Semantic Web 13 (2022) 441–475
DOI 10.3233/SW-212955
IOS Press

Taxonomy enrichment with text and graph vector representations

Irina Nikishina [a,*,**], Mikhail Tikhomirov [b], Varvara Logacheva [a], Yuriy Nazarov [b], Alexander Panchenko [a] and Natalia Loukachevitch [b]

[a] *Skolkovo Institute of Science and Technology, Moscow, Russia*
E-mails: irina.nikishina@skoltech.ru, v.logacheva@skoltech.ru, a.panchenko@skoltech.ru
[b] *Research Computing Center, Lomonosov Moscow State University, Moscow, Russia*
E-mails: tikhomirov.mm@gmail.com, nazarov.yuriy.pavlovich@gmail.com, louk_nat@mail.ru

Editors: Mehwish Alam, FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, Germany; Davide Buscaldi, LIPN, Université Sorbonne Paris Nord, France; Michael Cochez, Vrije University of Amsterdam, the Netherlands; Francesco Osborne, Knowledge Media Institute, (KMi), and The Open University, UK; Diego Reforgiato Recupero, University of Cagliari, Italy; Harald Sack, FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, Germany
Solicited review: six anonymous reviewers

Abstract. Knowledge graphs such as DBpedia, Freebase or Wikidata always contain a taxonomic backbone that allows the arrangement and structuring of various concepts in accordance with hypo-hypernym ("class-subclass") relationship. With the rapid growth of lexical resources for specific domains, the problem of automatic extension of the existing knowledge bases with new words is becoming more and more widespread. In this paper, we address the problem of *taxonomy enrichment* which aims at adding new words to the existing taxonomy.

We present a new method which allows achieving high results on this task with little effort. It uses the resources which exist for the majority of languages, making the method universal. We extend our method by incorporating deep representations of graph structures like node2vec, Poincaré embeddings, GCN etc. that have recently demonstrated promising results on various NLP tasks. Furthermore, combining these representations with word embeddings allows us to beat the state of the art.

We conduct a comprehensive study of the existing approaches to taxonomy enrichment based on word and graph vector representations and their fusion approaches. We also explore the ways of using deep learning architectures to extend taxonomic backbones of knowledge graphs. We create a number of datasets for taxonomy extension for English and Russian. We achieve state-of-the-art results across different datasets and provide an in-depth error analysis of mistakes.

Keywords: Taxonomy enrichment, graph vector representations, word embeddings, graph convolutional auto-encoder

1. Introduction

The central idea of Semantic Web is to make the content of the Internet pages machine-interpretable. For that, web pages should be linked to *ontologies* [6,28] – databases which contain the information on classes of objects, their properties, and relations between the classes. The relations between objects are particularly important in an

*Equal contribution with Mikhail Tikhomirov.
**Corresponding author. E-mail: irina.nikishina@skoltech.ru.

ontology, because they form its structure. They can be of different types corresponding to different types of relationships between real-world objects. One of the most important relationships is the class-subclass relation. It allows organising entities into a *taxonomy* – a tree structure where entities are represented as nodes and the edges between them denote *subclass-of* or *instance-of* relationship. The class-subclass relations and taxonomies built from them are crucial for understanding the place and the purpose of an object or a concept in the world. This relation and the hierarchical structure created by it are also a basis of many knowledge bases and *knowledge graphs* – a particular type of knowledge bases where objects are organised in a graph structure. There, the nodes of a graph are objects, and the edges of a graph are relations between the objects.

To support the use of specific knowledge base structures, such as thesauri or taxonomies, there exist specifications and standards classification schemes. For instance, Simple Knowledge Organization Systems (SKOS)[1] define several types of lexical-semantic relations in the terms of semantic web, e.g. "has broader/narrower", "has exact match", and "is in mapping relation with". SKOS are used to create structures like thesauri.

However, SKOS does not fully comply with our taxonomy. The class-subclass relation which is the base of taxonomies is a relation between objects X and Y such that the sentence "X is a kind of Y" is acceptable for native speakers. On the other hand, the closest SKOS analogue of taxonomic class-subclass relation is the "broader term relation". It is different from the class-subclass relation, because it is less specific. For example, it can include the part-whole relations.

The usefulness of an ontology or a knowledge base depends largely on its completeness and its ability to fully reflect the real world. However, since the world is changing, the ontologies need to be constantly updated to stay relevant. There currently exist comprehensive knowledge bases such as Freebase, DBPedia or Wikidata as well as ontologies for specific domains. Many areas of knowledge require their own knowledge bases, and all of them need to be maintained and extended. This is an expensive and time-consuming process which can only be conducted by an expert who is proficient in the discipline and understands the structure of a knowledge base. Thus, in order to speed up and simplify this task, it becomes more and more important to develop systems that could automatically enrich the existing knowledge bases with new words or at least facilitate the manual extension process. The task of automatically or semi-automatically adding new entities to hierarchical structures is referred to as *taxonomy enrichment*.

In this work, we aim at reviewing the existing taxonomy enrichment models and propose new methods which address their drawback. We also aim at evaluating the scalability of different methods to new languages and datasets.

The state-of-the-art taxonomy enrichment methods have two main drawbacks. First of all, they often use unrealistic formulations of the task. For example, SemEval-2016 task 14 [33] which was the first effort to evaluate this task in a controlled environment, provided definitions of the query words (words to be added to a taxonomy). This is very informative resource, so the majority of the presented methods heavily depended on those definitions [25,76]. However, in the real-world scenarios, such information is usually unavailable, which makes the developed methods inapplicable. We tackle this problem by testing our new methods and the state-of-the-art methods in a realistic setting.

Another gap in the existing research is that the majority of methods use the information from only one source. Namely, some researchers use the information from distributional word embeddings, whereas others consider graph-based models which represent a word based on its position in a taxonomy. Our intuition is that the information from these two sources is complementary, so combining them can improve the performance of taxonomy enrichment models. Therefore, we propose a number of ways to incorporate various sources of information.

First, we propose the new **DWRank** method which uses only distributional information from pre-trained word embeddings and is similar to other existing methods. We then enable this method to incorporate the different sources of graph information. We compare the various ways of getting the information from a knowledge graph. Finally, another modification of our method successfully combines the information from different sources, beating the current state of the art.

To place our models in the context of the research on taxonomy enrichment, we compare them with a number of state-of-the-art models. To the best of our knowledge, this is the first large-scale evaluation of taxonomy enrichment methods. We are also the first to evaluate the methods on datasets of different sizes and in different languages.

[1] https://www.w3.org/2004/02/skos/intro

This work is an extended version of the work described in [57,58,77]. The novelty of this particular article as compared to the previous publications is as follows:

1. We present a new taxonomy enrichment method **DWRank** which combines distributional information and the information extracted from Wiktionary.
2. We present an extension of DWRank called **DWRank-Graph** which uses various graph-based representations via a common interface.
3. We present **DWRank-Meta** – an extension of DWRank which combines the information from different sources and beats the state-of-the-art models.
4. We present **WBSR** – a method for taxonomy extension which leverages the information from the Web. This approach is the current state of the art in the task.
5. We conduct a large-scale computational study of various approaches to taxonomy enrichment, which features multiple methods (including ours as well as state-of-the-art approaches), multiple datasets and languages.
6. We present datasets for studying the evolution of wordnets for English and Russian, extending the monolingual setup of the RUSSE'2020 shared task [56] with a larger Russian dataset and similar English versions.
7. We explore the benefits of meta-embeddings (combinations of embeddings) and graph embeddings for the task of taxonomy enrichment.
8. We provide an in-depth error analysis of different types of mistakes of the state-of-the art models.
9. We provide mappings to WordNet Linked Open Data (LOD) Inter-Lingual Index (ILI) from Russian to English synsets. A dataset of multilingual hypernyms opens possibilities for various use-cases for cross-lingual operations on taxonomies.

We release all our code and datasets to enable further research in this direction.[2]

The rest of the paper is organised as follows. In Section 2 we formulate the task and provide the relevant definitions. Section 3 is devoted to the existing work on taxonomy enrichment. Then, in Section 4 we present our datasets and describe their creation process and features. Sections 5, 6, and 7 contain the descriptions of our methods: in Section 5 we introduce our core method DWRank and in Sections 6 and 7 describe its extensions DWRank-Graph and DWRank-Meta. We report the performance of our models and compare them with the other approaches in Section 8. Finally, in Section 9 we analyse the potential limitations of our methods.

2. Task formulation and definitions

In this subsection we formulate the task and explain the main concepts and task-related terms.

2.1. Task formulation

Let us consider an example of taxonomy. Figure 1 demonstrates a subgraph for the word "Papuan" retrieved from WordNet. There, each concept has one or more *parents* (concepts which it is derived from). The parents in their turn have their own parents, and one can trace the word affiliation until the most abstract *root* concepts of the taxonomy. Here, the word "Papuan" is attached to the synsets "indonesian" and "natural_language" as it can mean both an ethnicity and a language. Note that words can have multiple parents for different reasons. Two or more parents can point to the fact that a word has multiple meanings (as in the case with "Papuan"). Alternatively, a word meaning can be a combination of meanings of two higher-level concepts.

Therefore, in order to add a word to a taxonomy, we need to find its hypernym among the entities (synsets in case of wordnets) of this taxonomy. Here, we refer to a word absent from the taxonomy (a word which we would like to add) as a *query word*. Our task is to attach query words to an existing taxonomic tree.

The task of finding a single suitable hypernym synset is difficult for a machine, because the number of nodes in the existing taxonomy can be very large (e.g. in WordNet the number of noun synsets is around 29,000). Thus,

[2] https://github.com/skoltech-nlp/diachronic-wordnets

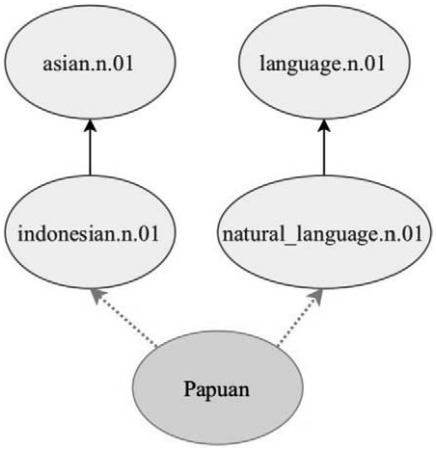

Fig. 1. Example of adding a new word "Papuan" to the taxonomy.

a model trained to solve this task will inevitably return many false answers if asked to provide only one synset candidate. Moreover, as we can see from Fig. 1, a word may have multiple hypernyms.

Thus, in the majority of works on taxonomy enrichment the requirement of providing a single correct answer is relaxed. Instead of that, a common approach is to provide k (typically 10 to 20) most suitable candidates. This list is more likely to contain correct synsets. This setting is also consistent with the manual computer-assisted annotation. Presenting an annotator with a small list of candidates will facilitate the taxonomy extension process: annotators will only have to look through a short list of high-probability hypernym candidates instead of searching hypernyms in a list of all synsets of the taxonomy. Thus, we formulate the task of *taxonomy extension* as a soft ranking problem, where the synsets are ranked according to their suitability for a given word.

This is an established approach to this task, the same formulation was also used in research works on taxonomy enrichment and in taxonomy enrichment shared tasks [33,56].

2.2. Definitions

In this section, we provide several key definitions crucial for understanding of our work.

Knowledge base is a data structure used to store knowledge which comprise of concepts and relations (known as TBox-es) and individuals, concept, and relation instantiations (also known as ABox-es) [21].

Knowledge graph is a knowledge base that uses a graph-structured data model to represent data. Formally, a knowledge graph G is a set of triplets $\{h, r, t\} \subseteq E \times R \times E$ where E is the entity set and R is the relation set. Normally, knowledge graphs comprise multiple types of relations R.

Synset is a set $S \subseteq \{s_1, \ldots, s_n\}$ of words or phrases corresponding to a concept or a knowledge base entity in the set E. Synset is the major element which denotes a node in a knowledge graph G, therefore, in our case S equals E.

Hypernymy relation is a relation $r \in R$. According to [52], hypernymy relation exists between objects X and Y if native speakers accept sentences constructed using such patterns as "An X is a (kind of) Y". Hypernymy is transitive and asymmetrical. Such relations are central organizing relations for nouns and verbs in WordNet [52]. In fact, hypernymy relations comprise class (or subsumption) relations and instantiation relations considered in ontology studies and description logics [48].

Taxonomy is a special case of a knowledge graph G. It is a tree-based structure where nodes from the set E are connected with the hypernymy relation r. Thus, the set of relations $R = \{r\}$. Elements included in the set E are words or concepts. In a taxonomy G they are arranged in a hierarchical structure from the most abstract concept (*root*) to the most specific concepts (*leaves*).

Query words (new words) – words to be added to the taxonomy T, usually manually collected by experts. In this paper, we use the following notation for this set $Q \subseteq \{q_1, \ldots, q_n\}$. Note that words may be ambiguous: one q_i can correspond to more than one synset s_i.

Knowledge graph completion is a task of completing a Knowledge Graph G by finding a set of missing triples $\{h, r, t | h \in E, r \in R, t \in E, h, r, t \notin G\}$.

Taxonomy enrichment can be considered as a special case of the knowledge base completion task. This task aimed at associating each new word $q \in Q$, which is not yet included in the taxonomy T, with the appropriate hypernyms from it.

3. Related work

There exist numerous approaches to knowledge base completion, which make use of various neural network architectures described in review papers [55,61]. Most of them apply low-dimensional graph embeddings [13,72], deep learning architectures like autoencoders [86] or graph convolutional networks [23,41,70]. Another group of approaches makes use of tensor decomposition approaches: Tucker decomposition [3] and Canonical Polyadic decomposition (CANDECOMP/PARAFAC) [39,40]. Another task, which is closely related to the taxonomy creation is the Knowledge base construction. There exists multiple approaches solving the problem: Text2Onto [17] a pillar language-independent approach which applies user interaction or fully automated methods [22,59] which apply text-mining tools and external sources, which are mostly applied for the scholarly domain. However, knowledge base completion task assumes a generic graph while in the taxonomy enrichment task deals with tree structures and specific methods of tree processing are commonly used in this field. In this article we focus on this specific task and narrow down our scope to enrichment and population of taxonomic structures. It should be noted however that the majority of the ontologies and knowledge bases possess some kind of taxonomic backbone and therefore the task of construction and maintaining such a semantic structure is fundamental.

The existing studies on the taxonomies can be divided into three groups. The first one addresses the Hypernym Discovery problem [14]: given a word and a text corpus, the task is to identify hypernyms in the text. However, in this task the participants are not given any predefined taxonomy to rely on. The second group of works deals with the Taxonomy Induction problem [11,12,81], in other words, creation of a taxonomy from scratch. Finally, the third direction of research is the Taxonomy Enrichment task: the participants extend a given taxonomy with new words. In this article, we focus on the taxonomy enrichment task and explore various approaches based on text and graph embeddings and their combinations to the solution of this problem. The following sections provide overview of the various strains of research related to the given problem.

3.1. Prior art on taxonomy enrichment

Until recently, the only dataset for the taxonomy enrichment task was created under the scope of SemEval-2016. It contained definitions for the new words, so the majority of models solving this task used the definitions. For instance, *Deftor* team [76] computed definition vector for the input word, comparing it with the vector of the candidate definitions from WordNet using cosine similarity. Another example is *TALN* team [25] which also makes use of the definition by extracting noun and verb phrases for candidates generation.

This scenario may be unrealistic for manual annotation because annotators are writing a definition for a new word and adding new words to the taxonomy simultaneously. Having a list of candidates would not only speed up the annotation process but also identify the range of possible senses. Moreover, it is possible that not yet included words may have no definition in any other sources: they could be very rare (“apparatchik”, “falanga”), relatively new (“selfie”, “hashtag”) or come from a narrow domain (“vermiculite”).

Thus, following RUSSE-2020 shared task [56], we stick to a more realistic scenario when we have no definitions of new words, but only examples of their usage. For this shared task we provide a baseline as well as training and evaluation datasets based on RuWordNet [43] which will be discussed in the next section. The task exploited words which were recently added to the latest release of RuWordNet and for which the hypernym synsets for the words were already identified by qualified annotators. The participants of the competition were asked to find synsets which could be used as hypernyms.

The participated systems mainly relied on vector representations of words and the intuition that words used in similar contexts have close meanings. They cast the task as a classification problem where words need to be

assigned one or more hypernyms [37] or ranked all hypernyms by suitability for a particular word [20]. They also used a range of additional resources, such as Wiktionary, dictionaries, additional corpora [2]. Interestingly, only one of the well-performing models [78] used context-informed embeddings (BERT [24]) or external tools such as online Machine Translation (MT) and search engines (the best-performing model denoted as *Yuriy* in the workshop description paper).

In this paper, we would like to work out methods which depend on graph based structures and combine them with the approaches applying word embeddings. At the same time, we want our methods to benefit from the existing data (e.g. corpora, pre-trained embeddings, Wiktionary).

3.2. Word vector representations for taxonomies

Approaches using word vector representations are the most popular choice for all tasks related to taxonomies. When solving the *Hypernym Discovery* problem in SemEval-2018 Task 9 [14] most of participants use word embeddings. For instance, Bernier-Colborne and Barriere [7] predict the likelihood of the relationship between an input word and a candidate using word2vec embeddings. Berend et al. [5] use Word2vec vectors as features of a logistic regression classifier. Maldonado and Klubicka [47] simply consider top-10 closest associates from the Skip-gram word2vec model as hypernym candidates. Pre-trained GloVe embeddings [62] are also used to initialize embeddings for an LSTM-based Hypernymy Detection model [73].

Participants also solve the SemEval-2016 Task 13 on taxonomy induction with word embeddings [65]: they compute the vector offset as the average offset of all the pairs generated and exploit it to predict hypernyms for the new data. Afterwards, in [1] the authors apply word2vec embeddings similarity to improve the approaches of the SemEval-2016 Task 13 participants.

The vast majority of participants of SemEval-2016 task 14 [33] and RUSSE'2020 [56] also apply word embeddings to find the correct hypernyms in the existing taxonomy. For instance, the participants compute a definition vector for the input word by comparing it with the definition vectors of the candidates from the wordnet using cosine similarity [76]. Another option is to train word2vec embeddings from scratch and cast the task as a classification problem [37]. Some participants compare the approach based on XLM-R model [19] with the word2vec "hypernyms of co-hyponyms" method [2]. It considers nearest neighbours as co-hyponyms and takes their hypernyms as candidate synsets.

Summing up, the usage of distributed word vector representations is a simple yet efficient approach to the taxonomy-related tasks and should be considered a strong baseline [14,56].

3.3. Meta-embedding approaches to word representation

Vector representations can be learned on various datasets and using various models. It has been shown that combining word embeddings is beneficial for NLP tasks, e.g. dependency parsing [4], and in medical domain [16].

Coates et al. [18] show that simple vector combining approaches, such as concatenation or averaging, can significantly improve the overall performance for several tasks. For instance, singular value decomposition (SVD) demonstrates good results with the ability to control the final dimension of vectors [85]. Autoencoders [9] further promote the idea of creating meta-embeddings. The authors propose several algorithms for combining various word vectors into one vector by encoding initial vectors into some meta-embedding space and then decoding backwards.

As for the CAEME approach, all word vectors are encoded into meta-vectors and then concatenated. Then, the decoding step uses a concatenated representation to predict the original vector representations. The AAEME approach is similar to CAEME, except that each vector is mapped to a fixed-size vector and all encoded representations are averaged, but not concatenated. An obvious advantage of this approach is the ability to control the dimension of meta-embeddings.

For any AEME approach, different loss functions can be used at the decoding stage: MSE loss, KL-divergence loss, cosine distance loss and also their combinations. In [53] the authors investigated the performance of the autoencoders depending on the loss function. They discover that there is no evident winner across tasks and that different loss functions are defined be different tasks.

Meta-embeddings are already used in such machine learning tasks as dependency parsing [4], classification in healthcare [16], named-entity recognition [53,83], sentiment analysis [53], word similarity and analogy tasks [9, 18,85]. To the best of our knowledge, meta-embeddings have not been applied to the taxonomy enrichment task, especially for the fusion of texts and graph embeddings.

3.4. Graph-based representations for taxonomies

Taxonomies can be represented as graphs and there exist various approaches to learn graph-based representations. They are thoroughly compared in [46] on the link prediction task, which is closely related to Taxonomy Enrichment. The paper also demonstrates that the combination of text and graph embeddings gives a boost on the link prediction task. Most of those methods listed in [46] have also been tested on tasks related to the taxonomy enrichment.

For instance, node2vec embeddings [29] are used for taxonomy induction among other network embeddings [42]. In [1], the authors perform the same task. They use hyperbolic Poincaré embeddings to enhance automatically created taxonomies. The SemEval-2016 subtask of reattaching query words to the taxonomy is quite similar to taxonomy enrichment which we perform. However, the datasets of the SemEval-2016 Task 13 are restricted to specific domains, which leaves an open question of the efficiency of Poincaré embeddings for the general domain and larger datasets. Moreover, [1] use Hearst Patterns to discover hyponym-hypernym relationships. This technique operates on words, and cannot be transferred to word-synset relations without extra manipulation.

As for the Knowledge Graph Construction task, which is a more general task in relation to the Taxonomy Induction, the vast majority of approaches also use word embeddings as node representations. Several approaches like [44] and apply ELMO embeddings [64] to predict entities and their relations for the knowledge graph. Other approaches [32] utilize a combination of ELMO, Poincaré and node2vec embeddings to enhance knowledge graph build upon Wikipedia.

Graph convolutional networks (GCNs) [36] as well as graph autoencoders [34] are mostly applied to the link prediction task on large knowledge bases. For example, in [67] the authors present an expanded review of the field and compare a wide variety of existing approaches. Graph embeddings are also often used for other taxonomy-related tasks, e.g. entity linking [66]. As for the taxonomy enrichment task, we are only aware of a recent approach TaxoExpan [71] which applies position-enhanced graph neural networks (GCN [35] and GAT [82]) that we also evaluate on our datasets.[3]

Thus, to the best of our knowledge, our work is the first computational study of Taxonomy enrichment task which aggregates and considers different existing and new approaches for taxonomy enrichment. We compare graph- and word-based representations computed from the synsets and hypo-hypernym relations for hypernym prediction demonstrating state-of-the-art results.

4. Diachronic WordNet datasets

The important part of our study is the observation that one can learn from the history of the development of lexical resources though time. More specifically, we make use of the various historic snapshots (versions) of WordNet lexical graphs and setup a task of their automatic completion assuming the manual update of the ground truth. This diachronic analysis – similar to diachronic lexical analysis of word meanings – is used to build two datasets in our study: one for English, another one for Russian based respectively on Princeton WordNet [51] and RuWordNet taxonomies. It is important to mention that by using the word "diachronic" we do not imply lexical diachrony, e.g., semantic shifts of words over time [69], but the temporal extension of Wordnet stored in its versions. Each dataset consists of a taxonomy and a set of novel words to be added to this resource. The statistics are provided in Table 1.

[3]The results achieved on our datasets are significantly lower than the baseline, probably because of the incorrect model launching.

Table 1
Statistics of two diachronic WordNet datasets used in this study

Dataset	Nouns	Verbs
WordNet1.6–WordNet3.0	17043	755
WordNet1.7–WordNet3.0	6161	362
WordNet2.0–WordNet3.0	2620	193
RuWordNet1.0–RuWordNet2.0	14660	2154
RUSSE'2020	2288	525

Table 2
Statistics of the English WordNet taxonomies used in this study

Taxonomy	Synsets		Lemmas		New words	
	Nouns	Verbs	Nouns	Verbs	Nouns	Verbs
WordNet 1.6	66025	12127	94474	10319	–	–
WordNet 1.7	75804	13214	109195	11088	11551	401
WordNet 2.0	79689	13508	114648	11306	4036	182
WordNet 2.1	81426	13650	117097	11488	2023	158
WordNet 3.0	82115	13767	117798	11529	678	33

4.1. English dataset

To compile dataset, we choose two versions of WordNet and then select words which appear only in a newer version. For each word, we get its hypernyms from the newer WordNet version and consider them as gold standard hypernyms. We add words to the dataset if only their hypernyms appear in both versions. We do not consider adjectives and adverbs, because they often introduce abstract concepts and are difficult to interpret by context. Besides, the taxonomies for adjectives and adverbs are worse connected than those for nouns and verbs making the task more difficult.

In order to find the most suitable pairs of releases, we compute WordNet statistics (see Table 2). New words demonstrate the difference between the current and the previous WordNet version. For example, it shows that the dataset generated by "subtraction" of WordNet 2.1 from WordNet 3.0 would be too small, they differ by 678 nouns and 33 verbs. Therefore, we create several datasets by skipping one or more WordNet versions. The statistics for the datasets we select for our study are provided in Table 1.

As gold standard hypernyms, we use not only the immediate hypernyms of each *lemma* (initial form of a word – infinitive for a verb, single number and nominative case for a noun, etc.) but also the second-order hypernyms: hypernyms of the hypernyms. We include them in order to make the evaluation less restricted. According to our empirical observations, the task of automatically identifying the exact hypernym might be too challenging, and finding the "region" where a word belongs ("parents" and "grandparents") can already be considered a success.

This method of dataset construction does not use any language-specific or database-specific features, so it could be transferred to other wordnets or taxonomies with timestamped releases.

All datasets[4] created for this research and the code[5] for their construction are publicly available.

4.2. Russian datasets

In order to create an analogous version to English dataset for Russian, we use the RuWordNet taxonomy [43]. RuWordNet comprises *synsets* – sets of synonyms expressing a particular concept. A synset consists of one or more *senses* – words or multi-word constructions in the initial form. Therefore, we use the current version of RuWordNet and the extended version of RuWordNet which has not been published yet to compile the dataset (cf. Table 1).

[4] https://zenodo.org/record/4279821
[5] https://github.com/skoltech-nlp/diachronic-wordnets

The RUSSE'2020 dataset was created for the Dialogue Evaluation [56] and can be viewed as a restricted subset of the Russian dataset. In the RUSSE'2020 the following categories of words were excluded:

- all three-symbol words and the majority of four-symbol words;
- diminutive word forms and feminine gender-specific job titles;
- words which are derived from words which are included in the published RuWordNet;
- words denoting inhabitants of cities and countries;
- geographic and personal names;
- compound words that contain their hypernym as a substring.

4.3. WordNet ILI mapping (ru-en)

In order to connect wordnets in different languages the Inter-Lingual Index (ILI) is used [10]. This mapping is designed to make possible coordination between wordnet projects. For the Russian test sets we also provide mapping from RuWordNet to WordNet.[6] For each hypernym synset of each query word we present the corresponding WordNet synset index. Table 3 demonstrates several examples of this kind of mapping. As we can see, datasets for the Russian nouns and verbs are extended with additional column called "WordNet synsets", where the corresponding WordNet3.0 synsets are listed in accordance with the Russian synset list.

However, not all synsets have an equivalent in the other language, as there exist untranslatable concepts and lacunae. Therefore, we present in the Table 4 the coverage of the WordNet synsets for the hypernyms of query words from the test set. This mapping can be further used for multilingual experiments.

4.4. Evaluation metric

The goal of diachronic taxonomy enrichment is to build a newer version of a wordnet by attaching the new given terms to the older wordnet version. We cast this task as a soft ranking problem and use Mean Average Precision (MAP) score for the quality assessment:

$$\begin{aligned} MAP &= \frac{1}{N}\sum_{i=1}^{N} AP_i, \\ AP_i &= \frac{1}{M}\sum_{i}^{n} prec_i \times I[y_i = 1], \end{aligned} \tag{1}$$

Table 3
Examples of Russian nouns with translation mapped to English WordNet

Word	Translation	RuWordNet hypernyms	RuWordNet hypernym names	WordNet hypernyms
абсентеизм	absenteeism	["147309-n", "117765-n", "117017-n"]	['неучастие, отказ от участия', 'уклониться (отказаться)', 'отказаться, не согласиться']	["non-engagement.n.01", "evasion.n.03", "rejection.n.01"]
кибертерроризм	cyber terrorism	["7334-n", "4590-n", "2400-n"]	['преступление против общественной безопасности', 'компьютерное преступление', 'терроризм']	[null, "cybercrime.n.01", "terrorism.n.01"]
метропоезд	subway train	["141975-n", "7133-n"]	['электрическое транспортное средство', 'электропоезд']	[null, null]

[6] https://doi.org/10.5281/zenodo.4969267

Table 4
Coverage of the WordNet synsets for the hypernyms in the Russian test sets

Input type	Total	Have ILI mapping
	Non-restricted nouns	
All synsets	41,694	28,425
Unique synsets	4,777	2,791
Query words	17,475	15,251
	Restricted (private) nouns	
All synsets	4,456	3,087
Unique synsets	1,376	885
Query words	1,920	1,720
	Non-restricted verbs	
All synsets	6,783	4,860
Unique synsets	1,473	931
Query words	2,872	2,606
	Restricted (private) verbs	
All synsets	1,110	821
Unique synsets	611	419
Query words	477	440

where N and M are the number of predicted and ground truth values, respectively, $prec_i$ is the fraction of ground truth values in the predictions from 1 to i, y_i is the label of the i-th answer in the ranked list of predictions, and I is the indicator function.

This metric is widely acknowledged in the Hypernym Discovery shared tasks, where systems are also evaluated over the top candidate hypernyms [14]. The MAP score takes into account the whole range of possible hypernyms and their rank in the candidate list.

However, the design of our dataset disagrees with MAP metric. As we described in Section 4, the gold-standard hypernym list is extended with second-order hypernyms (parents of parents). This extension can distort MAP. If we consider all gold standard answers as compulsory for the maximum score, it means that we demand models to find both direct and second-order hypernyms. This disagrees with the original motivation of including second-order hypernyms to the gold standard – it was intended to make the task easier by allowing a model to guess a direct *or* a second-order hypernym.

On the other hand, if we decide that guessing *any* synset from the gold standard yields the maximum MAP score, we will not be able to provide an adequate evaluation for words with multiple direct hypernyms. There exist two cases thereof:

1. the target word has two or more hypernyms which are co-hyponyms or one is a hypernym of the other – this word has a single sense, but the annotator decided that multiple related hypernyms are needed to reflect all shades of the meaning,
2. the target word has two or more hypernyms which are not directly connected in the taxonomy and neither are their hypernyms. This happens if:
   (a) the word's sense is a composition of senses of its hypernyms, e.g. "impeccability" possesses two components of meaning: ("correctness", "propriety") and ("morality", "righteousness");
   (b) the word is polysemous and different hypernyms reflect different senses, e.g. "pop-up" is a book with three-dimensional pages ("book, publication") and a baseball term ("fly, hit").

While the case 2(a) corresponds to a monosemous word and the case 2(b) indicates polysemy, this difference does not affect the evaluation process. We propose that in both these cases in order to get the maximum MAP score a model should capture all the unrelated hypernyms which correspond to different components of sense. At the same time, we should bear in mind that guessing a direct hypernym or a second-order hypernym are equally good

options. Therefore, following [56], we evaluate our models with modified MAP. It transforms a list of gold standard hypernyms into a list of connected components. Each of these components includes hypernyms (both direct and second-order) which form a connected component in a taxonomy graph. (According to graph theory, connected component is a subgraph, in which there is a path between any two nodes.) Thus, in the case 1 we will have a single connected component, and a model should guess *any* hypernym from it to get the maximum MAP score. In the cases 2(a) and 2(b) we will have multiple components, and a model should guess *any* hypernym from *each* of the components.

5. Base methods

Here we first describe our baseline model which is a method of synset ranking based on distributional embeddings and hand-crafted features (the method was proposed as a baseline for RUSSE-2020 shared task [58]). We then propose extending it with new features extracted from Wiktionary and use the alternative sources of information about words (e.g. graph representations) and their combinations.

5.1. Baseline

We consider the approach by Nikishina et al. [58] as our baseline. There, we first create a vector representation for each synset in the taxonomy by averaging vectors (pretrained embeddings) of all words from this synset. Then, we retrieve top 10 synsets whose vectors are the closest to that of the *query word* (we refer to these synsets as *synset associates*). For each of these *associates*, we extract their immediate hypernyms and hypernyms of all hypernyms (second-order hypernyms). This list of the first- and second-order hypernyms forms our *candidate set*. We need to rank the candidates by their relevance for the query word. Note that the lists of candidates for different associates can have intersections. When forming the overall candidate set, we make sure that each candidate occurs in it only once.

The intuition behind the method is the following. We propose that if a synset of a taxonomy is a *parent* of a word which is similar to our query word, it can also be a parent of this query word.

To rank the candidate set of synsets we train a Linear Regression model with L2-regularisation on the training dataset formed of the words and synsets of WordNet. Candidate hypernyms are ranked by their model output score. We limit the output to the $k = 10$ best candidates.

We rank the candidate set using the following features:

- $n \times sim(v_i, v_{h_j})$, where v_x is a vector representation of a word or a synset x, h_j is a hypernym, n is the number of occurrences of this hypernym in the merged list, $sim(v_i, v_{h_j})$ is the cosine similarity of the vector of the input word i and hypernym vector h_j;
- the candidate presence in the Wiktionary hypernyms list for the input word (binary feature);
- the candidate presence in the Wiktionary synonyms list (binary feature);
- the candidate presence in the Wiktionary definition (binary feature);
- the average cosine similarity between the candidate and the Wiktionary hypernyms of the input word.

5.2. DWRank

We present a new method of taxonomy enrichment – Distributional Wiktionary-based synset Ranking (**DWRank**). It combines distributional features with features from Wiktionary. DWRank builds up on the baseline described in Section 5.1. We extend the baseline Logistic Regression model with the new features which mainly account for the number of occurrences of a synset in the candidate lists of different synset associates (nearest neighbours) of the query word. We introduce the following new features:

- the number of occurrences (n) of the synset in the merged candidate list and the quantity $\log_2(2 + n)$ which serves for smoothing,
- the minimum, average, and maximum proximity level of the synset in the merged candidate list:

  - the level is 0 if the synset was added based on similarity to the query word,
  - the level of 1 is for the immediate hypernyms of the query word,
  - the level of 2 is for the hypernyms of the hypernyms,
- the minimum, average, and maximum similarities of the query word to all words of the synset,
- the features based on hyponyms of a candidate synset ("children-of-parents"):
  - we extract all hyponyms ("children") of the candidate synset,
  - for each word/phrase in each hyponym synset we compute their similarity to the query word,
  - we compute the minimum, average, and maximum similarity for each hyponym synset,
  - we form three vectors: a vector of minimums of similarities, average similarities, and maximum similarities of hyponym synsets,
  - for each of these vectors we compute minimum, average, and maximum. We use these resulting 9 numbers as features.

  These features account for different aspects of similarity of the candidate's children to the query word and help defining if these children can be the query word's co-hyponyms ("siblings").

Moreover, in this approach we use cross-validation and feature scaling when training the Logistic Regression model.

This methods could be easily extended to other languages that possess a taxonomy, a wiki-based open content dictionary (Wiktionary) and text embeddings like fastText or/and word2vec and GloVe.

5.3. Web-based Synset Ranking (WBSR)

In this section, we describe **WBSR** (Web-based Synset Ranking) a method which leverages the power of the existing general-purpose services. It makes use of the two famous search engines: Google (for both English and Russian datasets) and Yandex[7] (for Russian only). According to our hypothesis, the search results for a word are likely to contain its hypernyms or co-hyponyms as they are often used to define a word via generalisation or by providing synonyms (co-hyponyms). For instance, if we do not know what "abdominoplasty" is, searching for it with a search engine can yield its definition "a cosmetic surgery procedure".

Another source that we could probably benefit from is another taxonomy, preferably larger than the one we work with. However, there might be no other taxonomies available in the same language. Therefore, in this case we can resort to Machine Translation and automatically translate query words into a rich-resource language (e.g. English) in order to use an existing taxonomy (e.g. English Princeton WordNet). In this study we use Yandex Machine Translation system[8] to translate query words into English and then translate hypernyms (if they are found) back into Russian.

The main drawback of using external sources such as search engines and machine translation systems is their weak reproducibility. The search results are dependent on the search history, so reproducing the experiment on a different account or after a relatively long period of time is problematic. However, since the method greatly improves the performance even with trivial handling of the collected data, we use it despite its drawback. To make our results reproducible, we release all data from the external sources used in our approach.[9]

Similarly to the approaches described above, here we also make use of Wiktionary and fastText embeddings cosine similarity. However, we treat synsets and words/phrases that they consist of in a different way. In the previously described approaches we computed embeddings for multiword phrases by averaging word embeddings of individual words in them. Here we treat them as sentences — we compute their embeddings using the `get_sentence_vector` method from fastText Python library. There, fastText vectors are divided by their norms and then averaged, so that only vectors with the positive L_2-norm value are considered. Secondly, we do not combine the word/phrase vectors into a synset vector but operate with the word/phrase embeddings directly.

[7] https://yandex.com
[8] https://translate.yandex.ru/
[9] https://doi.org/10.5281/zenodo.4540717

Similarly to DWRank, the algorithm consists of two steps: candidate generation and candidate ranking. Our candidate list is formed of the following synsets:

- synsets which contain words/phrases from the list of top-10 nearest neighbours of the query word;
- hypernyms and second-order hypernyms of those synsets;
- Wiktionary-based candidates:
  - synsets that contain words/phrases listed in Wiktionary as the hypernyms of the query word;
  - hypernym synsets of these synsets;
- cross-lingual candidates (for the Russian language only):
  - synsets that contain words/phrases listed in the English WordNet as the hypernyms of the query word;
  - hypernym synsets of these synsets.

Analogously to DWRank, we then rank all candidates by a logistic regression model which uses the following features:

- the candidate synset contains a word/phrase from the list of query word's nearest neighbours;
- the candidate synset is a hypernym of one of the nearest neighbours;
- the candidate is a second-order hypernym of one of the nearest neighbours;
- the candidate synset contains a hypernym of the query word from Wiktionary;
- the candidate synset is a hypernym of the synset which contains a hypernym of the query word from Wiktionary;
- the candidate synset contains a word which is present in the definition of the query word from Wiktionary;
- the candidate synset is present in the list of English WordNet synset candidates (for the Russian language only);
- the candidate synset is present in the list of hypernyms of the WordNet candidates (for the Russian language only);
- the candidate synset contains words which occur on the Google results page;
- the candidate synset contains words which occur on the Yandex results page (for the Russian language only).

The training set for the logistic regression model is formed from a wordnet in the relevant language as follows. For each query word in the list of query words we first find the most similar lemma which is contained in the wordnet. We know hypernyms for these lemmas and use them to generate the training set. We generate the candidate list as described before. First- and second-order hypernyms in this candidate list are used as positive examples for the corresponding lemmas, and synsets from the candidate list which are not hypernyms are considered negative examples.

This approach participated in the RUSSE'2020 Taxonomy Enrichment task for the Russian Language. The method achieved the best result on the nouns track. Therefore, we consider it as the-state-of-the-art method for Russian.

5.4. WordNet path prediction

A completely different approach to make use of fastText embeddings is presented in the work of Cho et al. [15]. The authors experiment with encoder-decoder models in order to solve the task of the direct hypernym prediction. They use a standard LSTM-based sequence-to-sequence model [75] with Luong attention [45]. First, they average fastText embeddings for the input word or phrase and put it through the encoder. The decoder sequentially generates a chain of synsets from the encoder hidden state, conditioned on the previously generated ones. The authors consider two different setups:

- *hypo2path* – given the input word, generate a sequence of synsets starting from the root synset and going down the taxonomy to the closest hypernym;
- *hypo2path reverse* – given the input word, generate a sequence of synsets starting from the closest hypernym up to the root entity.

To be able to apply this sequence-to-sequence architecture to our data, we build new datasets similar to the ones described in [15]. We generate a path from the WordNet starting from the root node to the target synset or word. Analogously to the original work, we include multiple paths from the root to the parents of the query word. We filter the validation set to only include queries that do not occur anywhere in the full taxonomy paths of the training data. To sort candidates generated by the decoder, we enumerate the generated *hypo2path* sequence from the right to the left or the *hypo2path reverse* from the left to the right and get the first 10 synsets.

Additionally, we extend this approach by replacing the LSTM+attention architecture with the Transformer architecture [79]. During training we provide an embedding of a synset as input to the Transformer and expect the model to generate a sequence of synsets starting from the hypernym of the input synset. During inference we provide embedding of query words as input expect the model to output sequences of synsets starting with the direct hypernyms.

5.5. Word representations for DWRank

We test our baseline approach and DWRank with different types of embeddings: fastText [8], word2vec [50] embeddings for English and Russian datasets and also GloVe embeddings [62] for the English dataset.

We use the fastText embeddings from the official website[10] for both English and Russian, trained on Common Crawl from 2019 and Wikipedia CC including lexicon from the previous periods as well. For word2vec we use models from [26,38] for both English[11] and Russian.[12] We lemmatise words and synsets for both languages with the same UDPipe [74] model which was used while training the representations. For the out-of-vocabulary (OOV) words we find all words in the vocabulary with the longest prefix matching this word and average their embeddings like in [20]. As for the GloVe embeddings, we also use them from the official website[13] trained on Common Crawl, the vocabulary size is 840 billion tokens.

6. DWRank-Graph

The DWRank method extracts a set of candidate synsets based on the similarities of word vectors. So far we used only distributional word vectors (fastText, GloVe, etc.) to represent words. On the other hand, graph-based representations can contain the taxonomic information which is absent in distributional embeddings [46].

Here, we present **DWRank-Graph**. This is the same DWRank method where the distributional embeddings are replaced with graph representations. The score prediction model and the features it uses do not change. Below we describe the graph representations and their combinations we applied in DWRank-graph.

6.1. Poincaré embeddings

Poincaré embeddings is an approach for "learning hierarchical representations of symbolic data by embedding them into hyperbolic space – or more precisely into an "n-dimensional Poincaré ball" [54]. Poincaré models are trained on hierarchical structures and simultaneously capture hierarchy and similarity due to the underlying hyperbolic geometry. According to the authors, hyperbolic embeddings are more efficient on the hierarchically structured data and may outperform Euclidean embeddings in several tasks, e.g, in Taxonomy Induction [1].

Therefore, we use Poincaré embeddings of our wordnets for the taxonomy enrichment task. We train Poincaré ball model for our wordnets using the default parameters and the dimensionality of 10, which yields the best results on the link prediction task [54].

However, applying these embeddings to the task is not straightforward, because Poincaré model's vocabulary is non-extensible. It means that new words that we need to attach to the existing taxonomy will not have any

[10] https://fasttext.cc/docs/en/crawl-vectors.html
[11] http://vectors.nlpl.eu/repository/20/29.zip
[12] http://vectors.nlpl.eu/repository/20/185.zip
[13] https://nlp.stanford.edu/projects/glove/

Poincaré embeddings at all and we cannot make use of the embeddings similarity. To overcome this limitation, we compute top-5 fastText nearest synsets (analogously to the procedure described in Section 5.1) and then aggregate embeddings in hyperbolic space using Einstein midpoint, following [30]. The resulting vector is considered as an embedding of the input word in the Poincaré space.

Then, we use vectors from this vector space to generate candidates for the DWRank approach. As the model we present in Section 5.2 does not depend on the types of input embeddings, we are able to provide Poincaré embeddings as input.

6.2. Node2vec embeddings

The hierarchical structure of the taxonomy is a graph structure, and we may also consider taxonomies as graphs and apply random walk approaches to compute embeddings for the synsets. For this purpose we apply node2vec [29] approach which represents a "random walk of a fixed length l" with "two parameters p and q which guide the walk in breadth or in depth". Node2vec randomly samples sequences of nodes and then applies the skip-gram model [49] to train their vector representations. We train node2vec representations of all synsets in our wordnets with the following parameters: $dimensions = 300$, $walk\_length = 30$, $num\_walks = 200$. The other parameters are taken from the original implementation.

However, analogously to Poincaré vector space, node2vec model has no techniques for representing out-of-vocabulary words. Thus, it is unable to map new words to the vector space. To overcome this limitation, we apply the same technique of averaging top-5 nearest neighbours from fastText and considering their mean vector as the new word embedding and search for the most similar synsets.

6.3. Graph neural networks

The models described above have a major shortcoming: the resulting vectors for the input words heavily depend on their representations in the fastText model. This can lead to the incorrect results if the word's nearest neighbour list is noisy and does not reflect its meaning. In this case the noise will propagate through the graph embedding (Poincaré or node2vec) model and result in inaccurate output even if the graph embedding model is of high quality.

Therefore, we test different graph neural network (GNN) architectures – Graph Convolutional Network [36], Graph Attention Network [82] and GraphSAGE [31] (SAmple and aggreGatE) which make use of both fastText embeddings and the graph structure of the taxonomy.

All the above mentioned models work similarly. According to [46], "GCN works similarly to the fully-connected layers for neural networks. It multiplies weight matrices with the original features but masking them with an adjacency matrix. Such a method allows to account not only node representation, but also representations of neighbours and two-hop neighbours." The GraphSAGE model addresses the problem of unseen nodes representation by training a set of aggregator functions that learn to aggregate feature information from a node's local neighborhood. GCN, on the contrary, learns a distinct embedding vector for each node. In GAT, the convolution operation from GCN is replaced with the attention mechanism. It uses the self-attention mechanism of Transformers [80] to aggregate the information from the one-hop neighbourhood.

FastText embeddings are used as input node features for all models, which is definitely an advantage of the model over Poincaré and node2vec, as they do not use word embeddings for training. Even though new words are not connected to the taxonomy, it is still possible to compute their embeddings according to their input node features.

We get the vector representations of query words from one of the pre-trained GNN models and then use them as the input to DWRank. Even though all methods work similarly, they demonstrate different performance on different datasets.

6.4. Text-associated deep walk

Text-Associated Deep Walk (TADW) [84] is another approach that incorporates text and graph information into one vector representation. The method is based on the DeepWalk algorithm [63] which learns feature representations

by simulating uniform random walks. To be specific, the sampling strategy in DeepWalk can be seen as a special case of node2vec with $p = 1$ and $q = 1$.

The authors prove that the DeepWalk approach is equivalent to matrix factorization. They incorporate text features of vertices into network representation learning within the framework of matrix factorization. First, they define matrix $M \in \mathbb{R}^{|V|\times|V|}$ where each entry M_{ij} is the logarithm of the average probability that vertex v_i randomly walks to vertex v_j in a fixed number of steps. In comparison to DeepWalk, where the goal is to factorize matrix M into the product of two low-dimensional matrices $W \in \mathbb{R}^{k\times|V|}$ and $H \in \mathbb{R}^{k\times|V|}$ ($k \ll |V|$), TADW aims to factorize matrix M into the product of three matrices: $W \in \mathbb{R}^{k\times|V|}$, $H \in \mathbb{R}^{k\times f_t}$ and text features $T \in \mathbb{R}^{f_t\times|V|}$. As text features, in this work we apply fastText embeddings.

After having learnt the factorisation of matrix M, we use rows of matrix W as node (synset) embeddings in DWRank.

6.5. High-order proximity preserved embeddings

High-Order Proximity preserved Embeddings (HOPE) [60] is yet another approach that embeds a graph into a vector space preserving information about graph properties and structure. Unfortunately, most structures cannot preserve the asymmetric transitivity, which is a critical property of directed graphs. To solve the problem, the authors employ matrix factorization to directly reconstruct asymmetric distance measures like Katz index, Adamic-Adar or common neighbors. This approach is scalable – it preserves high-order proximities of large scale graphs, and capable of capturing the asymmetric transitivity. HOPE outperforms state-of-the-art algorithms in tasks of reconstruction, link prediction and vertex recommendation.

As for our Taxonomy Enrichment task, we also apply HOPE to generate graph embeddings to be used as input in the DWRank model. The main difference with the other embeddings is that HOPE does not incorporate textual information from the nodes.

7. DWRank-Meta

In DWRank we employed only distributional information, i.e. pre-trained word embeddings, whereas in DWRank-Graph we represented words using the information from the graph structure of the taxonomy and usually ignoring their distributional properties. Meanwhile, taxonomy enrichment models may benefit from combining these two types of information. Therefore, we present **DWRank-Meta** – an extension of DWRank which combines multiple types of input word representations.

As in DWRank-Graph, the process of candidates selection, the feature set and the algorithm of synset ranking stay intact. Here we change the input representations of words and synsets.

7.1. Base Meta-Embeddings

The easiest way of combining embeddings of different types is to concatenate them and use the concatenated vector as an input. We refer to this method as **Concat** and combine different subsets of distributional (fastText, word2vec, GloVe) and graph-based (Poincaré, node2vec, GCN, GAT, GraphSAGE, TADW, HOPE) embeddings. In addition to that, we perform Singular value decomposition (SVD) over this concatenation as proposed in [85]. This approach is referred to as **SVD**.

7.2. Autoencoded Meta-Embeddings

We propose using two variants of autoencoders for the generation of meta-embeddings: Concatenated Autoencoded Meta-Embeddings (CAEME) and Averaged Autoencoded Meta-Embeddings (AAEME) [9]. They have shown good results on the task of evaluating lexical similarity. However, they have never been applied to taxonomy enrichment.

We generate meta-embeddings as follows. Let us consider an embedding model $s(w)$. For each of such embedding models we train an autoencoder consisting of an encoder and a decoder:

$$\begin{aligned} &E\big(s(w)\big) = h(w), \\ &D\big(h(w)\big) = \hat{s}(w), \\ &L_{ED} = dist\big(s(w), \hat{s}(w)\big), \end{aligned} \tag{2}$$

where E and D are the encoder and the decoder, and L is the loss used for training of the autoencoder. The loss is implemented as the distance (*dist*) between the original and the reconstructed embeddings. The *dist* can be any distance or similarity measure such as MSE, KL-divergence, or cosine distance. In our preliminary study, the cosine distance showed the best results, so we use it in our experiments.

Let us consider two embedding models $s_1(w)$ and $s_2(w)$. In such a case, the input to each decoder is not the result of the corresponding encoder, but meta-embeddings, which depends on the both encoders. Depending on the approach, meta-embeddings can be built in different ways, we construct the meta-embeddings as follows. In case of CAEME, we take an L_2-normalised concatenation of the two source embeddings encoded with respective encoders $E_1(s_1(w))$ and $E_2(s_2(w))$:

$$m(w) = \frac{E_1(s_1(w)) \oplus E_2(s_2(w))}{\|E_1(s_1(w)) \oplus E_2(s_2(w)\|_2}, \tag{3}$$

where $\oplus$ is the concatenation operation.

The drawback of this model is the growing dimensionality of meta-embeddings for cases where we combine multiple source embeddings. To fight that, we can replace the concatenation operation with averaging, yielding AAEME. It computes meta-embedding of a word w from its two source embeddings $s_1(w)$ and $s_2(w)$ as the L_2-normalised sum of internal representations $E_1(s_1(w))$ and $E_2(s_2(w))$:

$$m(w) = \frac{E_1(s_1(w)) + E_2(s_2(w))}{\|E_1(s_1(w)) + E_2(s_2(w))\|_2}. \tag{4}$$

In CAEME, the dimensionality of the meta-embedding space is the sum of the dimensions of the source embeddings, whereas in AAEME it stays the same. The AAEME encoder can be seen as a special case of the CAEME encoder where the meta-embedding is computed by averaging the two encoded sources in equation (3) instead of their concatenation.

7.3. Training of autoencoders

We can impose additional restrictions on *AEME models during training. One of such restrictions is the use of triplet loss. We restrict a word to be closer to the words that are semantically related to it according to the taxonomy than to a randomly chosen word with some *margin*:

$$\begin{aligned} L(w_a, w_p, w_n) = \max\big(&\|m(w_a) - m(w_p)\big)\| \\ &- \|m(w_a) - m(w_n))\| + margin, 0), \end{aligned} \tag{5}$$

where $\|\cdot\|$ is a distance function, w_a is the target word, w_p and w_n are positive and negative words, respectively.

The algorithm of calculating triplet loss is as follows:

1. for each word presented in the taxonomy, we compile a list of semantically related words which includes synonyms, hyponyms and hypernyms;
2. at each epoch, we randomly select K positive words from this related words set and form a set of K negative words by selecting them randomly from the vocabulary;
3. if the word is not presented in the taxonomy, then we cannot form a list of related words for it. In this case, we generate positive vectors for it by adding random noise to its vector;

4. next, we calculate the triplet margin loss by combining the triplet loss with the original loss as $\alpha * loss + (1 - \alpha) * triplet\_loss$.

We use the following parameters for the triplet loss: $K = 5$, $margin = 0.1$, $alpha = 0.005$. These parameters were selected via grid search with AAEME algorithm on the English 1.7 dataset.

8. Experiments

In this section, we report and discuss the performance of our models in the taxonomy enrichment task. We experiment with our DWRank approach and its modifications DWRank-Graph and DWRank-Meta. In addition to that, we compare them with the baseline introduced in RUSSE'2020 shared task and with a number of state-of-the-art methods. We conduct the experiments with English and Russian wordnets.

8.1. Experimental setup

For each result we add the standard deviation values. We calculate them as follows. We randomly sample 80% of the test data and calculate the MAP scores on that part of the test set. We repeat the same procedure 30 times and then calculate the standard deviation on those 30 MAP values.

The MAP metric should be interpreted as follows: the higher the score, the better the results. For instance, $MAP@k = 1.0$ means that all of the N correct hypernyms are present in the first top-N positions in the ranked list of candidates. Moreover, with $MAP@k = 1.0$ the first candidate is always correct. $MAP@k \geqslant 0.5$ means that at least one of the correct hypernyms is present in the two first positions (*top-2*) in the ranked lists of candidates. $MAP@k \geqslant 0.3$ means that at least one of the correct hypernyms is present in the first three positions (*top-3*) in the ranked lists of candidates.

We show the performance of different methods on the nouns attribution task for English (nouns 1.6) in Fig. 2 and for Russian (non-restricted nouns) in Fig. 3. The X axis shows the MAP score for each method, the methods are listed along the Y axis. For DWRank-Meta models, we list the embeddings used in the model in brackets.

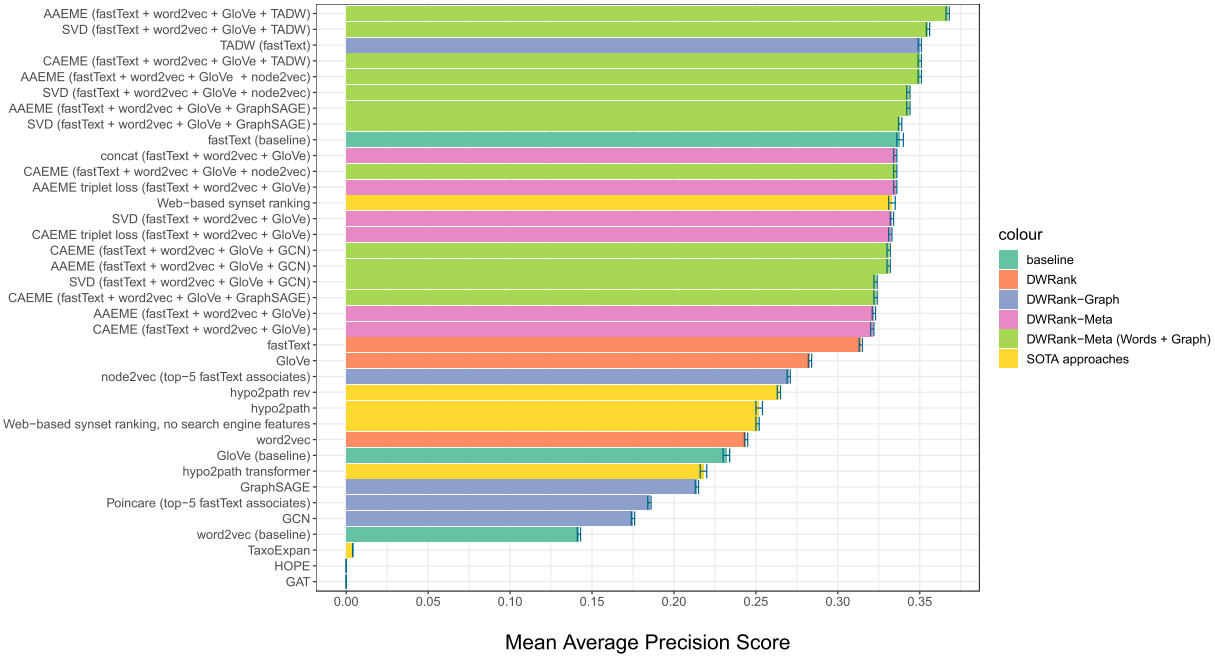

Fig. 2. Comparison of the method performance on *nouns_1.6* dataset for English. Each colour denotes the method type and the embeddings type used.

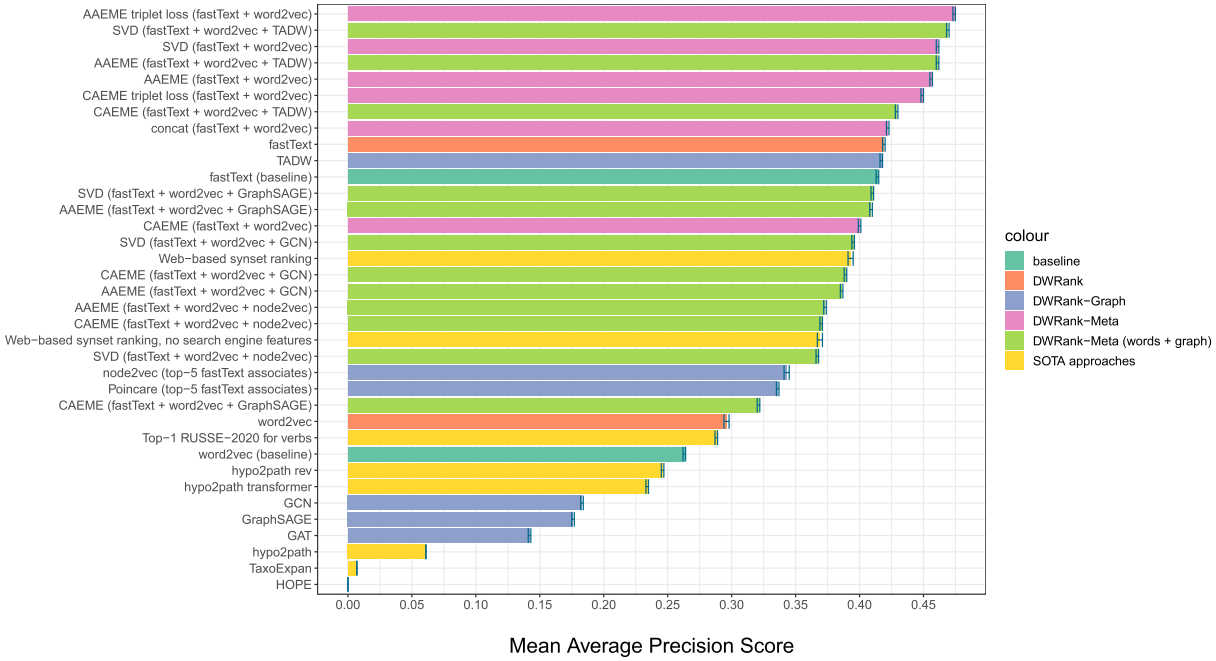

Fig. 3. Performance of different models on the Russian non-restricted dataset. Each colour denotes the method type and the embeddings type used.

The colors of bars in figures correspond to different types of input embeddings. The orange color stands for the vanilla DWRank – DWRank which uses only distributional embeddings. Purple denotes the DWRank-Graph variants. For the DWRank-Meta approaches exploiting only distributional embeddings we use pink, and DWRank-Meta on word and graph embeddings is denoted with the bright green color. Previous SOTA approaches are shown in yellow.

The full results for all experiments on the English and Russian datasets can be seen in Appendix 10 in Tables 12 and 13, respectively. Here, when listing embeddings used in DWRank-Meta models, we use the shortcut "*words*" to denote the combination of fastText, word2vec, and GloVe embeddings.

8.2. *Results*

DWRank-Meta Figures 2 and 3 show that the leaderboard for both English and Russian nouns is dominated by DWRank-Meta models. While English benefits from the union of distributional and graph embeddings, for Russian distributional embeddings alone perform on par with their combinations with graph embeddings. Besides that, high-performing variants of DWRank-Meta for English feature TADW, node2vec, and GraphSAGE, whereas for Russian TADW is the only graph embedding model which does not decrease the scores of DWRank-Meta.

We see that triplet loss significantly improves the results for DWRank-Meta models (cf. AAEME/CAEME with and without triplet loss) for both English and Russian.

DWRank-Graph On the other hand, DWRank-Graph fails in the task of taxonomy extension for all datasets. TADW model is the only graph embedding model which can compete with DWRank-Meta models. This can be explained by the fact that TADW is an extended version of DeepWalk and applies the skip-gram model with the pre-trained fastText representations. In contrast to that, the other graph models suffer from the noisy representations of OOV query words.

At the same time, despite the success of TADW, it does not outperform models based solely on distributional embeddings, showing that graph representations apparently do not contribute any information which is not already contained in distributional word vectors.

Baselines We also notice that for both languages the baselines are quite competitive. They are substantially worse than the best-performing models, but they are much simpler to implement and is fast and easy to train. Therefore, we suggest that it should be preferred in the situation of limited resources and time.

However, the choice of embedding models is crucial for the baselines (as well as for the vanilla DWRank which performs closely). We see that fastText outperforms word2vec and GloVe embeddings for almost all languages and datasets. The low scores of GloVe and word2vec embeddings on baseline and DWRank methods can be explained by data coverage issues. Fixed vocabularies of word2vec and Glove do not allow generating any representation for missing query words, whereas fastText can handle them.

SOTA models Neither of SOTA models managed to outperform the fastText baseline or approach the best DWRank-Meta variants. Web-based synset ranking (WBSR) model shows that the information from online search engines and Machine Translation models is beneficial for the task – its performance without this information drops dramatically. However, this information is not enough to outperform the word embedding-based models.

The performance of hypo2path model is even lower than that of WBSR. Being an autoregressive generative model, it is very sensitive to its own mistakes. Generating one senseless hypernym can ruin all the following chain. Conversely, when starting with the root hypernym "entity.n.01", it often takes a wrong path. Finally, TaxoExpan model relies on definitions of words which we did not provide in this task. Therefore, its results are close to zero. We do not consider them credible and provide them in italics.

Performance for different datasets Figures 2 and 3 as well as the results in the appendix show that there is no single best-performing model. While DWRank-Meta is almost always the best, for different datasets different variations of this model are the most successful. The results are usually consistent for the same language and part of speech (e.g. for different versions of English nouns datasets the best-performing model is the same), but there are exceptions to this regularity.

Interpretable evaluation MAP metric which is the standard way of evaluating taxonomy enrichment models has a serious drawback. Namely, it is not interpretable, which hampers the understanding of the models' performance.

Therefore, in addition to MAP we report the Precision@k score which can be interpreted as the ratio of correct synsets in the *top-k* outputs of a model. We evaluate our best systems automatically with the Precision@k ($k = 1, 2, 3$) score. The choice of the values of k is explained by the fact that the average number of true ancestors is 2 for the English words and 3 for the Russian words. Thus, Precision@k for $k > 3$ will be unfairly understated, because there will always be at most 3 correct answers out of k. This means that for the $k = 4$ the maximum is 0.75, for $k = 5$ it is 0.6, etc.

Table 5 shows the Precision@k scores for the best performing English and Russian models on the nouns datasets. Both of them are DWRank-Meta models with AAEME autoencoders. The English model uses three types of distributional embeddings and TADW graph embeddings, while the Russian model uses only fastText and word2vec but benefits from the triplet loss. We see that Precision@k is particularly high for Russian. There, over a half of generated lists contain a correct synset in the first position. This shows that DWRank-Meta can successfully be used as a helper tool for taxonomy extension.

The results for English are lower. However, this should not be considered as a sign of lower performance of models for English. The Russian and English datasets consist of different words, so they cannot be directly compared.

9. Error analysis

To better understand the difference in systems performance and their main difficulties, we made a quantitative and qualitative analysis of the results.

Table 5
Precision@k for the best-performing models for the English and Russian nouns datasets

Method	Pr@1	Pr@2	Pr@3
	English nouns 1.6–3.0		
baseline	0.260	0.184	0.144
DWRank	0.245	0.170	0.135
DWRank-Meta	0.288	0.185	0.143
DWRank-Graph	0.278	0.189	0.150
DWRank-Meta (Words + Graph)	**0.311**	**0.199**	**0.154**
	English verbs 1.6-3.0		
baseline	0.173	0.126	0.101
DWRank	**0.260**	**0.169**	0.126
DWRank-Meta	0.238	0.168	**0.131**
DWRank-Graph	0.238	0.158	0.118
DWRank-Meta (Words + Graph)	0.259	0.164	0.124
	Russian non-restricted nouns		
baseline	0.346	0.228	0.171
DWRank	0.347	0.228	0.172
DWRank-Meta	0.396	**0.257**	**0.196**
DWRank-Graph	0.347	0.224	0.168
DWRank-Meta (Words + Graph)	**0.397**	0.255	0.192
	Russian non-restricted verbs		
baseline	0.251	0.181	0.139
DWRank	0.282	0.196	0.154
DWRank-Meta	**0.368**	**0.245**	**0.190**
DWRank-Graph	0.274	0.191	0.149
DWRank-Meta (Words + Graph)	0.341	0.231	0.180

9.1. Comparison of graph-based approaches with word vector baselines

First of all, we wanted to know to what extent the set of correct answers of graph-based models overlaps with the one of fastText-based models. In other words, we would like to know if the graph representations are able to discover hypernymy relations which could not be identified by word embeddings.

Therefore, for each new word we computed average precision (AP) score and compared those scores across different approaches. We found that at least 90% words for which fastText failed to identify correct hypernyms (i.e. words with $AP = 0$) also have the AP of 0 in all the graph-based models. This means that if fastText cannot provide correct hypernyms for a word, other models cannot help either. Moreover, all words which are correctly predicted by graph-based approaches, are also correctly predicted by fastText. Moreover, only 8% to 55% words correctly predicted by fastText are also correctly predicted by any of the graph-based models. At the same time, the number of cases where graph-based models perform better than fastText is very low (3–5% cases). Thus, combining them cannot improve the performance significantly. This observation is corroborated by the scores of the combined models.

To contrast the performance of the text and graph embeddings and to demonstrate the input and the output formats of the models we present Tables 7 and 8 in Appendix 10. They demonstrate the main features of the tested approaches. The examples do not pretend do be the general case example, however, they do provide the idea about ranking of the results and the performance of text, graph and fusion embedding types.

9.2. Performance on polysemous words

The differences in word semantics make the dataset uneven. In addition to that, we would also like to understand whether the performance of models depends on the number of connected components (possible meanings)

(a) Russian dataset (nouns and verbs) (b) English dataset (nouns and verbs)

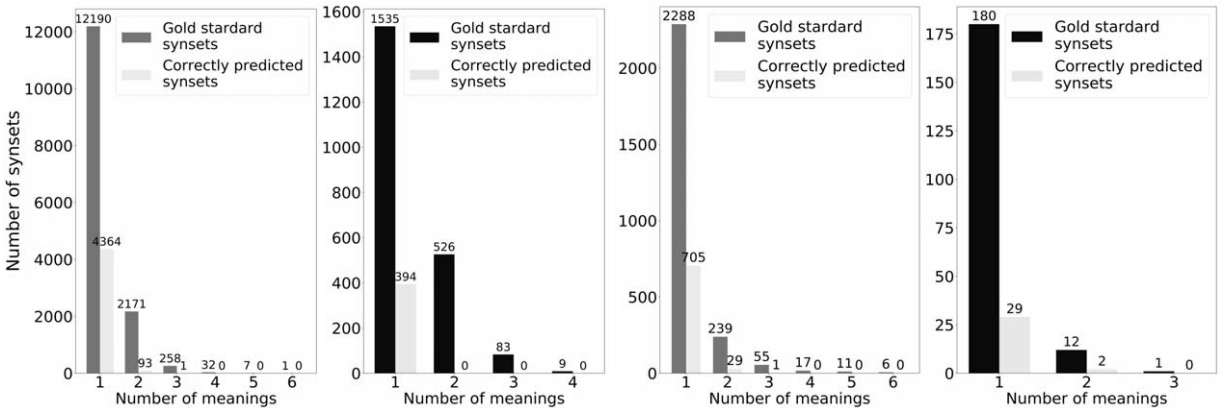

Fig. 4. Distribution of words over the number of senses.

for each word. Thus, we examine how many words with more than one meaning can be predicted by the system.

Figure 4 depicts the distribution of synsets over the number of senses they convey. As we can see, the vast majority of words are monosemous. For Russian nouns, the system correctly identifies almost half of them, whereas for other datasets the share of correctly predicted monosemous words is below 30%. This stems from the fact that for distributional models it is difficult to capture multiple senses in one vector. They usually capture the most widespread sense of a word. Therefore, the number of predicted synsets with two or more senses is extremely low. A similar power law distribution would be obtained using BERT embeddings [24], as we are still averaging embeddings from all contexts. This may be one of the reasons why contextualised models did not perform better than the fastText models which capture the main meaning only but do it well.

9.3. Error types

In order to understand why a large number of word hypernyms (at least 60%) are too difficult for models to predict, we turn to manual analysis of the system outputs. We find out that errors can be divided into two groups: system errors caused by distributional models limitations and taxonomy inaccuracies. Therefore, we come across five main error types:

Type 1. Extracted nearest neighbours can be semantically related words but not necessary co-hyponyms:

– delist (WordNet); expected senses: get rid of; predicted senses: remove, delete;
– хэштег (hashtag, RuWordNet); expected senses: отличительный знак, пометка (tag, label); predicted senses: символ, короткий текст (symbol, short text).

Type 2. Distributional models are unable to predict multiple senses for one word:

– latakia (WordNet); expected senses: tobacco; municipality city; port, geographical point; predicted senses: tobacco;
– запорожец (zaporozhets, RuWordNet); expected senses: житель города (citizen, resident); марка автомобиля, автомобиль (car brand, car); predicted senses: автомобиль, мототранспортное средство, марка автомобиля (car, motor car, car brand).

Type 3. System predicts too broad / too narrow concepts:

– midweek (WordNet); expected senses: day of the week, weekday; predicted senses: time period, week, day, season;

– медянка (smooth snake, RuWordNet); expected senses: неядовитая змея, уж (non-venomous snake, grass snake); predicted senses: змея, рептилия, животное (snake, reptile, animal).

Type 4. Incorrect word vector representation: nearest neighbours are semantically far from the meaning of the inputt word:

– falanga (WordNet); expected senses: persecution, torture; predicted senses: fish, bean, tree, wood.;
– кубокилометр (cubic kilometer, RuWordNet); expected senses: единица объема, единица измерения (unit of capacity, unit of measurement); predicted senses: город, городское поселение, кубковое соревнование, спортивное соревнование (city, settlement, competition, sports contest).

Type 5. Unaccounted senses in the gold standard datasets, inaccuracies in the manual annotation:

– emeritus (WordNet); expected senses: retiree, non-worker; predicted senses: professor, academician;
– сепия (sepia, RuWordNet); expected senses: морской моллюск “sea mollusc”; predicted senses: цвет, краситель (color, dye).

In order to check how useful the predicted synsets are for a human annotator (i.e. if a short list of possible hypernyms can speed up the manual extension of a taxonomy), we conduct the manual evaluation of 10 random nouns and 10 random verbs for both languages (the words are listed in Table 6). We focus on worse-quality cases and thus select words whose MAP score is below 1. Annotators with the expertise in the field and the knowledge of English and Russian were provided with guidelines and asked to evaluate the outputs from our best-performing system. Each word was labelled by 4 expert annotators, Fleiss’s kappa is 0.63 (substantial agreement) for both datasets.

We compute Precision@k score (the share of correct answers in the generated lists from position 1 to k) for k from 1 to 10, shown in Fig. 5. We can see that even for words with MAP below 1 our model manages to extract useful hypernyms.

Table 6
Words selected for the manual evaluation

Language	Word List
English	Falanga, venerability, ambulatory, emeritus, salutatory address, eigenvalue of a matrix, liposuction, moppet, dinette, snoek, to fancify, to google, to expense, to porcelainize, to junketeer, to delist, to podcast, to deglaze, to shoetree, to headquarter
Russian	барабашка, листинг, стихосложение, аукционист, точилка, гиперреализм, серология, огрызок, фен, марикультура, уломать, отфотошопить, тяпнуть, растушевать, завраться, леветь, мозолить, загоститься, распеваться, оплавить

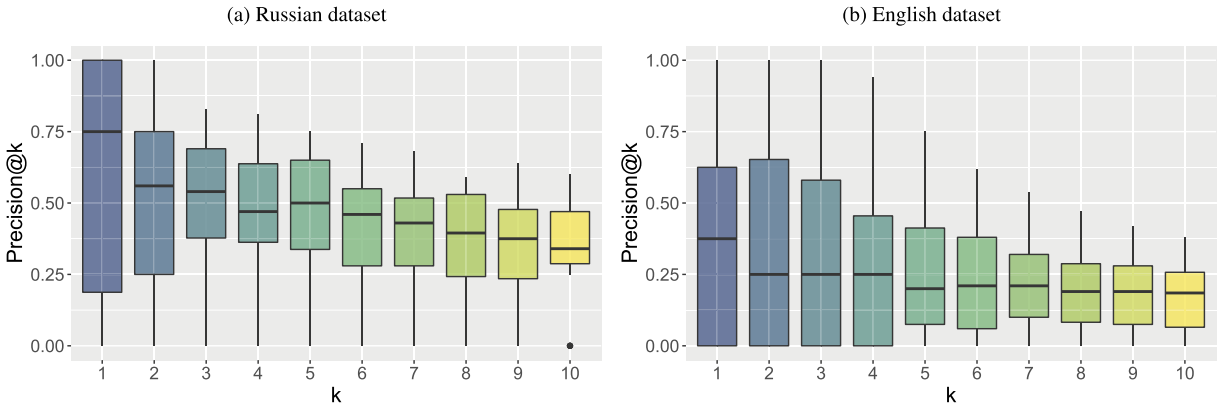

Fig. 5. Manual datasets evaluation results: Precision@10.

10. Conclusions

In this work, we performed a large-scale computational study of various methods for taxonomy enrichment. We also presented datasets for studying diachronic evolution of wordnets for English and Russian, extending the monolingual setup of the RUSSE'2020 shared task [56] with a larger Russian dataset and similar English versions.

We presented a new taxonomy enrichment method called DWRank, which combines distributional information and the information extracted from Wiktionary outperforming the baseline method from [58] on the English datasets. We also presented its extensions: DWRank-Graph and DWRank-Meta which use graph and meta- embeddings via a common interface.

We also explored the benefits of meta-embeddings (combinations of embeddings) and graph embeddings for the task of taxonomy enrichment. On the Russian datasets DWRank-Meta performed best using fastText and word2vec word embeddings. For the English dataset the combination of word (fastText, word2vec and GloVe) and graph (TADW) embeddings demonstrated the best performance.

In this paper we also presented WBSR – a method for taxonomy extension which leverages the information from the Web. This approach was the current state of the art in the task for the Russian language. Now it lags significantly behind the new DWRank-meta approaches.

According to our experiments, word vector representations are simple, powerful, and extremely effective instrument for taxonomy enrichment, as the contexts (in a broad sense) extracted from the pre-trained word embeddings (fastText, word2vec, GloVe) and their combination are sufficient to attach new words to the taxonomy. TADW embeddings are also useful and efficient for the taxonomy enrichment task and in combination with the fastText, word2vec and GloVe approaches demonstrate SOTA results for the English language and compatible results for the Russian language.

Error analysis also reveals that the correct synsets identified by graph-based models are usually retrieved by the fastText-based model alone. This makes graphs representations mostly irrelevant and excessive. Nonetheless, there exist cases where graph representations were able to identify correctly some hypernyms which were not captured by fastText.

Despite the mixed results of the application of graph-based methods, we propose further exploration of the graph-based features as the existing resource contains principally different and complementary information to the distributional signal contained in text corpora. One way to improve their performance, may be to use more sophisticated non-linear projection transformations from word to graph embeddings. Another promising way in our opinion is to explore other types of meta-embeddings to mix word and graph signals, e.g. GraphGlove [68]. Moreover, we find it promising to experiment with temporal embeddings such of those of [27] for the taxonomy enrichment task.

Last but not least, we plan to explore methods that do not rely on the set of pre-defined candidates for inclusion in a taxonomy, as generation and mining of such a set may be a challenging problem in its own.

Acknowledgements

The participation of Mikhail Tikhomirov in the reported study (experiments with meta-embeddings) was funded by RFBR, project number 19-37-90119. The work of Natalia Loukachevitch (preparation of RuWordNet data for the experiments, mappings of the RuWordNet synsets to English WordNet Linked Open Data (LOD) Inter-Lingual Index (ILI)) is supported by the Russian Science Foundation (project 20-11-20166). The work of Alexander Panchenko, Varvara Logacheva, and Irina Nikishina was conducted in the framework of joint MTS-Skoltech laboratory.

Appendix. Ranked examples and all results

In this Appendix we provide Tables 7, 8, 9, 10 and 11 with the examples that demonstrate the input and the output formats of the models as well as Tables 12 and 13 that show the performance of all models for both English and Russian datasets.

Table 7
Examples of predictions for noun from the English v 1.6-3.0 dataset with various models

Play therapy psychotherapy.n.02, therapy.n.01			
fastText (baseline)	fastText (DWRank)	AAEME triplet loss (fastText + word2vec + GloVe)	AAEME (fastText + word2vec + GloVe + TADW)
play.n.03	play.n.03	**therapy.n.01**	**therapy.n.01**
play.n.01	activity.n.01	activity.n.01	medical_care.n.01
baseball_play.n.01	plan_of_action.n.01	medical_care.n.01	play.n.03
therapy.n.01	**therapy.n.01**	diversion.n.01	play.n.01
activity.n.01	play.n.01	play.n.01	activity.n.01
diversion.n.01	dramatic_composition.n.01	act.n.02	dramatic_composition.n.01
plan_of_action.n.01	outdoor_game.n.01	**psychotherapy.n.02**	plan_of_action.n.01
action.n.01	medical_care.n.01	play.n.03	show.n.04
action.n.02	golf.n.01	dramatic_composition.n.01	treatment.n.01
dramatic_composition.n.01	diversion.n.01	behaviour_therapy.n.01	act.n.02

GraphSAGE	TADW	Poincaré	node2vec
baseball_play.n.01	play.n.03	activity.n.01	presentation.n.03
play.n.03	play.n.01	exposure.n.03	presentation.n.01
play.n.01	plan_of_action.n.01	rejection.n.01	operation.n.01
squeeze_play.n.02	dramatic_composition.n.01	agreement.n.06	performance.n.02
activity.n.01	activity.n.01	light_unit.n.01	contact.n.01
diversion.n.01	baseball_play.n.01	blessing.n.01	exposure.n.03
dramatic_composition.n.01	show.n.04	vulnerability.n.02	activity.n.01
attempt.n.01	diversion.n.01	action.n.02	exposure.n.08
plan_of_action.n.01	use.n.01	influence.n.02	union.n.04
play.n.17	action.n.02	assent.n.01	exposure.n.06

From almost all Tables 7, 8, 9 and 11 (underlined bold text denotes predictions of the model from the ground truth) we can see that at least one of the correct candidates usually appears in the list of candidates from word embeddings (first part of the table), whereas among candidates from graph embeddings we do not see any decent synsets. Poincaré embeddings retrieved by aggregating words from fastText provide too broad concepts which are clearly too far from the correct answers (“activity.n.01”, “exposure.n.03”, “action.n.02”). Node2vec embeddings are both semantically far and abstract. GraphSAGE is sticking to the word “play” and are too far from the correct answers in general. TADW manages to predict the correct synset “therapy.n.01” in the list of candidates, however, its position is much lower than the positions of the same synsets among the candidates provided by word embeddings-based systems.

The candidates for the words “play therapy” and “eyewitness” provided by models based on text embeddings do contain at least one true answer in the list. The position of such words varies from the forth to the sixth position.

DWRank-Meta on both word and graph embeddings may provide the results that improve the ranking, e.g. “therapy.n.01” is the correct candidate on the first place in the list for both AAEME triple loss (fastText, word2vec and GloVe) approach and for the AAEME (fastText, word2vec, Glove, TASDW) approach. For the word “eyewitness” the DWRank-Meta models on words and graphs the true candidates are placed in the worse positions, however, they still win before the DWRank-GRaph approach.

Table 8
Examples of predictions for nouns from the English v 1.6-3.0 dataset with various models

Ramadan			
islamic_calendar_month.n.01, calendar_month.n.01, fast.n.01, abstinence.n.02			
fastText (baseline)	fastText (DWRank)	AAEME triplet loss (fastText + word2vec + GloVe)	AAEME (fastText + word2vec + GloVe + TADW)
islamic_calendar_month.n.01	**islamic_calendar_month.n.01**	calendar.n.01	**islamic_calendar_month.n.01**
calendar_month.n.01	**calendar_month.n.01**	**islamic_calendar_month.n.01**	**calendar_month.n.01**
holiday.n.02	time_period.n.01	**calendar_month.n.01**	time_period.n.01
religious_holiday.n.01	calendar.n.01	lunar_calendar.n.01	calendar.n.01
abstinence.n.02	islam.n.01	time_period.n.01	muslim.n.01
hindu_calendar_month.n.01	lunar_calendar.n.01	religionist.n.01	religionist.n.01
day.n.04	asian.n.01	muslim.n.01	religious_holiday.n.01
sacred_text.n.01	religion.n.02	religion.n.02	holiday.n.02
god.n.01	muslim.n.01	islam.n.01	lunar_calendar.n.01
place_of_worship.n.01	holiday.n.02	person.n.01	islam.n.01
graphSAGE	TADW	Poincaré	node2vec
islamic_calendar_month.n.01	**islamic_calendar_month.n.01**	time_period.n.01	**calendar_month.n.01**
calendar_month.n.01	**calendar_month.n.01**	**islamic_calendar_month.n.01**	**islamic_calendar_month.n.01**
arab.n.01	time.n.02	religion.n.02	revolutionary_calendar_month.n.01
muslim.n.01	religious_holiday.n.01	time.n.02	islam.n.01
semite.n.01	calendar.n.01	measure.n.03	time_period.n.01
religionist.n.01	time_period.n.01	**calendar_month.n.01**	hindu_calendar_month.n.01
god.n.01	holiday.n.02	term.n.02	muharram.n.01
saint.n.02	lunar_calendar.n.01	year.n.01	shawwal.n.01
zoysia.n.01	god.n.01	time_off.n.01	rabi_i.n.01
islam.n.01	jewish_holy_day.n.01	leisure.n.01	rajab.n.01

Table 9
Examples of predictions for verbs from the English v 1.6-3.0 dataset with various models

eyewitness witness.v.01, watch.v.01			
fastText (baseline)	fastText (DWRank)	AAEME triplet loss (fastText + word2vec + GloVe)	AAEME (fastText + word2vec + GloVe + TADW)
be.v.01	inform.v.01	inform.v.01	inform.v.01
be.v.03	testify.v.02	testify.v.02	testify.v.02
be.v.08	communicate.v.02	communicate.v.02	confirm.v.02
watch.v.01	announce.v.01	see.v.10	communicate.v.02
testify.v.01	confirm.v.02	confirm.v.02	**watch.v.01**
testify.v.02	**watch.v.01**	**watch.v.01**	be.v.01
man.v.02	testify.v.01	witness.v.02	affirm.v.03
talk.v.01	affirm.v.03	affirm.v.03	testify.v.01
guard.v.01	report.v.03	testify.v.01	reject.v.01
confirm.v.01	record.v.01	verify.v.01	experience.v.01

graphSAGE	TADW	Poincaré	node2vec
see.v.05	inform.v.01	confirm.v.02	affirm.v.03
testify.v.02	testify.v.02	examine.v.02	confirm.v.02
err.v.01	communicate.v.02	affirm.v.03	understand.v.02
confirm.v.01	declare.v.01	testify.v.02	uphold.v.03
pronounce.v.02	announce.v.01	inform.v.01	determine.v.08
idealize.v.01	testify.v.01	justify.v.02	stay_in_place.v.01
judge.v.02	record.v.01	declare.v.01	justify.v.02
negate.v.03	**watch.v.01**	validate.v.03	fall_asleep.v.01
reason.v.01	report.v.03	uphold.v.03	resettle.v.01
disbelieve.v.01	report.v.01	testify.v.01	settle.v.04

Table 10
Examples of predictions for verbs from the English v 1.6-3.0 dataset with various models

theologise			
cover.v.05, broach.v.01, chew_over.v.01, think.v.03			
fastText (baseline)	fastText (DWRank)	AAEME triplet loss (fastText + word2vec + GloVe)	AAEME (fastText + word2vec + GloVe + TADW)
change.v.01	change.v.01	change.v.01	change.v.01
match.v.01	preface.v.01	preface.v.01	preface.v.01
date.v.03	make.v.03	convert.v.05	equal.v.03
reconstruct.v.01	date.v.03	match.v.01	date.v.03
determine.v.03	equal.v.03	chronologize.v.01	reconstruct.v.01
make.v.03	reason.v.01	reason.v.01	reason.v.01
speculate.v.01	chronologize.v.01	change_state.v.01	convert.v.05
convert.v.05	match.v.01	represent.v.09	formulate.v.03
reason.v.01	film.v.02	make.v.03	reflect.v.04
automatize.v.01	formulate.v.03	change.v.02	film.v.02

graphSAGE	TADW	Poincaré	node2vec
process.v.02	change.v.01	state.v.01	settle.v.04
affect.v.01	preface.v.01	speculate.v.01	stay_in_place.v.01
change.v.01	date.v.03	reason.v.01	understand.v.02
dive.v.01	film.v.02	preface.v.01	study.v.03
tame.v.01	reconstruct.v.01	match.v.01	resettle.v.01
sensitize.v.02	equal.v.03	generalize.v.01	fall_asleep.v.01
convert.v.01	convert.v.05	announce.v.02	speculate.v.01
estimate.v.01	formulate.v.03	express.v.02	discover.v.07
subject.v.01	reason.v.01	add.v.02	explicate.v.02
compound.v.05	commemorate.v.03	equal.v.01	behave.v.02

Table 11
Examples of predictions for verbs from the English v 1.6-3.0 dataset with various models

immunise			
protect.v.01, defend.v.02, inject.v.01, administer.v.04			
fastText (baseline)	fastText (DWRank)	AAEME triplet loss (fastText + word2vec + GloVe)	AAEME (fastText + word2vec + GloVe + TADW)
indoctrinate.v.01	indoctrinate.v.01	teach.v.01	**inject.v.01**
inject.v.01	**protect.v.01**	treat.v.01	remove.v.01
defend.v.02	**defend.v.02**	insert.v.02	inform.v.01
remove.v.01	teach.v.01	prevent.v.01	change.v.01
treat.v.01	prevent.v.02	isolate.v.01	better.v.02
destroy.v.01	**inject.v.01**	inoculate.v.01	insert.v.02
prevent.v.02	insert.v.02	indoctrinate.v.01	defend.v.02
insert.v.02	defend.v.01	kill.v.01	kill.v.01
teach.v.01	kill.v.01	change.v.01	indoctrinate.v.01
administer.v.04	discriminate.v.02	enable.v.01	protect.v.01

graphSAGE	TADW	Poincaré	node2vec
defend.v.02	**protect.v.01**	teach.v.01	receive.v.01
protect.v.01	indoctrinate.v.01	insert.v.02	insert.v.02
act.v.01	teach.v.01	inform.v.01	stay_in_place.v.01
negociate.v.01	**defend.v.02**	treat.v.01	get.v.01
attack.v.03	prevent.v.02	train.v.01	**inject.v.01**
prevent.v.02	**inject.v.01**	deceive.v.02	fall_asleep.v.01
insert.v.02	prevent.v.01	indoctrinate.v.01	accept.v.02
demilitarize.v.01	insert.v.02	misinform.v.01	settle.v.04
disarm.v.02	isolate.v.01	gull.v.02	resettle.v.01
foreswear.v.02	discriminate.v.02	interact.v.01	discover.v.07

Table 12

MAP scores for the taxonomy enrichment methods for the English datasets. Numbers **in bold** show the best model within the category, **underlined** numbers denote the best score across all the models. The combination of word embeddings (fastText, word2vec, GloVe) is denoted as *words*

Method	Nouns			Verbs		
	1.6-3.0	1.7-3.0	2.0-3.0	1.6-3.0	1.7-3.0	2.0-3.0
Baseline [58]						
fastText [8]	**0.338 ± 0.002**	**0.371 ± 0.002**	**0.400 ± 0.004**	**0.270 ± 0.007**	**0.203 ± 0.010**	**0.236 ± 0.011**
word2vec [50]	0.142 ± 0.001	0.178 ± 0.002	0.164 ± 0.004	0.229 ± 0.006	0.155 ± 0.008	0.212 ± 0.009
GloVe [62]	0.232 ± 0.002	0.188 ± 0.001	0.233 ± 0.004	0.146 ± 0.005	0.149 ± 0.008	0.191 ± 0.010
DWRank-Word						
fastText [8]	**0.314 ± 0.001**	**0.373 ± 0.003**	**0.418 ± 0.004**	**0.286 ± 0.007**	**0.218 ± 0.008**	**0.254 ± 0.012**
word2vec [50]	0.244 ± 0.001	0.271 ± 0.003	0.298 ± 0.004	0.099 ± 0.005	0.118 ± 0.008	0.141 ± 0.010
GloVe [62]	0.283 ± 0.001	0.329 ± 0.003	0.377 ± 0.004	0.182 ± 0.007	0.159 ± 0.008	0.203 ± 0.011
DWRank-Meta (Meta-embeddings based on Word Embeddings)						
concat (*words*)	**0.335 ± 0.001**	0.386 ± 0.003	0.386 ± 0.003	0.270 ± 0.007	0.194 ± 0.009	0.226 ± 0.011
SVD (*words*)	0.333 ± 0.001	**0.399 ± 0.003**	**0.456 ± 0.004**	0.277 ± 0.007	0.209 ± 0.010	0.264 ± 0.012
CAEME*words*)	0.321 ± 0.001	0.386 ± 0.003	0.448 ± 0.005	0.278 ± 0.007	0.205 ± 0.008	0.266 ± 0.015
AAEME (*words*)	0.322 ± 0.001	0.384 ± 0.003	0.453 ± 0.004	0.271 ± 0.007	**0.218 ± 0.008**	**0.273 ± 0.012**
CAEME triplet loss (*words*)	0.332 ± 0.001	0.394 ± 0.003	0.451 ± 0.004	0.273 ± 0.007	0.205 ± 0.007	0.256 ± 0.013
AAEME triplet loss (*words*)	**0.335 ± 0.001**	0.391 ± 0.003	0.453 ± 0.004	**0.280 ± 0.008**	0.212 ± 0.007	0.262 ± 0.014
DWRank-Graph						
GCN [35]	0.175 ± 0.001	0.249 ± 0.002	0.267 ± 0.002	0.162 ± 0.006	0.113 ± 0.005	0.149 ± 0.010
GAT [82]	0.000 ± 0.000	0.252 ± 0.002	0.000 ± 0.000	0.081 ± 0.003	0.064 ± 0.004	0.000 ± 0.000
GraphSAGE [31]	0.214 ± 0.001	0.282 ± 0.002	0.224 ± 0.003	0.127 ± 0.004	0.114 ± 0.004	0.090 ± 0.008
TADW [84] (on fastText)	**0.350 ± 0.001**	**0.392 ± 0.002**	**0.435 ± 0.004**	**0.268 ± 0.007**	**0.201 ± 0.007**	**0.217 ± 0.010**
Poincaré [54] (top-5 fastText associates)	0.185 ± 0.001	0.211 ± 0.002	0.229 ± 0.002	0.208 ± 0.006	0.147 ± 0.006	0.172 ± 0.012
node2vec [29] (top-5 fastText associates)	0.270 ± 0.001	0.312 ± 0.002	0.341 ± 0.004	0.175 ± 0.006	0.128 ± 0.007	0.118 ± 0.012
HOPE [60]	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
DWRank-Meta (Meta-embeddings based on Word and Graph Embeddings)						
SVD (*words* + node2vec)	0.343 ± 0.001	0.383 ± 0.003	0.434 ± 0.005	0.272 ± 0.006	0.194 ± 0.009	0.239 ± 0.011
CAEME (*words* + node2vec)	0.335 ± 0.001	0.379 ± 0.003	0.426 ± 0.004	0.242 ± 0.005	0.184 ± 0.009	0.221 ± 0.012
AAEME (*words* + node2vec)	0.350 ± 0.001	0.394 ± 0.003	0.446 ± 0.004	0.252 ± 0.007	0.184 ± 0.008	0.208 ± 0.012
SVD (*words* + TADW)	0.355 ± 0.001	0.414 ± 0.003	0.472 ± 0.004	**0.288 ± 0.007**	0.222 ± 0.009	**0.280 ± 0.013**
CAEME (*words* + TADW)	0.350 ± 0.001	0.404 ± 0.003	0.458 ± 0.004	0.267 ± 0.007	0.212 ± 0.007	0.247 ± 0.011
AAEME (*words* + TADW)	**0.367 ± 0.001**	**0.418 ± 0.002**	**0.480 ± 0.004**	0.283 ± 0.007	**0.227 ± 0.007**	0.260 ± 0.012
SVD (*words* + GCN)	0.323 ± 0.001	0.385 ± 0.003	0.443 ± 0.004	0.260 ± 0.005	0.209 ± 0.009	0.249 ± 0.011
CAEME (*words* + GCN)	0.331 ± 0.001	0.395 ± 0.003	0.457 ± 0.004	0.251 ± 0.006	0.207 ± 0.009	0.235 ± 0.012
AAEME (*words* + GCN)	0.331 ± 0.001	0.392 ± 0.003	0.456 ± 0.004	0.243 ± 0.006	0.200 ± 0.008	0.228 ± 0.012
SVD (*words* + GraphSAGE)	0.338 ± 0.001	0.401 ± 0.003	0.464 ± 0.004	0.239 ± 0.006	0.194 ± 0.009	0.221 ± 0.011
CAEME (*words* + GraphSAGE)	0.323 ± 0.001	0.382 ± 0.003	0.435 ± 0.004	0.200 ± 0.006	0.170 ± 0.007	0.202 ± 0.01
AAEME (*words* + GraphSAGE)	0.343 ± 0.001	0.406 ± 0.003	0.468 ± 0.004	0.238 ± 0.007	0.178 ± 0.008	0.209 ± 0.011
State-of-the-art Approaches						
WBSR (Top-1 RUSSE'2020 for nouns)	**0.333 ± 0.002**	**0.393 ± 0.003**	**0.436 ± 0.003**	**0.252 ± 0.006**	**0.206 ± 0.011**	**0.252 ± 0.013**
WBSR, no search engine features	0.251 ± 0.001	0.309 ± 0.003	0.344 ± 0.004	0.231 ± 0.006	0.180 ± 0.008	0.222 ± 0.009
hypo2path rev [15]	0.264 ± 0.001	0.283 ± 0.003	0.238 ± 0.007	0.173 ± 0.005	0.104 ± 0.008	0.118 ± 0.009
hypo2path [15]	0.252 ± 0.002	0.261 ± 0.002	0.208 ± 0.006	0.162 ± 0.005	0.093 ± 0.006	0.067 ± 0.008
hypo2path transformer	0.218 ± 0.002	0.229 ± 0.002	0.057 ± 0.002	0.140 ± 0.003	0.120 ± 0.006	0.100 ± 0.008
TaxoExpan [71]	*0.004 ± 0.000*	*0.003 ± 0.000*	*0.054 ± 0.002*	*0.001 ± 0.000*	*0.000 ± 0.000*	*0.000 ± 0.000*

Table 13

MAP scores for the taxonomy enrichment methods for the Russian datasets. Numbers **in bold** show the best model within the category, **underlined** numbers denote the best score across all the models

Method	Nouns		Verbs	
	Non-restricted	Restricted	Non-restricted	Restricted
Baseline				
fastText [8]	**0.414 ± 0.001**	**0.549 ± 0.006**	0.296 ± 0.004	0.389 ± 0.011
word2vec [50]	0.263 ± 0.001	0.427 ± 0.006	**0.343 ± 0.004**	**0.445 ± 0.013**
DWRank (Word Embeddings)				
fastText [8]	**0.419 ± 0.001**	**0.572 ± 0.005**	**0.337 ± 0.003**	**0.428 ± 0.007**
word2vec [50]	0.296 ± 0.002	0.569 ± 0.005	0.250 ± 0.003	0.284 ± 0.011
DWRank (Meta-embeddings based on Word Embeddings)				
concat (*words*)	0.422 ± 0.001	0.589 ± 0.005	0.351 ± 0.004	0.426 ± 0.009
SVD (*words*)	0.461 ± 0.001	**0.600 ± 0.005**	**0.426 ± 0.005**	**0.475 ± 0.010**
CAEME (*words*)	0.400 ± 0.001	0.561 ± 0.005	0.342 ± 0.003	0.416 ± 0.008
AAEME (*words*)	0.456 ± 0.001	0.582 ± 0.005	0.368 ± 0.004	0.442 ± 0.009
CAEME triplet loss (*words*)	0.449 ± 0.001	0.581 ± 0.005	0.374 ± 0.003	0.427 ± 0.010
AAEME triplet loss (*words*)	**0.474 ± 0.001**	0.593 ± 0.006	0.399 ± 0.004	0.449 ± 0.010
DWRank (Graph embeddings)				
GCN [36]	0.183 ± 0.001	0.306 ± 0.005	0.220 ± 0.003	0.287 ± 0.009
GAT [82]	0.142 ± 0.001	0.318 ± 0.004	0.000 ± 0.000	0.000 ± 0.000
GraphSAGE [31]	0.176 ± 0.001	0.348 ± 0.005	0.181 ± 0.003	0.226 ± 0.008
TADW [84]	**0.417 ± 0.001**	**0.562 ± 0.005**	**0.328 ± 0.003**	**0.423 ± 0.008**
Poincaré [54] (top-5 fastText associates)	0.336 ± 0.001	0.476 ± 0.005	0.244 ± 0.004	0.339 ± 0.009
node2vec [29] (top-5 fastText associates)	0.343 ± 0.002	0.477 ± 0.005	0.226 ± 0.003	0.322 ± 0.010
HOPE [60]	0.000 ± 0.000	0.000 ± 0.000	0.003 ± 0.001	0.003 ± 0.001
DWRank (Meta-embeddings based on Word and Graph Embeddings)				
SVD (*words* + node2vec)	0.367 ± 0.001	0.521 ± 0.005	0.252 ± 0.003	0.351 ± 0.010
CAEME (*words* + node2vec)	0.370 ± 0.001	0.533 ± 0.005	0.267 ± 0.003	0.362 ± 0.010
AAEME (*words* + node2vec)	0.373 ± 0.001	0.529 ± 0.005	0.272 ± 0.003	0.358 ± 0.010
SVD (*words* + TADW)	**0.469 ± 0.001**	**0.604 ± 0.006**	**0.394 ± 0.005**	**0.455 ± 0.010**
CAEME (*words* + TADW)	0.429 ± 0.001	0.571 ± 0.005	0.349 ± 0.003	0.437 ± 0.009
AAEME (*words* + TADW)	0.461 ± 0.001	0.584 ± 0.005	0.362 ± 0.004	0.439 ± 0.009
SVD (*words* + GCN)	0.395 ± 0.001	0.554 ± 0.005	0.291 ± 0.004	0.356 ± 0.009
CAEME (*words* + GCN)	0.389 ± 0.001	0.544 ± 0.005	0.302 ± 0.003	0.381 ± 0.008
AAEME (*words* + GCN)	0.386 ± 0.001	0.545 ± 0.006	0.295 ± 0.004	0.365 ± 0.008
SVD (*words* + GraphSAGE)	0.410 ± 0.001	0.603 ± 0.005	0.336 ± 0.004	0.426 ± 0.009
CAEME (*words* + GraphSAGE)	0.321 ± 0.001	0.541 ± 0.005	0.266 ± 0.004	0.345 ± 0.007
AAEME (*words* + GraphSAGE)	0.409 ± 0.001	0.577 ± 0.006	0.323 ± 0.004	0.419 ± 0.009
State-of-the-art Approaches				
WBSR (Top-1 RUSSE'2020 for nouns)	**0.393 ± 0.002**	**0.552 ± 0.005**	0.293 ± 0.004	0.428 ± 0.010
WBSR, no search engine features	0.369 ± 0.002	0.497 ± 0.005	0.267 ± 0.004	0.387 ± 0.009
Top-1 RUSSE'2020 for verbs: [20]	0.288 ± 0.001	0.418 ± 0.006	**0.341 ± 0.004**	**0.452 ± 0.012**
hypo2path [15]	0.061 ± 0.000	0.097 ± 0.002	0.137 ± 0.003	0.174 ± 0.009
hypo2path rev [15]	0.246 ± 0.001	0.342 ± 0.006	0.151 ± 0.003	0.194 ± 0.008
hypo2path rev transformer [15]	0.234 ± 0.001	0.331 ± 0.004	0.152 ± 0.003	0.201 ± 0.008
TaxoExpan [71]	*0.007 ± 0.000*	*0.006 ± 0.001*	*0.009 ± 0.001*	*0.008 ± 0.002*

References

[1] R. Aly, S. Acharya, A. Ossa, A. Köhn, C. Biemann and A. Panchenko, Every child should have parents: A taxonomy refinement algorithm based on hyperbolic term embeddings, in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics*, Florence, Italy, 2019, pp. 4811–4817. https://www.aclweb.org/anthology/P19-1474. doi:10.18653/v1/P19-1474.

[2] N. Arefyev, M. Fedoseev, A. Kabanov and V. Zizov, Word2vec not dead: Predicting hypernyms of co-hyponyms is better than reading definitions, in: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference "Dialogue"*, 2020.

[3] I. Balazevic, C. Allen and T. Hospedales, TuckER: Tensor factorization for knowledge graph completion, in: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics*, Hong Kong, China, 2019, pp. 5185–5194. https://www.aclweb.org/anthology/D19-1522. doi:10.18653/v1/D19-1522.

[4] M. Bansal, K. Gimpel and K. Livescu, Tailoring continuous word representations for dependency parsing, in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014, pp. 809–815. doi:10.3115/v1/P14-2131.

[5] G. Berend, M. Makrai and P. Földiák, 300-sparsans at SemEval-2018 task 9: Hypernymy as interaction of sparse attributes, in: *Proceedings of the 12th International Workshop on Semantic Evaluation, Association for Computational Linguistics*, New Orleans, Louisiana, 2018, pp. 928–934. https://www.aclweb.org/anthology/S18-1152. doi:10.18653/v1/S18-1152.

[6] T. Berners-Lee, J. Hendler and O. Lassila, The semantic web, *Scientific american* **284**(5) (2001), 34–43. doi:10.1038/scientificamerican0501-34.

[7] G. Bernier-Colborne and C. Barrière, CRIM at SemEval-2018 task 9: A hybrid approach to hypernym discovery, in: *Proceedings of the 12th International Workshop on Semantic Evaluation, Association for Computational Linguistics*, New Orleans, Louisiana, 2018, pp. 725–731. https://www.aclweb.org/anthology/S18-1116. doi:10.18653/v1/S18-1116.

[8] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, Enriching word vectors with subword information, *Transactions of the Association for Computational Linguistics* **5** (2017), 135–146. doi:10.1162/tacl_a_00051.

[9] D. Bollegala and C. Bao, Learning word meta-embeddings by autoencoding, in: *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 1650–1661.

[10] F. Bond, P. Vossen, J. McCrae and C. Fellbaum, CILI: The collaborative interlingual index, in: *Proceedings of the 8th Global WordNet Conference (GWC), Global Wordnet Association*, Bucharest, Romania, 2016, pp. 50–57, https://www.aclweb.org/anthology/2016.gwc-1.9.

[11] G. Bordea, P. Buitelaar, S. Faralli and R. Navigli, SemEval-2015 task 17: Taxonomy Extraction Evaluation (TExEval), in: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics*, Denver, Colorado, 2015, pp. 902–910. https://www.aclweb.org/anthology/S15-2151. doi:10.18653/v1/S15-2151.

[12] G. Bordea, E. Lefever and P. Buitelaar, SemEval-2016 task 13: Taxonomy Extraction Evaluation (TExEval-2), in: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics*, San Diego, California, 2016, pp. 1081–1091. https://www.aclweb.org/anthology/S16-1168. doi:10.18653/v1/S16-1168.

[13] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston and O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: *Proceedings of the 26th International Conference on Neural Information Processing Systems–Volume 2, NIPS'13*, Curran Associates Inc., Red, Hook, NY, USA, 2013, pp. 2787–2795.

[14] J. Camacho-Collados, C. Delli Bovi, L. Espinosa-Anke, S. Oramas, T. Pasini, E. Santus, V. Shwartz, R. Navigli and H. Saggion, SemEval-2018 task 9: Hypernym discovery, in: *Proceedings of the 12th International Workshop on Semantic Evaluation, Association for Computational Linguistics*, New Orleans, Louisiana, 2018, pp. 712–724. https://www.aclweb.org/anthology/S18-1115. doi:10.18653/v1/S18-1115.

[15] Y. Cho, J.D. Rodriguez, Y. Gao and K. Erk, Leveraging WordNet paths for neural hypernym prediction, in: *Proceedings of the 28th International Conference on Computational Linguistics, International Committee on Computational Linguistics*, Barcelona, Spain (Online), 2020, pp. 3007–3018. https://www.aclweb.org/anthology/2020.coling-main.268. doi:10.18653/v1/2020.coling-main.268.

[16] S. Chowdhury, C. Zhang, P.S. Yu and Y. Luo, Mixed Pooling Multi-View Attention Autoencoder for Representation Learning in Healthcare, 2019, arXiv preprint 1910.06456.

[17] P. Cimiano and J. Völker, Text2Onto: A framework for ontology learning and data-driven change discovery, in: *Proceedings of the 10th International Conference on Natural Language Processing and Information Systems, NLDB'05*, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 227–238. ISBN 3540260315. doi:10.1007/11428817_21.

[18] J. Coates and D. Bollegala, Frustratingly Easy Meta-Embedding–Computing Meta-Embeddings by Averaging Source Word Embeddings, 2018, arXiv preprint 1804.05262.

[19] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer and V. Stoyanov, Unsupervised cross-lingual representation learning at scale, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, 2020, pp. 8440–8451. https://www.aclweb.org/anthology/2020.acl-main.747. doi:10.18653/v1/2020.acl-main.747.

[20] D. Dale, A simple solution for the taxonomy enrichment task: Discovering hypernyms using nearest neighbor search, in: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference "Dialogue"*, 2020.

[21] G. De Giacomo and M. Lenzerini, TBox and ABox Reasoning in Expressive Description Logics, 1996, pp. 37–48.

[22] D. Dessì, F. Osborne, D. Reforgiato Recupero, D. Buscaldi and E. Motta, Generating knowledge graphs by employing natural language processing and machine learning techniques within the scholarly domain, *Future Generation Computer Systems* **116** (2021), 253–264. https://www.sciencedirect.com/science/article/pii/S0167739X2033003X. doi:10.1016/j.future.2020.10.026.

[23] T. Dettmers, P. Minervini, P. Stenetorp and S. Riedel, Convolutional 2d knowledge graph embeddings, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, 2018.

[24] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. https://www.aclweb.org/anthology/N19-1423. doi:10.18653/v1/N19-1423.

[25] L. Espinosa-Anke, F. Ronzano and H. Saggion, TALN at SemEval-2016 task 14: Semantic taxonomy enrichment via sense-based embeddings, in: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics*, San Diego, California, 2016, pp. 1332–1336. https://www.aclweb.org/anthology/S16-1208. doi:10.18653/v1/S16-1208.

[26] M. Fares, A. Kutuzov, S. Oepen and E. Velldal, Word vectors, reuse, and replicability: Towards a community repository of large-text resources, in: *Proceedings of the 21st Nordic Conference on Computational Linguistics, Association for Computational Linguistics*, Gothenburg, Sweden, 2017, pp. 271–276. https://www.aclweb.org/anthology/W17-0237.

[27] R. Goel, S.M. Kazemi, M. Brubaker and P. Poupart, Diachronic embedding for temporal knowledge graph completion, in: *Proceedings of the AAAI Conference on Artificial Intelligence 34(04)*, 2020, pp. 3988–3995, https://ojs.aaai.org/index.php/AAAI/article/view/5815. doi:10.1609/aaai.v34i04.5815.

[28] A. Gómez-Pérez and O. Corcho, Ontology languages for the semantic web, *IEEE Intelligent systems* **17**(1) (2002), 54–60. doi:10.1109/5254.988453.

[29] A. Grover and J. Leskovec, node2vec: Scalable feature learning for networks, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[30] C. Gülcehre, M. Denil, M. Malinowski, A. Razavi, R. Pascanu, K. Hermann, P. Battaglia, V. Bapst, D. Raposo, A. Santoro and N.D. Freitas, Hyperbolic Attention Networks, 2019, arXiv preprint 1805.09786.

[31] W.L. Hamilton, R. Ying and J. Leskovec, Inductive representation learning on large graphs, in: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

[32] K. Han, P. Yang, S. Mishra and J. Diesner, WikiCSSH: Extracting computer science subject headings from Wikipedia, in: *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, Springer, 2020, pp. 207–218. doi:10.1007/978-3-030-55814-7_17.

[33] D. Jurgens and M.T. Pilehvar, SemEval-2016 task 14: Semantic taxonomy enrichment, in: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics*, San Diego, California, 2016, pp. 1092–1102. https://www.aclweb.org/anthology/S16-1169. doi:10.18653/v1/S16-1169.

[34] T.N. Kipf and M. Welling, *Variational Graph Auto-Encoders, NIPS Workshop on Bayesian Deep Learning*, 2016.

[35] T.N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, in: *International Conference on Learning Representations (ICLR)*, 2017.

[36] T.N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint 1609.02907.

[37] M. Kunilovskaya, A. Kutuzov and A. Plum, Taxonomy enrichment: Linear hyponym-hypernym projection vs synset ID classification, in: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference "Dialogue"*, 2020.

[38] A. Kutuzov and E. Kuzmenko, WebVectors: A toolkit for building web interfaces for vector semantic models, in: *Analysis of Images, Social Networks and Texts: 5th International Conference, AIST 2016*, Yekaterinburg, Russia, April 7–9, 2016, Revised Selected Papers, D.I. Ignatov, M.Y. Khachay, V.G. Labunets, N. Loukachevitch, S.I. Nikolenko, A. Panchenko, A.V. Savchenko and K. Vorontsov, eds, Springer International Publishing, Cham, 2017, pp. 155–161. ISBN 978-3-319-52920-2. doi:10.1007/978-3-319-52920-2_15.

[39] T. Lacroix, G. Obozinski and N. Usunier, Tensor decompositions for temporal knowledge base completion, in: *8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa, Ethiopia, April 26–30, 2020, OpenReview.net 2020. https://openreview.net/forum?id=rke2P1BFwS.

[40] T. Lacroix, N. Usunier and G. Obozinski, Canonical tensor decomposition for knowledge base completion, in: *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, eds, Proceedings of Machine Learning Research, Vol. 80, PMLR, Stockholmsmässan, Stockholm Sweden, 2018, pp. 2863–2872, http://proceedings.mlr.press/v80/lacroix18a.html.

[41] N. Li, Z. Bouraoui and S. Schockaert, Ontology completion using graph convolutional networks, in: *The Semantic Web – ISWC 2019*, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois and F. Gandon, eds, Springer International Publishing, Cham, 2019, pp. 435–452. ISBN 978-3-030-30793-6. doi:10.1007/978-3-030-30793-6_25.

[42] N. Liu, X. Huang, J. Li and X. Hu, On interpretation of network embedding via taxonomy induction, in: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &*, Data Mining, 2018, pp. 1812–1820. doi:10.1145/3219819.3220001.

[43] N.V. Loukachevitch, G. Lashevich, A.A. Gerasimova, V.V. Ivanov and B.V. Dobrov, Creating Russian wordnet by conversion, in: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference "Dialogue*, 2016, pp. 405–415.

[44] Y. Luan, L. He, M. Ostendorf and H. Hajishirzi, Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction, in: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics*, Brussels, Belgium, 2018, pp. 3219–3232, https://www.aclweb.org/anthology/D18-1360. doi:10.18653/v1/D18-1360.

[45] T. Luong, H. Pham and C.D. Manning, Effective approaches to attention-based neural machine translation, in: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics*, Lisbon, Portugal, 2015, pp. 1412–1421, https://www.aclweb.org/anthology/D15-1166. doi:10.18653/v1/D15-1166.

[46] I. Makarov, M. Makarov and D. Kiselev, Fusion of text and graph information for machine learning problems on networks, *PeerJ Computer Science* **7** (2021), 00.

[47] A. Maldonado and F. Klubička, ADAPT at SemEval-2018 task 9: Skip-gram word embeddings for unsupervised hypernym discovery in specialised corpora, in: *Proceedings of the 12th International Workshop on Semantic Evaluation, Association for Computational Linguistics*, New Orleans, Louisiana, 2018, pp. 924–927. https://www.aclweb.org/anthology/S18-1151. doi:10.18653/v1/S18-1151.

[48] D.L. McGuinness and A. Borgida, Explaining subsumption in description logics, *IJCAI (1)* **3** (1995), 00.

[49] T. Mikolov, K. Chen, G. Corrado and J. Dean, Efficient estimation of word representations in vector space, in: *1st International Conference on Learning Representations, ICLR 2013*, Scottsdale, Arizona, USA, May 2–4, 2013, Y. Bengio and Y. LeCun, eds, Workshop Track Proceedings, 2013. http://arxiv.org/abs/1301.3781.

[50] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado and J. Dean, Distributed representations of words and phrases and their compositionality, in: *Advances in Neural Information Processing Systems 26*, C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K.Q. Weinberger, eds, Curran Associates, Inc., 2013, pp. 3111–3119.

[51] G.A. Miller, WordNet: A lexical database for English, *Communications of the ACM* **38**(11) (1995), 39–41. doi:10.1145/219717.219748.

[52] G.A. Miller, *WordNet: An Electronic Lexical Database*, MIT press, 1998.

[53] J.O. Neill and D. Bollegala, Meta-embedding as auxiliary task regularization, 2018, arXiv preprint 1809.05886.

[54] M. Nickel and D. Kiela, Poincaré embeddings for learning hierarchical representations, in: *Advances in Neural Information Processing Systems 30*, I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds, Curran Associates, Inc., 2017, pp. 6341–6350.

[55] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, A review of relational machine learning for knowledge graphs, *Proceedings of the IEEE* **104** (2015), 00. doi:10.1109/JPROC.2015.2483592.

[56] I. Nikishina, V. Logacheva, A. Panchenko and N. Loukachevitch, RUSSE'2020: Findings of the first taxonomy enrichment task for the Russian language, in: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference "Dialogue"*, 2020.

[57] I. Nikishina, N. Loukachevitch, V. Logacheva and A. Panchenko, Exploring graph-based representations for taxonomy enrichment, in: *Proceedings of the 11th Global Wordnet Conference, Global Wordnet Association*, Africa (UNISA), 2021, pp. 126–136, University of South. https://www.aclweb.org/anthology/2021.gwc-1.15.

[58] I. Nikishina, A. Panchenko, V. Logacheva and N. Loukachevitch, Studying taxonomy enrichment on diachronic WordNet versions, in: *Proceedings of the 28th International Conference on Computational Linguistics, Association for Computational Linguistics*, Barcelona, Spain, 2020.

[59] F. Osborne and E. Motta, Klink-2: Integrating Multiple Web Sources to Generate Semantic Topic Networks, 2015. ISBN 978-3-319-25006-9. doi:10.1007/978-3-319-25007-6_24.

[60] M. Ou, P. Cui, J. Pei, Z. Zhang and W. Zhu, Asymmetric transitivity preserving graph embedding, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1105–1114. doi:10.1145/2939672.2939751.

[61] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic web* **8**(3) (2017), 489–508. doi:10.3233/SW-160218.

[62] J. Pennington, R. Socher and C. Manning, GloVe: Global vectors for word representation, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics*, Doha, Qatar, 2014, pp. 1532–1543, https://www.aclweb.org/anthology/D14-1162. doi:10.3115/v1/D14-1162.

[63] B. Perozzi, R. Al-Rfou and S. Skiena, Deepwalk: Online learning of social representations, in: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710. doi:10.1145/2623330.2623732.

[64] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, Deep contextualized word representations, in: *Proceedings of the 2018 Conference of the North American*, Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 2227–2237. https://www.aclweb.org/anthology/N18-1202. doi:10.18653/v1/N18-1202.

[65] J. Pocostales, NUIG-UNLP at SemEval-2016 task 13: A simple word embedding-based approach for taxonomy extraction, in: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics*, San Diego, California, 2016, pp. 1298–1302. https://www.aclweb.org/anthology/S16-1202. doi:10.18653/v1/S16-1202.

[66] D. Pujary, C. Thorne and W. Aziz, Disease normalization with graph embeddings, in: *Proceedings of SAI Intelligent Systems Conference*, Springer, 2020, pp. 209–217.

[67] A. Rossi, D. Firmani, A. Matinata, P. Merialdo and D. Barbosa, Knowledge Graph Embedding for Link Prediction: A Comparative Analysis, 2020, arXiv preprint 2002.00819.

[68] M. Ryabinin, S. Popov, L. Prokhorenkova and E. Voita, Embedding Words in Non-Vector Space with Unsupervised Graph Learning, 2020, arXiv preprint 2010.02598.

[69] D. Schlechtweg, B. McGillivray, S. Hengchen, H. Dubossarsky and N. Tahmasebi, SemEval-2020 task 1: Unsupervised lexical semantic change detection, in: *Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics*, Barcelona, 2020, pp. 1–23, (online), https://www.aclweb.org/anthology/2020.semeval-1.1.

[70] C. Shang, Y. Tang, J. Huang, J. Bi, X. He and B. Zhou, End-to-end structure-aware convolutional networks for knowledge base completion, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 3060–3067.

[71] J. Shen, Z. Shen, C. Xiong, C. Wang, K. Wang and J. Han, TaxoExpan: Self-supervised taxonomy expansion with position-enhanced graph neural network, in: *Proceedings of the Web Conference 2020*, 2020, pp. 486–497. doi:10.1145/3366423.3380132.

[72] B. Shi and T. Weninger, ProjE: Embedding projection for knowledge graph completion, in: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17*, AAAI Press, 2017, pp. 1236–1242.

[73] V. Shwartz, Y. Goldberg and I. Dagan, Improving hypernymy detection with an integrated path-based and distributional method, 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. 2389–2398. https://www.aclweb.org/anthology/P16-1226. doi:10.18653/v1/P16-1226.

[74] M. Straka, J. Straková Tokenizing and P.O.S. Tagging, Lemmatizing and parsing UD 2.0 with UDPipe, in: *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Association for Computational Linguistics*, Vancouver, Canada, 2017, pp. 88–99, http://www.aclweb.org/anthology/K/K17/K17-3009.pdf. doi:10.18653/v1/K17-3009.

[75] I. Sutskever, O. Vinyals and Q.V. Le, Sequence to sequence learning with neural networks, *Advances in neural information processing systems* **27** (2014), 3104–3112.

[76] H. Tanev and A. Rotondi, Deftor at SemEval-2016 task 14: Taxonomy enrichment using definition vectors, in: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics*, San Diego, California, 2016, pp. 1342–1345. https://www.aclweb.org/anthology/S16-1210. doi:10.18653/v1/S16-1210.

[77] M. Tikhomirov and N. Loukachevitch, Meta-embeddings in taxonomy enrichment task, in: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference "Dialogue"*, 2021.

[78] M. Tikhomirov, N. Loukachevitch and E. Parkhomenko, Combined approach to hypernym detection for thesaurus enrichment, in: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference "Dialogue"*, 2020.

[79] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L.U. Kaiser and I. Polosukhin, Attention is all you need, in: *Advances in Neural Information Processing Systems*, I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds, Vol. 30, Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[80] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L.U. Kaiser and I. Polosukhin, Attention is all you need, in: *Advances in Neural Information Processing Systems*, I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds, Vol. 30, Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[81] P. Velardi, S. Faralli and R. Navigli, OntoLearn reloaded: A graph-based algorithm for taxonomy induction, *Computational Linguistics* **39**(3) (2013), 665–707. https://www.aclweb.org/anthology/J13-3007. doi:10.1162/COLI_a_00146.

[82] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, *Graph Attention Networks, ICLR*, 2018.

[83] G.I. Winata, Z. Lin and P. Fung, Learning multilingual meta-embeddings for code-switching named entity recognition, in: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, 2019, pp. 181–186. doi:10.18653/v1/W19-4320.

[84] C. Yang, Z. Liu, D. Zhao, M. Sun and E. Chang, Network representation learning with rich text information, in: *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[85] W. Yin and H. Schütze, Learning word meta-embeddings, in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1351–1360. doi:10.18653/v1/P16-1128.

[86] W. Yu, C. Zheng, W. Cheng, C.C. Aggarwal, D. Song, B. Zong, H. Chen and W. Wang, Learning deep network representations with adversarially regularized autoencoders, in: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &Amp; Data Mining, KDD '18*, Association for Computing Machinery, New York, NY, USA, 2018, pp. 2663–2671. ISBN 9781450355520. doi:10.1145/3219819.3220000.