges between entities.

Here [11], graph nodes were embedded in a low-dimensional space and represent logical operators as learned geometric operations (e.g., translation, rotation) in this embedding space. By performing logical operations within a lowdimensional embedding space, a time complexity that is linear in the number of query variables was achieved, compared to the exponential complexity required by a naive enumeration-based approach.

One particularly useful set of graph queries and the focus of this work is conjunctive queries, which correspond to the subset of first-order logic using only the conjunction and existential quantification operators. Conjunctive queries allow one to reason about the existence of subgraph relationships between sets of nodes, which makes conjunctive queries a natural focus for knowledge graph applications.

Graph nodes are embed in a low-dimensional space and represent logical operators as learned geometric operations (e.g., translation, rotation) in this embedding space. After training, we can use the model to predict which nodes are likely to satisfy any valid conjunctive query, even if the query involves unobserved edges. Moreover, we can make this prediction efficiently, in time complexity that is linear in the number of edges in the query and constant concerning the size of the input network.

Autoencoder [41]

Similar to SDNE (Structural Deep Network Embedding), these models rely on the autoencoder to learn non-linear node embeddings from local graph neighborhoods.

The autoencoder model aims to learn a set of low dimensional latent variables for the nodes that can produce an approximate reconstruction output such that the error between adjacency matrix and output is minimized, thereby preserving the global graph structure. In many applications, only a small fraction of the nodes are labeled. For semi-supervised learning, it is advantageous to utilize unlabeled examples in conjunction with labeled instances to better capture the underlying data patterns for improved learning and generalization. Here it was achieved by training the autoencoder with a masked softmax classifier to collectively learn node labels from minimizing their combined losses.

The resulting models outperform related methods in accuracy performance on a range of real-world graph-structured datasets. The success of this models is primarily attributed to extensive parameter sharing between the encoder and decoder parts of the architecture, coupled with the capability to learn expressive non-linear latent node representations from both local graph neighborhoods and explicit node features. Further, this novel architecture is capable of simultaneous multi-task learning of both link prediction and node classification in one efficient end-to-end training stage.

DeepWalk [34]

Traditional approaches to relational classification pose the problem as inference in an undirected Markov network and then use iterative approximate inference algorithms (such as the iterative classification algorithm, Gibbs Sampling, or label relaxation) to compute the posterior distribution of labels given the network structure. We propose a different approach to capture the network topology information. Instead of mixing the label space as part of the feature space, we propose an unsupervised method which learns features that capture the graph structure independent of the labels' distribution.

DeepWalk is distance themselves from approximate inference techniques to leverage the dependency information by learning label-independent representations of the graph. The choice of labeled vertices does not influence Their representation quality so that they can be shared among tasks.

Learning Structural Node Embeddings via Diffusion Wavelets [8]

Learning structural representations of nodes is a challenging problem, and it has typically involved manually specifying and tailoring topological features for each node. GraphWave uses spectral graph wavelets to generate a structural embedding for each node, which we accomplish by treating the wavelets as a distribution and evaluating the resulting characteristic functions. Considering the wavelets as distributions instead of vectors is a crucial insight needed to capture structural similarity in graphs.

The proposed method provides mathematical guarantees on the optimality of learned structural embeddings. Using spectral graph theory, structurally equivalent (or similar) nodes have near-identical (or similar) embeddings in GraphWave. Various experiments on real and synthetic networks provide empirical evidence for analytical results and yield substantial gains in performance over state-of-the-art baselines.

Existing Knowledge Graph systems overview

There are lots of large graph systems that can provide excellent results and contain billions of entities. In this section we consider some of them and describe their characteristics

1. WordNet
2. NELL
3. Probase
4. Freebase
5. DBpedia
6. YAGO
7. Google knowledge graph
8. Facebook graph search

WordNet [27] was initially conceived as a lexical database for machine translation. Currently, WordNet is used as a semantic network and as an ontology. It contains 117 000 synsets, which are groups of synonyms corresponding to a concept. These synsets connect with each other through several semantic relations. WordNet has also been extended to a multilingual version, Multi- WordNet [45].

Never-ending language learner (NELL) [28] is a knowledge base implemented by the ReadTheWeb Project. It keeps populating a growing knowledge base of structured facts. Given an initial ontology containing 123 categories and 55 relations, it can extract 242

453 beliefs with an estimated precision of 74 percent in 67 days. So far, NELL has accumulated over 50 million candidate beliefs by reading the Web.

Probase [45] is a probabilistic knowledge base consisting of about 2.7 million concepts. The concepts are extracted from a corpus of 1.68 billion Web pages. To model inconsistent and uncertain data, it uses probabilistic models to build a probabilistic taxonomy. The goal of Probase is to understand human communication in the text using common-sense knowledge or general knowledge.

Freebase [14] is a graph-shaped database of structured, general human knowledge. It is a stable, practical platform for collecting knowledge by crowdsourcing. The current data in stored Freebase consists of 3.2 billions of triplets.

DBpedia [18] is a multilingual knowledge base in which the structured contents are extracted from Wikipedia. The structural knowledge in DBpedia is accumulated using crowdsourced techniques. DBpedia contains 24.9 million things in 119 languages, including 4.0 million in English.

YAGO/YAGO2 [13] is a huge semantic knowledge base in which the knowledge is extracted from Wikipedia and other sources. In 2014, it contained more than 10 million entities (e.g., persons, cites, organizations, etc.) and more than 120 million facts about these entities.

Google knowledge graph (GKG) [1] is a knowledge base used by Google to add semantic search functionality to its search engine. Google's search engine provides structural information about the topic inferred from the user's query using GKG. The KG has compiled more than 3.5 billion facts over 500 million objects or entities.

Facebook graph search [14] provides semantic search service by Facebook. It combines data acquired from over one billion users and external data to provide user-specific search results. Users can search for pages, people, places, check-ins, etc. using natural languages.

Knowledge Graph storages

1. **Neo4j** [37] is an open-source, embedded, disk-based graph database implemented in Java, that uses LPG (Labeled Property Graph) model. It is highly scalable and fully supports the properties of atomicity, consistency, isolation, and durability (ACID). As a network-oriented database, it can extend to several clusters to process billions of nodes in parallel. Neo4j has been applied in mission-critical applications.
2. **DEX** [22] is also a high-performance, scalable graph DBMS implemented in Java and C++. In DEX, A DEX graph is a labeled directed attributed multigraph (LDAM), making it suitable for storage of complex graph structures. The transactions in DEX support aCiD, meaning full consistency and durability support with partial isolation and atomicity.
3. **Cayley** [47] is an open-source graph inspired by the graph database behind Freebase and Google's Knowledge Graph. Its goal is to be a part of the developer's toolbox, where Linked Data and graph-shaped data (semantic webs, social networks, etc.) in general are concerned.
4. **Dgraph**[2] is a high-scalable, low-latency, and high-throughput distributed graph database. It emphasizes concurrency in distributed environment by minimizing network calls.
5. In July 2015, Manish Rai Jain created Dgraph based on his previous experience at Google. There he led a project to unite all data structures for serving web search with a backend graph system. The first version v0.1 was released on December 2015, with the goal offering an open-source, native, and distributed graph database never changes since then.
6. Dgraph's primary focus is low latency and high throughput. It references the design of Google's Bigtable and Facebook's Tao and achieves high scalability at the cost of lack of full ACID-compliant transactional support. Also, value data versioning is under consideration, and not yet implemented
7. One Map/Reduce platform that is worth mentioning is **Hadoop**[3]. Hadoop allows for distributed processing of massive data sets across computer clusters. It offers reliable and scalable computation for offline data processing but is not readily suitable for graphs. Inspired by the Map/Reduce framework, Malewicz et al. [25] propose Pregel.
8. In this system, programs are treated as a sequence of iterations called super steps. In each super step, each vertex can receive messages sent in the previous iteration, compute a specific function in parallel and send to other vertices. There are other graph database implementations on top of Pregel, including Phoebus[4] and Giraph[5], in order to take advantage of the Map/Reduce framework.
9. Besides the Hadoop's Map/Reduce framework, other graph databases are using distributed storage as well. InfiniteGraph [45] is a distributed-oriented system that combines the strengths of persisting and traversing complex relationships requiring multiple hops.

In conclusion, there are various graph databases nowadays available or under development, most of which are application-driven. Since there is no standard graph data model, database system, or query language, the choice of graph databases is based on its applications.

Importing Knowledge Graph

A Knowledge Graph is an exciting concept as it is, but to become useful, it should be loaded into some database. The largest Knowledge Graph available for the download is Freebase, so it was chosen as a target dataset for testing import.

Importing into Cayley

Cayley provides an ability to choose between several backends for storing the graph, so the set of experiments was done to test the fastest way to import the data. In the following table list of used backends and options along with corresponding time in minutes are presented. For example, it took 10 minutes to import 25 million of quads to Bolt backend from .pq file, with 1.25M-sized batches.

[2] Dgraph [Electronic resource]. Available at: https://dgraph.io/ (accessed 12.08.2019). (In Eng.)

[3] Hadoop [Electronic resource]. Available at: www.hadoop.org (accessed 12.08.2019). (In Eng.)

[4] Malewicz G., Austern M.H., Bik A.J., Dehnert J.C., Horn I., Leiser N., Czajkowski G. Pregel: a system for large-scale graph processing - "ABSTRACT."
In: Proceedings of the 28th ACM symposium on Principles of distributed computing (PODC '09). Association for Computing Machinery, New York, NY, USA, 2009, 6 pp. (In Eng.) DOI: 10.1145/1582716.1582723

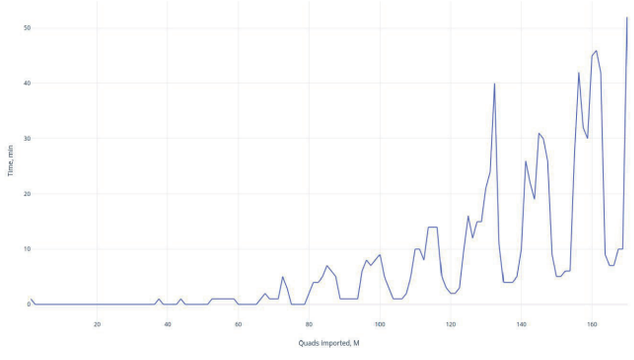
[5] Gigraph – Network Visualization for Excel [Electronic resource]. Available at: https://gigraph.io/ (accessed 12.08.2019). (In Eng.)

Table 1. Results of testing importing into Cayley different backends and different options

	5M	10M	15M	20M	25M
Bolt + nq + 10k	2	7	16	20	29
Bolt + nq + 50k	2	5	13	15	23
Bolt + nq + 100k	2	5	12	14	20
Bolt + nq + 200k	2	5	10	12	17
Bolt + nq + 500k	2	5	8	10	13
Bolt + nq + 500k	2	5	8	9	12
Bolt + nq + 1.25M	2	5	7	9	12
Bolt + nq + 2.5M	2	5	7	9	12
Bolt + nq + 5M	3	5	8	9	12
Bolt + nq + 1M + nosync	2	5	7	9	12
Bolt + nq + 2.5M + nosync	2	5	7	9	11
Bolt + pq + 1.25M	2	4	7	8	10
Leveldb + nq + buffer 20 + 10k	4	12	26	-	-
Leveldb + pq.gz + buffer 20M + 1.25M	1	8	16	18	27
Leveldb + nq + buffer 20M + 5M	2	18	-	-	-
Leveldb + pq.gz + buffer 200M + 1.25M	1	8	16	18	27
Leveldb + pq.gz + buffer 1G + 1.25M	1	8	16	18	27
Leveldb + pq.gz + buffer 1G + 500k	1	5	11	13	19
Leveldb + pq.gz + buffer 4G + 500k	1	5	11	13	19
Leveldb + pq.gz + buffer 4G + 1.25M	1	8	16	18	27
Leveldb + pq.gz + buffer 4G + cache 200M + 1.25M	1	8	16	18	28

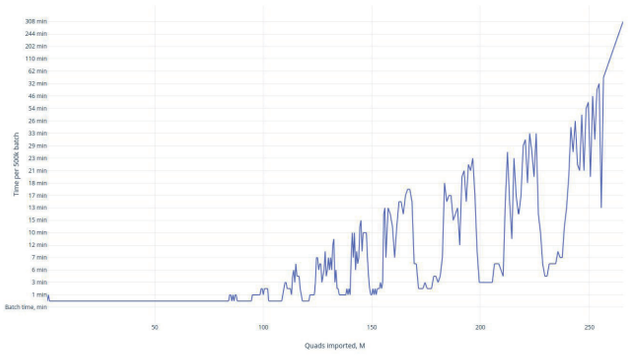
Table legend:
Bolt, Leveldb - key-value backends
nq - raw RDF file, pq - Cayley-specific binary format, *.gz - gzipped version of a file *nosync*
disable syncing to disk per transaction (for Bolt)
buffer - LevelDB write cache size
cache -x LevelDB block cache size
10k...5M - load batch size

Loading to Bolt from pq file and 1.25M batch size appeared to be the fastest way. After the load process was started, the cyclic performance degradation was observed. It means that loading each additional batch takes longer than loading the previous one (Fig. 1).



F i g. 1. Performance degradation visualization for batches of 1.25M quads

After that, the batch size was decreased to 500k and ”nosync” mode was added. It made things a bit better, yet we still were not able to import Freebase completely (Fig. 2).



F i g. 2. Performance degradation visualization for batches of 500k quads

By this moment it became evident that the current version of Cayley is not capable of importing the Freebase completely.

Importing to Dgraph

Dgraph uses backend named ”Badger” (Cayley had it as backend as well, though it was in the test stage). There were no performance issues with Dgraph, yet it had problems of another kind:

1. Dgraph requires schema for subjects of certain types. In the case of Freebase, we need to add schema for triples, where the subject is a literal with language tag e.g., “Association”@en.
2. Another issue is schema-related as well. Dgraph requires subjects to be either literal or UID. In Freebase, there are triples, with

<http://.../user.xandr.webscrapper.domain.ad_entry.ads_topic>
predicate, where some subjects are UID while others are literals. Such triples were deleted from the dataset to match Dgraph rules.

3. The main issue with Dgraph is that it uses RFC 3339[6] for dates parsing while Freebase has lots of different date/time formats. Here are some of them:

T00
T01:00
T10:00Z
T10:30:30 2001-10-13
1810
-0410
-0099-12 -0216-06-22
2014-05 1988-06-29T02 2010-06-24T16:00
2007-06-19T12:24Z 2007-10-09T20:22:05
2006-05-29T03:00:00Z 1986-03-05T09:03+01:00
2007-09-24T00:39:42.45Z 1975-05-15T22:00:00.000Z
2011-03-26T06:34:55.0000Z 2007-01-24T06:18:03.046839
2007-03-20T07:05:01.913933Z

Some of these formats (like -0410, 2014-05, T10:00Z) aren’t compatible with standard mentioned above, so the date/datetime strings were converted to Unix time. Conversion algorithm looks like this:

[6] Newman C., Klyne G. Date and Time on the Internet: Timestamps. RFC 3339, July 2002. (In Eng.) DOI: 10.17487/RFC3339

a) The default date is set to 0001-01-01-T00:00:00.000000Z
b) If date string is missing some parts - missing parts are taken from the default date. Example - "2014-05" is considered to be 2014-05-01T00:00:00.000000Z
c) If the date is earlier than 4799BC, "jyear" format is used. The reason for this is
that algorithm, that used for Gregorian to Julian calendar does not work with dates before 4799[7] January (according to Standards of Fundamental Astronomy[8]).
d)Conversion to Unix time. Timestamps like T01:30 are converted to seconds since
00:00 (e.g., T01:30 becomes 5400).
After performing all steps, we will receive a prepared RDF file along with schema sufficient to import the Freebase into Dgraph. Corresponding code can be found in a separate repo[9].

Conclusion

As we can see, Knowledge Graph has already become an efficient instrument in different tasks as recommendations or intelectual searching due to its complex structure and huge amount of stored semantic information. But it is still hard to understand where to start to build such architecture, and what technologies are inside it. Nevertheless, Knowledge Graph has proven itself as a promising technology that will help in solving different Natural Language Processing problems. In this paper, we provided all the steps for creating a Knowledge Graph from Entity Extraction to Triple Integration and also presented a brief introduction to some useful embedding methods in link prediction task. In addition to that, different data storages with existing knowledge bases that can help in KG implementing were described. Not all details were described in this article, but this review can be a starting point in the creation of this kind of structure.

References

[1] The Google Knowledge Graph: Information gatekeeper or a force to be reckoned with? *Strategic Direction.* 2014; 30(4):15-17. (In Eng.) DOI: 10.1108/SD-04-2014-0049

[2] Bian J., Gao B., Liu T.Y. Knowledge-Powered Deep Learning for Word Embedding. In: Calders T., Esposito F., Hüllermeier E., Meo R. (Eds.) *Machine Learning and Knowledge Discovery in Databases.* ECML PKDD 2014. Lecture Notes in Computer Science, vol. 8724. Springer, Berlin, Heidelberg, 2014, pp. 132-148. (In Eng.) DOI: 10.1007/978-3-662-44848-9_9

[3] Bordes A., Glorot X., Weston J., Bengio Y. A semantic matching energy function for learning with multi-relational data: Application to word-sense disambiguation. *Machine Learning*. 2014; 94(2):233-259. (In Eng.) DOI: 10.1007/s10994-013-5363-6

[4] Chen T., Dredze M., Weiner J.P., Hernandez L., Kimura J., Kharrazi H. Extraction of Geriatric Syndromes From Electronic Health Record Clinical Notes: Assessment of Statistical Natural Language Processing Methods. *JMIR Medical Informatics*. 2019; 7(1):e13039. (In Eng.) DOI: 10.2196/13039

[5] Culotta A., Sorensen J. Dependency Tree Kernels for Relation Extraction. In: *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04.* Association for Computational Linguistics, 2004, pp. 423-es. (In Eng.) DOI: 10.3115/1218955.1219009

[6] Dai Q., Li Q., Tang J., Wang D. Adversarial Network Embedding. *arXiv.* 2017. Available at: http://arxiv.org/abs/1711.07838 (accessed 12.08.2019). (In Eng.)

[7] Ding J., Ma S., Jia W., Guo M. Jointly Modeling Structural and Textual Representation for Knowledge Graph Completion in Zero-Shot Scenario. In: Cai Y., Ishikawa Y., Xu J. (Eds) *Web and Big Data. APWeb-WAIM 2018.* Lecture Notes in Computer Science, vol. 10987. Springer, Cham, 2018, pp. 369-384. (In Eng.) DOI: 10.1007/978-3-319-96890-2_31

[8] Donnat C., Zitnik M., Hallac D., Leskovec J. Learning Structural Node Embeddings via Diffusion Wavelets. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18).* ACM Press, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1320-1329. (In Eng.) DOI: 10.1145/3219819.3220025

[9] Finkel J.R., Grenager T., Manning C. Incorporating Non-Local Information into Information Extraction Systems by Gibbs Sampling. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05).* Association for Computational Linguistics, USA, 2005, pp. 363-370. (In Eng.) DOI: 10.3115/1219840.1219885

[10] Andrieu C., de Freitas N., Doucet A., Jordan M.I. An Introduction to MCMC for Machine Learning. *Machine Learning.* 2003; 50:5-43. (In Eng.) DOI: 10.1023/A:1020281327116

[11] Grover A., Leskovec J. Node2vec: Scalable Feature Learning for Networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16).* Association for Computing Machinery, New York, NY, USA, 2016, pp. 855-864. (In Eng.) DOI: 10.1145/2939672.2939754

[12] Hamilton W.L., Bajaj P., Zitnik M., Jurafsky D., Leskovec J. Embedding logical queries on knowledge graphs. *arXiv:*1806.01445 [cs.SI]. 2018. Available at: http://arxiv.org/abs/1806.01445 (accessed 12.08.2019). (In Eng.)

[13] Hearst M.A. Automatic Acquisition of Hyponyms from Large Text Corpora. In: *Proceedings of the 14th conference on Computational linguistics - Volume 2 (COLING '92).* Association for Computational Linguistics, USA, 1992, pp. 539-545. (In Eng.) DOI: 10.3115/992133.992154

[14] Hoffart J., Suchanek F.M., Berberich K., Weikum G. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence.* 2013; 194; 28-61. (In Eng.) DOI: 10.1016/j.artint.2012.06.

[15] Hogan M. Facebook Data Storage Centers as the Archive's Underbelly. *Television & New Media.* 2013; 16(1):3-18. (In Eng.) DOI: 10.1177/1527476413509415

[16] Huang X., Zhang J., Li D., Li P. Knowledge Graph Embedding

[7] Smart W.M. Spherical Astronomy, Cambridge University Press, 6th edition. Green, 1977, p. 49. (In Eng.)

[8] Standards of Fundamental Astronomy [Electronic resource]. Available at: http://www.iausofa.org (accessed 12.08.2019). (In Eng.)

[9] Bollacker K., Evans C., Paritosh P., Sturge T., Taylor J. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In:Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08). Association for Computing Machinery, New York, NY, USA, 2008, pp. 1247-1250. DOI: 10.1145/1376616.1376746

Based Question Answering. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19).* Association for Computing Machinery, New York, NY, USA, 2019, pp. 105-113. (In Eng.) DOI: 10.1145/3289600.3290956

[17] Lehmann J., Isele R., Jakob M., Jentzsch A., Kontokostas D., Mendes P.N., Hellmann S., Morsey M., van Kleef P., Auer S., Bizer C. DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web.* 2015; 6(2):167-195. (In Eng.) DOI: 10.3233/SW-140134

[18] Kambhatla N. Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations. In: *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions.* Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 22-es. (In Eng.) DOI: 10.3115/1219044.1219066

[19] Karsai L., Fekete A., Kay J., Missier P. Clustering Provenance Facilitating Provenance Exploration through Data Abstraction. In: *Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA'16)*. Association for Computing Machinery, New York, NY, USA, 2016, Article 6, pp. 1-5. (In Eng.) DOI: 10.1145/2939502.2939508

[20] Kertkeidkachorn N., Ichise R. T2KG: An End-to-End System for Creating Knowledge Graph from Unstructured Text. In: *AAAI-17 Workshop on Knowledge-Based Techniques for Problem Solving and Reasoning WS-17-12,* vol. WS-17. Association for the Advancement of Artificial Intelligence, 2017. (In Eng.) Available at: https://aaai.org/ocs/index.php/WS/AAAIW17/paper/view/15129 (accessed 12.08.2019). (In Eng.)

[21] Kushmerick N. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*. 2000; 118(1):15-68. (In Eng.) DOI: 10.1016/S0004-3702(99)00100-9

[22] Lin C.Y., Xue N., Zhao D., Huang X., Feng Y. (Eds.): *Natural Language Understanding and Intelligent Applications: 5th CCF Conference on Natural Language Processing and Chinese Computing, NLPCC 2016, and 24th International Conference on Computer Processing of Oriental Languages, ICCPOL 2016, Kunming, China, December 2-6, 2016, Proceedings*, Lecture Notes in Computer Science, vol. 10102. Springer International Publishing, Cham, 2016. (In Eng.) DOI: 10.1007/978-3-319-50496-4

[23] Lissandrini M., Brugnara M., Velegrakis Y. Beyond Macrobenchmarks: Microbenchmark-Based Graph Database Evaluation. *Proceedings of the VLDB Endowment*. 2018; 12(4):390-403. (In Eng.) DOI: 10.14778/3297753.3297759

[24] Luo A., Gao S., Xu Y. Deep Semantic Match Model for Entity Linking Using Knowledge Graph and Text. *Procedia Computer Science*. 2018; 129:110-114. (In Eng.) DOI: 10.1016/j.procs.2018.03.057

[25] Malewicz G., Austern M.H., Bik A.J., Dehnert J.C., Horn I., Leiser N., Czajkowski G. Pregel: A System for Large-Scale Graph Processing. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data.* Association for Computing Machinery, New York, NY, USA, 2010, pp. 135-146. (In Eng.) DOI: 10.1145/1807167.1807184

[26] Manning C., Surdeanu M., Bauer J., Finkel J., Bethard S., McClosky D. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, 2014, pp. 55-60. (In Eng.) DOI: 10.3115/v1/P14-5010

[27] Miller G.A. WordNet: A Lexical Database for English. *Communications of the ACM*. 1995; 38(11):39-41. (In Eng.) DOI: 10.1145/219717.219748

[28] Mitchell T., Cohen W., Hruschka E., Talukdar P., Yang B., Betteridge J., Carlson A., Dalvi B., Gardner M., Kisiel B., Krishnamurthy J., Lao N., Mazaitis K., Mohamed T., Nakashole N., Platanios E., Ritter A., Samadi M., Settles B., Wang R., Wijaya D., Gupta A., Chen X., Saparov A., Greaves M., Welling J. Never-Ending Learning. *Communications of the ACM*. 2018; 61(5):103-115. (In Eng.) DOI: 10.1145/3191513

[29] Moon C., Jones P., Samatova N.F. Learning Entity Type Embeddings for Knowledge Graph Completion. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17).* Association for Computing Machinery, New York, NY, USA, 2017, pp. 2215-2218. (In Eng.) DOI: 10.1145/3132847.3133095

[30] Muslea I., Minton S., Knoblock C.A. Hierarchical Wrapper Induction for Semistructured Information Sources. *Autonomous Agents and Multi-Agent Systems*. 2001; **4**(1):93-114. (In Eng.) DOI: 10.1023/A:1010022931168

[31] Nguyen N.T., Miwa M., Tsuruoka Y., Chikayama T., Tojo S. Wide-coverage relation extraction from MEDLINE using deep syntax. *BMC Bioinformatics.* 2015; **16**(1):107. (In Eng.) DOI: 10.1186/s12859-015-0538-8

[32] Pan S.J., Toh Z., Su J. Transfer Joint Embedding for Cross-Domain Named Entity Recognition. *ACM Transactions on Information Systems.* 2013; **31**(2):1-27. (In Eng.) DOI: 10.1145/2457465.2457467

[33] Peng N., Poon H., Quirk C., Toutanova K., Yih W-t. Cross-Sentence *N*-ary Relation Extraction with Graph LSTMs. *Transactions of the Association for Computational Linguistics*. 2017; 5:101-115. (In Eng.) DOI: 10.1162/tacl_a_00049

[34] Perozzi B., Al-Rfou R., Skiena S. DeepWalk: Online Learning of Social Representations. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '14).* Association for Computing Machinery, New York, NY, USA, 2014, pp. 701-710. (In Eng.) DOI: 10.1145/2623330.2623732

[35] Prokofyev R., Demartini G., Cudr´e-Mauroux P. Effective Named Entity Recognition for Idiosyncratic Web Collections. In: *Proceedings of the 23rd international conference on World wide web (WWW '14).* Association for Computing Machinery, New York, NY, USA, 2014; 397-408. (In Eng.) DOI: 10.1145/2566486.2568013

[36] Ribeiro L.F.R., Saverese P.H.P., Figueiredo D.R. Struc2vec: Learning Node Representations from Structural Identity. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17).* Association for Computing Machinery, New York, NY, USA, 2017, pp. 385-394. (In Eng.) DOI: 10.1145/3097983.3098061

[37] Rodriguez M.A., Neubauer P. Constructions from dots and lines. *Bulletin of the American Society for Information Science and Technology.* 2010; **36**(6):35-41. (In Eng.) DOI: 10.1002/bult.2010.1720360610

[38] Rodríguez J.M., Merlino H.D., Pesado P., García-Martínez R. Performance Evaluation of Knowledge Extraction Methods. In: Fujita H., Ali M., Selamat A., Sasaki J., Kurematsu M.

(Eds.) *Trends in Applied Knowledge-Based Systems and Data Science. IEA/AIE 2016*. Lecture Notes in Computer Science, vol. 9799. Springer, Cham, 2016, pp. 16-22. (In Eng.) DOI: 10.1007/978-3-319-42007-3_2

[39] Tang J., Qu M., Wang M., Zhang M., Yan J., Mei Q. LINE: Large-scale Information Network Embedding. In: *Proceedings of the 24th International Conference on World Wide Web (WWW '15).* International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2015, pp. 1067-1077. (In Eng.) DOI: 10.1145/2736277.2741093

[40] Tianlei Z., Xinyu Z., Mu G. KeEL: knowledge enhanced entity linking in automatic biography construction. *The Journal of China Universities of Posts and Telecommunications*. 2015; 22(1):57-64, 71. (In Eng.) DOI: 10.1016/S1005-8885(15)60625-2

[41] Tran P.V. Learning to Make Predictions on Graphs with Autoencoders. In: *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, Turin, Italy, 2018, pp. 237-245. (In Eng.) DOI: 10.1109/DSAA.2018.00034

[42] Wang L., Cao Z., de Melo G., Liu Z. Relation Classification via Multi-Level Attention CNNs. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers). Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1298-1307. (In Eng.) DOI: 10.18653/v1/P16-1123

[43] Wang Z., Zhang J., Feng J., Chen Z. Knowledge Graph and Text Jointly Embedding. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1591-1601. (In Eng.) DOI: 10.3115/v1/D14-1167

[44] Wu Y.C., Fan T.K., Lee Y.S., Yen S.J. Extracting Named Entities Using Support Vector Machines. In: Bremer E.G., Hakenberg J., Han EH., Berrar D., Dubitzky W. (Eds.) *Knowledge Discovery in Life Science Literature. KDLL 2006.* Lecture Notes in Computer Science, vol. 3886. Springer, Berlin, Heidelberg, 2006, pp. 91-103. (In Eng.) DOI: 10.1007/11683568_8

[45] Yan J., Wang C., Cheng W., Gao M., Zhou A. A retrospective of knowledge graphs. *Frontiers of Computer Science.* 2018; 12(1):55-74. (In Eng.) DOI: 10.1007/s11704-016-5228-9

[46] Zhang D., Li M., Jia Y., Wang Y., Cheng X. Efficient Parallel Translating Embedding for Knowledge Graphs. In: *Proceedings of the International Conference on Web Intelligence (WI '17).* Association for Computing Machinery, New York, NY, USA, 2017, pp. 460-468. (In Eng.) DOI: 10.1145/3106426.3106447

[47] Zhao S-L., Hao R-X., Stewart I. The Generalized Three-Connectivity of Two Kinds of Cayley Graphs. *The Computer Journal*. 2019; 62(1):144-149. (In Eng.) DOI: 10.1093/computer_journal/bxy054

[48] Zhou D., Zhong D., He Y. Biomedical Relation Extraction: From Binary to Complex. *Computational and Mathematical Methods in Medicine.* 2014; 2014:298473. 18 pp. (In Eng.) DOI: 10.1155/2014/298473

[49] Zhou G.D., Su J. Named entity recognition using an HMM-based chunk tagger. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02).* Association for Computational Linguistics, USA, 2002, pp. 473-480. (In Eng.) DOI: 10.3115/1073083.1073163

Submitted 12.08.2019; revised 15.11.2019; published online 23.12.2019.
Поступила 12.08.2019; принята к публикации 15.11.2019; опубликована онлайн 23.12.2019.

About the authors:

Vladislav Gurin, Postgraduate student of the Mathematics and Mechanics Faculty, Researcher, Saint Petersburg State University (7 Universitetskaya Emb., St Petersburg 199034, Russia), ORCID: http://orcid.org/0000-0002-2996-6226, vlad.gurin@gmail.com

Eugene Kostrov, Researcher, Saint Petersburg State University (7 Universitetskaya Emb., St Petersburg 199034, Russia), ORCID: http://orcid.org/0000-0001-9402-5337, kostrov.e@gmail.com

Yuliya Yu. Gavrilenko, Master's degree student of the Faculty of Cosmic Research, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia), ORCID: http://orcid.org/0000-0002-8704-6030, gavrilenko.yulia@bk.ru

Daniel F. Saada, Master's degree student of the Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia), ORCID: http://orcid.org/0000-0003-4959-8093, daniel.saada@gmail.com

Eugene A. Ilyushin, Postgraduate student, Senior Software Developer of the Laboratory of Open Information Technologies, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia), ORCID: http://orcid.org/0000-0002-9891-8658, eugene.ilyushin@gmail.com

Ivan V. Chizhov, Associate Professor of the Department of Information Security, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia), Ph.D. (Phys.-Math.), ORCID: http://orcid.org/0000-0001-9126-6442, ichizhov@cs.msu.ru

All authors have read and approved the final manuscript.

Об авторах:

Гурин Владислав Сергеевич, аспирант математико-механического факультета, исследователь, Санкт-Петербургский государственный университет (199034, Россия, г. Санкт-Петербург, Университетская наб., д. 7), ORCID: http://orcid.org/0000-0002-2996-6226, vlad.gurin@gmail.com

Костров Евгений Викторович, исследователь, Санкт-Петербургский государственный университет (199034, Россия, г. Санкт-Петербург, Университетская наб., д. 7), ORCID: http://orcid.org/0000-0001-9402-5337, kostrov.e@gmail.com

Гавриленко Юлия Юрьевна, магистрант факультета космических исследований, Московский государственный университет имени М.В. Ломоносова (119991, Россия, г. Москва, Ленинские горы, д. 1), ORCID: http://orcid.org/0000-0002-8704-6030, gavrilenko.yulia@bk.ru

Саада Даниель Фирасович, магистрант факультета вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова (119991, Россия, г. Москва, Ленинские горы, д. 1), ORCID: http://orcid.org/0000-0003-4959-8093, daniel.saada@mail.ru

Ильюшин Евгений Альбинович, аспирант, ведущий программист лаборатории открытых информационных технологий, факультет вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоно-

сова (119991, Россия, г. Москва, Ленинские горы, д. 1), ORCID: http://orcid.org/0000-0002-9891-8658, eugene.ilyushin@gmail.com

Чижов Иван Владимирович, доцент кафедры информационной безопасности, факультет вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова (119991, Россия, г. Москва, Ленинские горы, д. 1), кандидат физико-математических наук, ORCID: http://orcid.org/0000-0001-9126-6442, ichizhov@cs.msu.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.