Taxonomy Enrichment with Text and Graph Vector Representations

Irina Nikishina [a,*], Mikhail Tikhomirov [b], Varvara Logacheva [a], Yuriy Nazarov, 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

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 GCN, Poincaré embeddings, node2vec 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 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, knowledge graph completion, graph vector representations, word embeddings, graph convolutional auto-encoder, sequence-to-sequence architecture

1. Introduction

The central idea of Semantic Web is to make the content of the Internet pages machine-interpretable. For that, the web-pages should be provided with links to *ontologies* [1, 2] — 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 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.

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

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

changing, the ontologies need to be constantly updated to stay relevant. There currently exist comprehensive knowledge bases (Freebase, DBPedia, 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 suggest 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 [3] 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 [4, 5]. 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 suggest a number of ways to incorporate various sources of information.

First, we suggest 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.

This work is an extended version of the work described in [6–8]. The novelty of this particular work compared to the previous versions 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 diachronic evolution of wordnets for English and Russian, extending the monolingual setup of the RUSSE'2020 shared task [9] 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.[1]

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. In Section 9 we analyse the shortcomings of our methods. Finally, in Section 10 we suggest possible applications of our methods and conclude in Section 11.

2. Task Formulation and Definitions

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

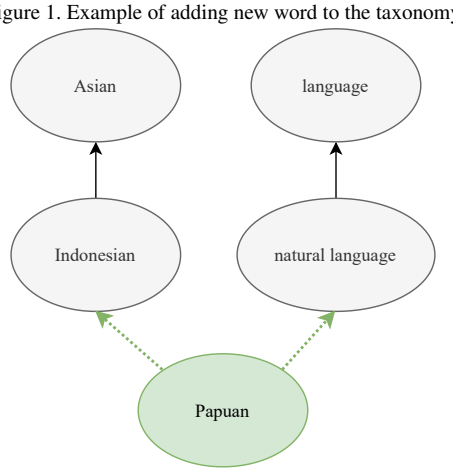
2.1. Task Formulation

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, 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 Figure 1, a word may have multiple hypernyms.

Figure 1. Example of adding new word to the taxonomy



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 the 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 [3, 9].

2.2. Definitions

Knowledge graph is a knowledge base that uses a graph-structured data model or topology to integrate data. Formally, 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.

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

Knowledge graph completion is the task of completing 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 is a tree structure where words or word phrases are represented as nodes and the edges between them denote hypernym (subclass-of or instance-of) relationship. Formally, taxonomy T can be represented as graph G, taxonomic entities (synsets) form the entity set E, and R contains one element, which is the hypo-hypernym relation.

Synset is a set $S \subseteq \{s_1, ..., s_n\}$ of synonyms (words or phrases) corresponding to a concept, which convey the concept in texts. Synset is the major element constituting taxonomy, also denotes a node in a taxonomy graph.

Query words *(new words)* — words to be added to the taxonomy, usually manually collected by experts. In this paper we use the following notation for this set $Q \subseteq \{q_1, ..., q_n\}$.

Taxonomy enrichment is a task aimed at associating each new word $q \in Q$, which is not yet included into 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 [10, 11]. Most of them apply low-dimensional graph embeddings [12, 13], deep learning architectures like autoencoders [14] or graph convolutional networks [15–17]. Another group of approaches makes use of tensor decomposition approaches: Tucker decomposition [18] and Canonical Polyadic decomposition (CANDECOMP/PARAFAC) [19, 20]. Another task, which is closely related to the taxonomy creation is the Knowledge base construction. There exists multiple approaches solving the problem: Text2Onto [21] a pillar language-independent approach which applies user interaction or fully automated methods [22, 23] 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 [24]: 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 [25–27], 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 [4] 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 [5] 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 [9], 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 [28] 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 [29] or ranked all hypernyms by suitability for a particular word [30]. They also used a range of additional resources, such as Wiktionary, dictionaries, additional corpora [31]. Interestingly, only one of the well-performing models [32] used context-informed embeddings (BERT) 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 [24] most of participants use word embeddings. For instance, Bernier-Colborne and Barriere [33] predict the likelihood of the relationship between an input word and a candidate using word2vec embeddings. Berend et al. [34] use Word2vec vectors as features of a logistic regression classifier. Maldonado and Klubicka [35] simply consider top-10 closest associates from the Skip-gram word2vec model as hypernym candidates. Pre-trained GloVe embeddings [36] are also used to initialize embeddings for an LSTM-based Hypernymy Detection model [37].

Participants also solve the SemEval-2016 Task 13 on taxonomy induction with word embeddings [38]: 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 [39] 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 [3] and RUSSE'2020 [9] 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 [4]. Another option is to train word2vec embeddings from scratch and cast the task as a classification problem [29]. Some participants compare the approach based on XLM-R model [40] with the word2vec "hypernyms of co-hyponyms" method [31]. 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 [9, 24].

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 [41], and in medical domain [42].

Coates et al. [43] 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 [44]. Autoencoders [45] 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 [46] the authors investigated the per-

formance 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 [41], classification in healthcare [42], named-entity recognition [46, 47], sentiment analysis [46], word similarity and analogy tasks [43–45]. 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 [48] 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 [48] have also been tested on tasks related to the taxonomy enrichment.

For instance, node2vec embeddings [49] are used for taxonomy induction among other network embeddings [50]. In [39], 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, [39] 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 [51] and apply ELMO embeddings [52] to predict entities and their relations for the knowledge graph. Other approaches [53] utilize a combination of ELMO, Poincare and node2vec embeddings to enhance knowledge graph build upon Wikipedia.

Graph convolutional networks (GCNs) [54] as well as graph autoencoders [55] are mostly applied to the link prediction task on large knowledge bases. For example, in [56] 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 [57]. As for the taxonomy enrichment task, we are only aware of a recent approach TaxoExpan [58] which applies position-enhanced graph neural networks (GCN [59] and GAT [60]) that we also evaluate on our datasets[2].

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 [61] and RuWordNet taxonomies. It is important to mention that by using the word “diacronic” we do not imply lexical diachrony, e.g., semantic shifts [62], 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.

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

[2]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	17 043	755
WordNet1.7 - WordNet3.0	6 161	362
WordNet2.0 - WordNet3.0	2 620	193
RuWordNet1.0 - RuWordNet2.0	14 660	2 154
RUSSE'2020	2 288	525

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[3] created for this research and the code[4] 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 [28]. 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).

The RUSSE'2020 dataset was created for the Dialogue Evaluation [9] 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 [63]. This mapping is designed to make possible coordination between wordnet projects. For the Russian test sets we also provide mapping from RuWordNet to WordNet[5]. 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.

[3] https://zenodo.org/record/4279821
[4] https://github.com/skoltech-nlp/diachronic-wordnets
[5] https://doi.org/10.5281/zenodo.4969267

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	66 025	12 127	94 474	10 319	-	-
WordNet 1.7	75 804	13 214	109 195	11 088	11 551	401
WordNet 2.0	79 689	13 508	114 648	11 306	4 036	182
WordNet 2.1	81 426	13 650	117 097	11 488	2 023	158
WordNet 3.0	82 115	13 767	117 798	11 529	678	33

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]

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:

$$MAP = \frac{1}{N}\sum_{i=1}^{N} AP_i;$$
$$AP_i = \frac{1}{M}\sum_{i}^{n} prec_i \times I[y_i = 1], \quad (1)$$

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 [24]. 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,

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	41694	28425
Unique synsets	4777	2791
Query words	17475	15251
Restricted (private) nouns		
All synsets	4456	3087
Unique synsets	1376	885
Query words	1920	1720
non-restricted verbs		
All synsets	6783	4860
Unique synsets	1473	931
Query words	2872	2606
restricted (private) verbs		
All synsets	1110	821
Unique synsets	611	419
Query words	477	440

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 2a corresponds to a monosemous word and the case 2b indicates polysemy, this difference does not affect the evaluation process. We suggest 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 [9], 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 2a and 2b 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 [6]). We then suggest 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. [6] 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 suggest 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 immedidate 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 propose **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[6] (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[7] to translate query words into English and then translate hypernyms (if they are found) back into Russian.

[6] https://yandex.com
[7] https://translate.yandex.ru/

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.[8]

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.

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. [64]. 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 [65] with Luong atten-

[8] https://doi.org/10.5281/zenodo.4540717

tion [66]. 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 [64]. 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 [67]. 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 [68], word2vec [69] embeddings for English and Russian datasets and also GloVe embeddings [36] for the English dataset.

We use the fastText embeddings from the official website[9] 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 [70, 71] for both English[10] and Russian.[11] We lemmatise words and synsets for both languages with the same UDPipe [72] 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 [30]. As for the GloVe embeddings, we also use them from the official website[12] trained on Common Crawl, the vocabulary size is 840 billion tokens.

6. DWRank-Graph

DWRank method extracts the 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 [48].

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

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" [73]. Poincare 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 [39].

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 [73].

[9] https://fasttext.cc/docs/en/crawl-vectors.html

[10] http://vectors.nlpl.eu/repository/20/29.zip

[11] http://vectors.nlpl.eu/repository/20/185.zip

[12] https://nlp.stanford.edu/projects/glove/

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 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 [74]. 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 [49] 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 [75] 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 [54], Graph Attention Network [60] and GraphSAGE [76] (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 [48], "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 [77] 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) [78] is another approach that incorporates text and graph information into one vector representation. The method is based on the DeepWalk algorithm [79] 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 (HOPE)

High-Order Proximity preserved Embeddings (HOPE) [80] 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 (Poincare, node2vec, GCN, GAT, GraphSAGE, TADW, HOPE) embeddings. In addition to that, we perform Singular value decomposition (SVD) over this concatenation as suggested in [44]. This approach is referred to as **SVD**.

7.2. Autoencoded Meta-Embeddings

We suggest using two variants of autoencoders for the generation of meta-embeddings: Concatenated Autoencoded Meta-Embeddings (CAEME) and Averaged Autoencoded Meta-Embeddings (AAEME) [45]. 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:

$$E(s(w)) = h(w)$$
$$D(h(w)) = \hat{s}(w)$$
$$L_{ED} = dist(s(w), \hat{s}(w))$$

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 $s_2(w)$. After having trained autoencoders for both of them, we construct the meta-embeddings as follows. In case of CAEME, we take an L_2-normalised concate-

nation of the two source embeddings encoded with respective encoders $E_1(s_1(w))$ and $E_2(s_2(w))$:

$$m(w) = \frac{E_1(s_1(w)) \oplus E_2(s_2(w))}{||E_1(s_1(w)) \oplus E_2(s_2(w)||_2} \quad (2)$$

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 summation, 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 $E_2(s_2(w))$:

$$m(w) = \frac{E_1(s_1(w)) + E_2(s_2(w))}{||E_1(s_1(w)) + E_2(s_2(w))||_2} \quad (3)$$

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 2 instead of their concatenation.

The CAEME and AAEME decoders reconstruct the source embeddings from the same meta-embedding $m(w)$, thereby implicitly using both common and complementary information from the source embeddings.

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*:

$$L(w_a, w_p, w_n) = max(||m(w_a) - m(w_p))|| - ||m(w_a) - m(w_n))|| + margin, 0) \quad (4)$$

where ||.|| 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.

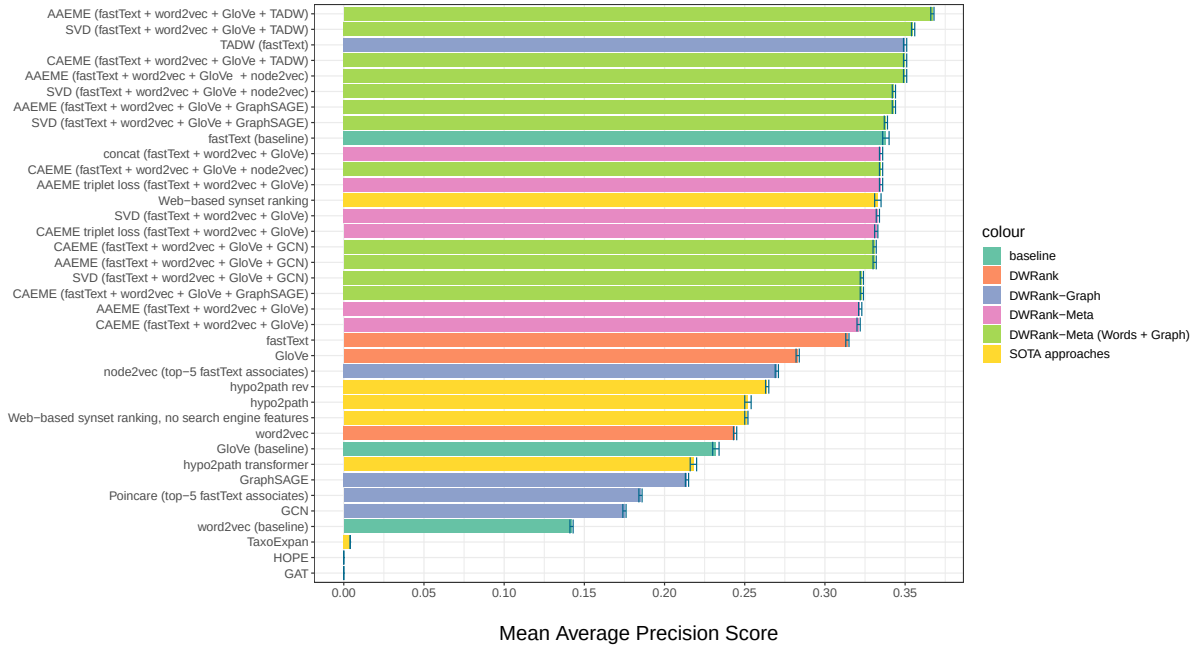
8. Experiments

In this section, we report and discuss the performance of our models on 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.

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 standard deviation on those 30 MAP values.

The results for English and Russian can be seen in Tables 5 and 6, respectively. For DWRank-Meta, we give the combined embeddings in brackets. The *words* embeddings denote the combination of fastText, word2vec, and GloVe. In addition to that, we show the performance of different methods on the nouns attribution task for English (nouns 1.6) in Figure 2 and for Russian (non-restricted nouns) in Figure 3. The colors of bars in figures correspond to different types of input embeddings (distributional, graph-based, combined, etc.) used by the methods.

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



8.1. Baseline and DWRank approaches

From the results we can see that fastText outperforms word2vec and GloVe embeddings for almost all languages and models. The low results of GloVe and word2vec embeddings on baseline and DWRank methods can be explained by data coverage: some query words can be absent from the word2vec or GloVe fixed vocabularies, whereas fastText allows generating vectors for out-of-vocabulary words.

At the same time, the performance of DWRank is quite promising. We see that it manages to improve the results over the baseline almost on all datasets for both languages. DWRank-Meta with distributional word embeddings are much more outstanding and useful, as they significantly improve the results of hypernym prediction for all datasets using SVD or AAEME autoencoder with the triplet loss.

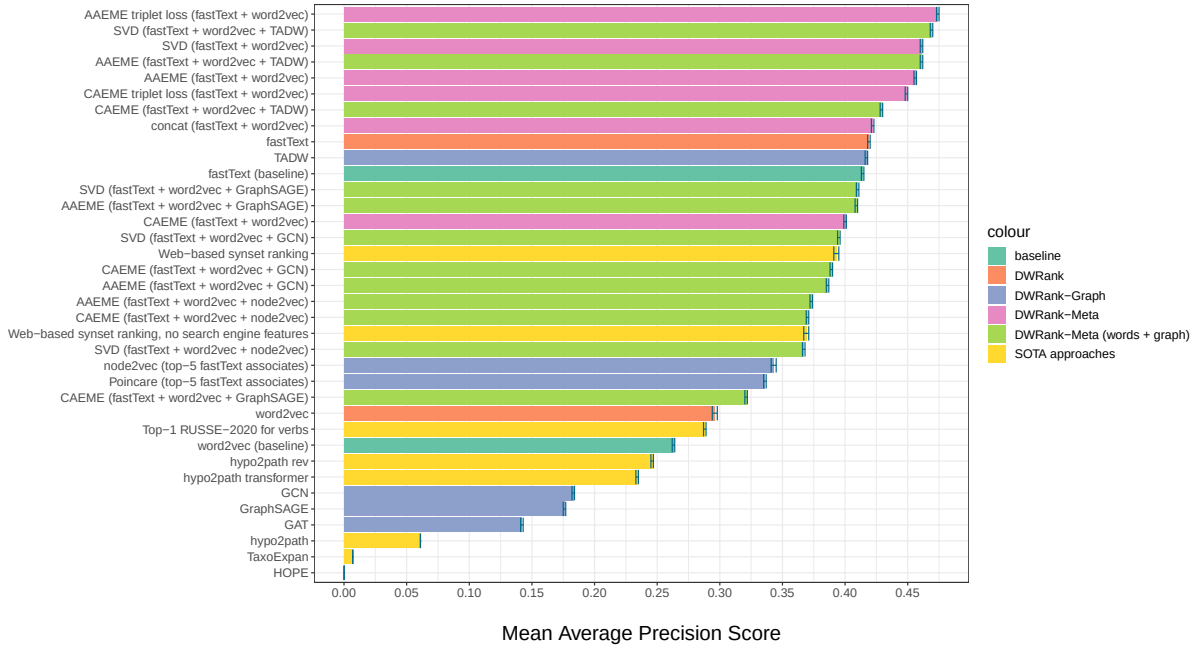
However, compared to the base forms of *AEME models, extending them with triplet loss does not significantly improve their results for all datasets.

8.2. DWRank-Graph and DwRank-Meta approaches

As for the DWRank-Graph approach, it is immediately seen from Tables 5 and 6 and Figures 2 and 3 that the graph representations underperform the distributional representations by a large margin. Even though Poincaré ball model is designed for the taxonomic structures, the absence of vector representations for the OOV words dramatically affects the results. The aggregated vector of top-5 fastText nearest neighbours (such vector is used as a proxy for an OOV word vector) usually provides a noisy or an overly general representation. Such representation is likely to yield incorrect hypernyms even if the Poincaré embeddings for the taxonomy are of perfect quality.

Graph Neural Network architectures that compute embeddings such as GCN [55], GraphSAGE [76] and GAT [60] do not outperform the majority of the approaches for neither of languages neither alone nor in the combination with word embeddings. Moreover, GAT embeddings are quite unstable and demonstrate extremely low results on the nouns_2.0 dataset and both English and Russian verbs. TADW vector representations are the only ones that demonstrate a reasonable result for both English and Russian. Their results are comparable to the word vector representations, and they are the best-performing embeddings among the graph representations. Even though GCN, GraphSAGE and GAT are also initialized with text embeddings (fastText), only TADW embeddings manage to benefit from the text and graph features. This might be explained by the fact that TADW is an extended

Figure 3. Comparison of the method performance on the Russian non-restricted dataset. Each colour denotes the method type and the embeddings type used.



version of DeepWalk and applies the skip-gram model with the pre-trained fastText representations.

Moreover, TADW manages to even improve the results on the English datasets in combination with the fastText, word2vec and GloVe embeddings, which make DWRank-Meta on such configuration the new state-of-the-art approach for English.

As for the Russian datasets, the combination with TADW does not change the scores dramatically, according to Figures 4 (c-d). The AAEME approach with the triplet loss and SVD on text meta-embeddings seem to be more efficient for both nouns and verbs. Other graph embeddings do not improve the results over the AAEME triplet loss approach either.

8.3. SOTA approaches

WBSR approach demonstrates decent results, performing on par with the DWRank methods. We can also see that using external tools such as online Machine Translation (MT) and search engines is beneficial for both languages even though it might be not easy to replicate. Its performance for different languages can vary significantly, and we have no means for quantifying this difference.

Unfortunately, the result for the implementation of *hypo2path* and its variants are not promising either. They perform closely to the graph-based embeddings approaches like node2vec, Poincare and GCN autoencoder. While looking through the candidates we discovered the following limitations that could probably affect the model performance:

- if the *hypo2path reverse* model incorrectly predicts the first hypernym the whole chain of hypernyms used as candidates can also be irrelevant;
- the *hypo2path* model starts from the root hypernyms “entity.n.01” which could be too broad as a concept for the decoder to find the correct sequence of hypernyms;
- sequence generation is not able to cover multiple connected components in the graph and predict multiple hypernyms;
- the average number of candidates per word is about 6 for nouns and about 3 for verbs, whereas other methods provide the maximum number of 10 candidates for both parts of speech.

Another ready-made approach applied to our datasets did not demonstrate decent results either. The results achieved by TaxoExpan [58] for some reason are significantly lower than the baseline. We do not consider those results credible and provided them in italics. Those results can be explained of the absence of definition that model applies during training.

Table 5

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 [6]						
fastText [68]	**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 [69]	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 [36]	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 [68]	**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 [69]	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 [36]	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 [59]	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 [60]	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 [76]	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 [78] (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**
Poincare [73] (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 [49] (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 [80]	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	**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 [64]	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 [64]	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 [58]	*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 6

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 [68]	**0.414±0.001**	**0.549±0.006**	0.296±0.004	0.389±0.011
word2vec [69]	0.263±0.001	0.427±0.006	**0.343±0.004**	**0.445±0.013**
DWRank (Word Embeddings)				
fastText [68]	**0.419±0.001**	**0.572±0.005**	**0.337±0.003**	**0.428±0.007**
word2vec [69]	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 [54]	0.183±0.001	0.306±0.005	0.220±0.003	0.287±0.009
GAT [60]	0.142±0.001	0.318±0.004	0.000±0.000	0.000±0.000
GraphSAGE [76]	0.176±0.001	0.348±0.005	0.181±0.003	0.226±0.008
TADW [78]	**0.417±0.001**	**0.562±0.005**	**0.328±0.003**	**0.423±0.008**
Poincare [73] (top-5 fastText associates)	0.336±0.001	0.476±0.005	0.244±0.004	0.339±0.009
node2vec [49] (top-5 fastText associates)	0.343±0.002	0.477±0.005	0.226±0.003	0.322±0.010
HOPE [80]	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	**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: [30]	0.288±0.001	0.418±0.006	**0.341±0.004**	**0.452±0.012**
hypo2path [64]	0.061±0.000	0.097±0.002	0.137±0.003	0.174±0.009
hypo2path rev [64]	0.246±0.001	0.342±0.006	0.151±0.003	0.194±0.008
hypo2path rev transformer [64]	0.234±0.001	0.331±0.004	0.152±0.003	0.201±0.008
TaxoExpan [58]	*0.007±0.000*	*0.006±0.001*	*0.009±0.001*	*0.008±0.002*

8.4. Overall Comparison

To sum up, we can see that the best models differ for the English and Russian datasets. DWRank-Meta on combination of word embeddings with TADW performs better for the English datasets, whereas DWRank-Meta on word embeddings only perform better on Russian.

From Figures 4 (a-d) we can see that almost all DWRank-based approaches do outperform the baseline applying different types of embeddings. Russian verbs dataset is the only exception: only the DWRank-Meta approach beats such a strong baseline. Moreover, from Figures 2 we can see that DWRank-Meta with word and graph embeddings in different combinations demostrate much higher results than DWRank-Meta with word embeddings only. On the contrary, DWRank-Meta with both word and graph embeddings is not the best-performing combination, according to the Figure 3. More specifically, we have a tie for first place between the approaches using DWRank-Meta on word vectors and their combination with TADW embeddings. Apparently, all relevant information is already embedded in the word embeddings, and structural information about taxonomy is unnecessary for the model performance on the Russian datasets.

In addition to the presented results we noticed an interesting outcome — models performance on different part-of-speech datasets. According to the Tables 5 and 6, all models except for *baseline word2vec* provide better results for nouns rather than for verbs. This phenomenon can be explained by the rich morphology of the Russian language: the query verbs are input to the models as infinitives, whereas verb infinitives are much less widespread in corpora that other verb forms. Interestingly, the baseline word2vec embeddings perform better on verbs rather than on nouns for all approaches across both languages. We can assume that lemmatisation is extremely beneficial when predicting hypernyms for verbs and is discouraged for nouns.

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.

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.

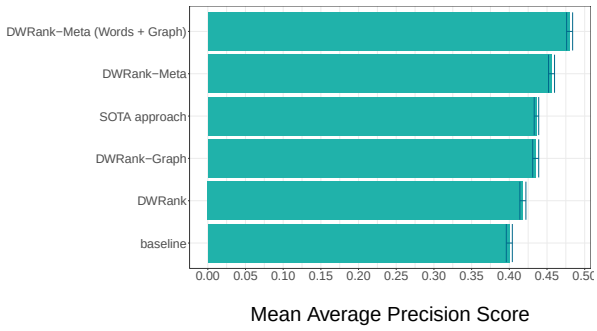
To contrast the performance of the text and graph embeddings and to demonstrate the input and the output formats of the models we present Tables 7 and 8. Underlined bold text denotes predictions of the model from the ground truth. They demonstrate the main features of the tested approaches.

From both tables 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.

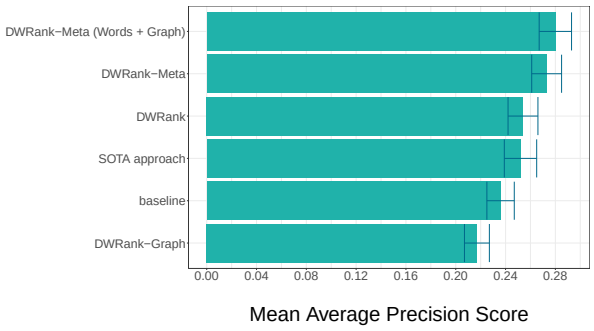
The candidates for the words “play therapy” and “eyewitness” provided by models based on text em-

Figure 4. Comparison of different approaches with the best embeddings configuration for both English and Russian languages.

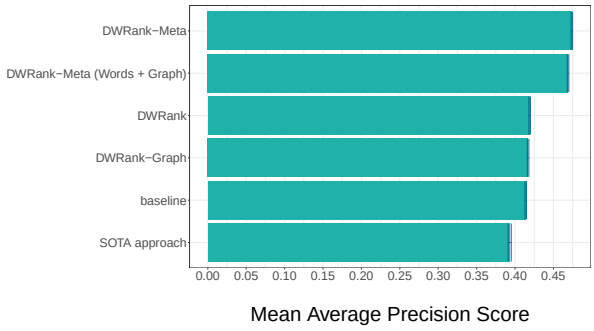
(a) English nouns 2.0 best methods from each group



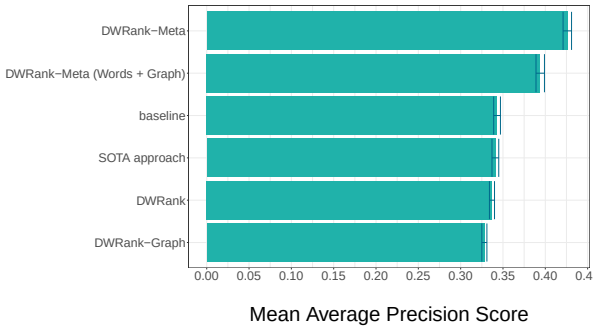
(b) English verbs 2.0 best methods from each group



(c) Russian non-restricted nouns best methods from each group



(d) Russian non-restricted verbs best methods from each group



beddings 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.

Those examples do not pretend do be the general case example, however, the do provide the idea about ranking of the results and the performance of text, graph and fusion embedding types.

9.2. Distribution of Scores

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

Figure 5 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, 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.

Table 7
Prediction noun examples from the English v 1.6-3.0 dataset.

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	Poincare	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

Figure 5. Distribution of words over the number of senses.
(a) Russian dataset (nouns and verbs) (b) English dataset (nouns and verbs)

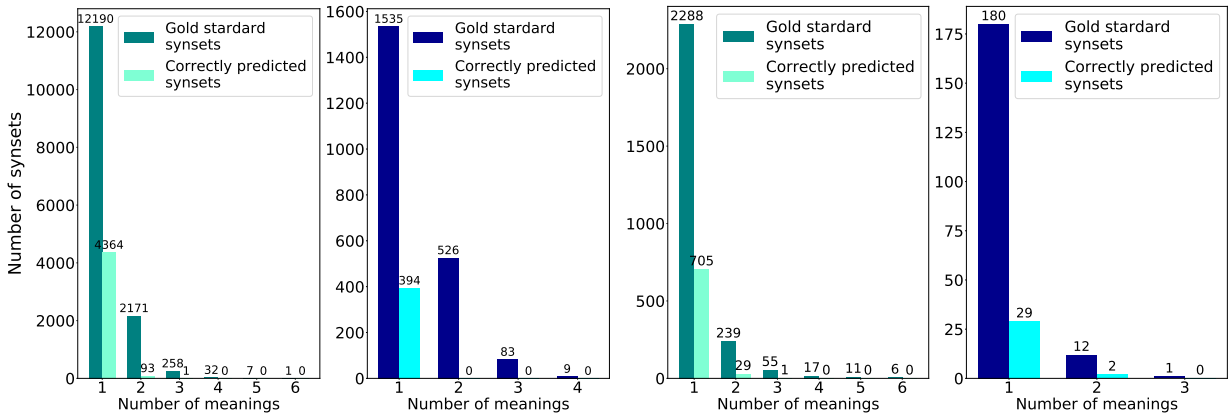
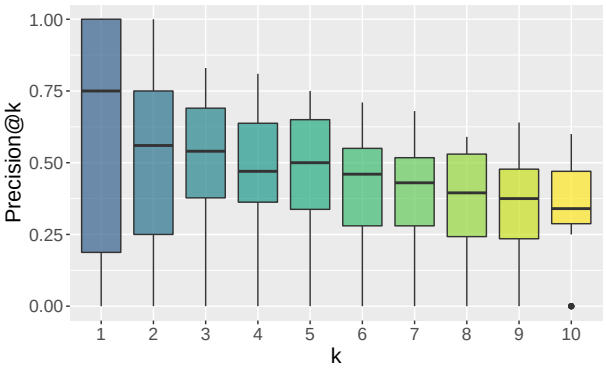


Table 8
Prediction verb examples from the English v 1.6-3.0 dataset.

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	Poincare	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

Figure 6. Manual datasets evaluation results: Precision@10.

(a) Russian dataset



(b) English dataset

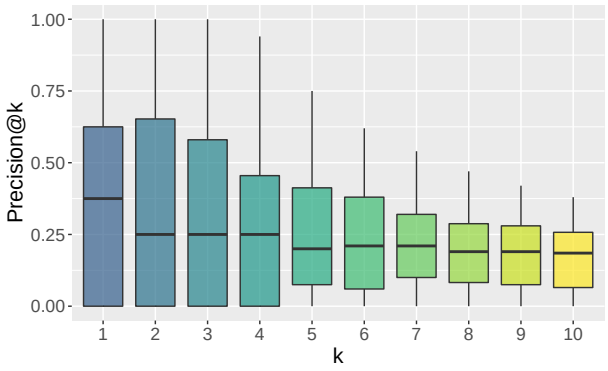


Table 9
Words selected for the manual evaluation.

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

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).

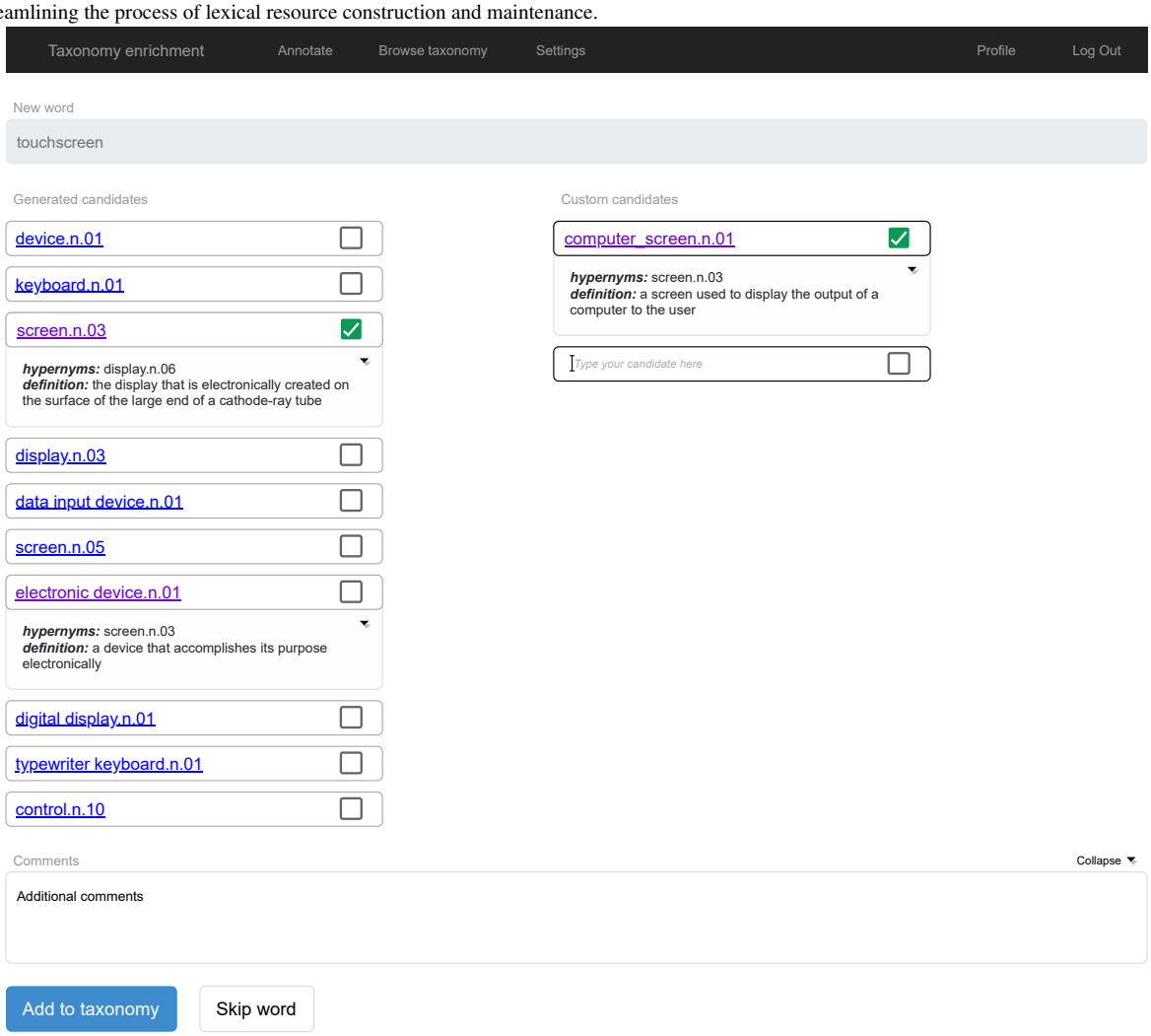
Type 5. Unaccounted senses in the gold standard datasets, inaccuracies in the manual annotation:

- 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 9). 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 Figure 6. We can see that even for words with MAP below 1 our model manages to extract useful hypernyms.

Figure 7. A mockup of an annotation system using automatically generated candidates based on the methods investigated in this article. The system is supposed to expose to the expert linguist/lexicographer the most likely candidates to the addition to a semantic taxonomy therefore streamlining the process of lexical resource construction and maintenance.



10. Application of the Developed Taxonomy Enrichment Methods

In this section, we describe a potential application of the developed taxonomy enrichment methods. Namely we describe how they could facilitate the daily routine of lexicographers who perform construction and maintenance of lexical resoures.

The achieved results demonstrate that for the large taxonomies like WordNet or RuWordNet automatic methods have full potential to predict the correct hypernyms for the new words within the top-3 positions of the ranked list of candidates. For the Russian dataset, the correct predictions could be found among the top-2 candidates on average. For the majority of novel words, the correct hypernyms appear within the top-10 candidates. Even though the automatic taxonomy enrichment systems do not always generate trustworthy results, they still can significantly facilitate the work of lexicographers or knowledge engineers through interface prompts.

Figure 7 presents a mockup interface that provides prompts during the annotating process. The expert is provided with the top-k candidates automatically generated by the model and can choose the correct ones from the list or propose his/her own hypernyms. For example, for the new term “touchscreen” the system correctly identifies the field and proposes meaningful candidates like “device.n.01”, “screen.n.03”, or “display.n.03”. However, if we assume that the

"screen.n.03" synset is too general as a hypernym, we can manually find the "computer_screen.n.01" synset. Thus, such candidates from the system could be quite helpful for an expert: manual searching for the candidates will be mainly replaced with checking boxes in front of the correct candidates generated by the system.

The described automatic methods and the presented interface allow much faster adaptation of existing taxonomic resources to new domains or new texts.

11. 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 [9] 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 [6] 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 suggest 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 [81]. Moreover, we find it promising to experiment with temporal embeddings such of those of [82] 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.

Acknowledgments

The participation of M. 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).

References

[1] T. Berners-Lee, J. Hendler and O. Lassila, The semantic web, *Scientific american* **284**(5) (2001), 34–43.

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

[3] 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. doi:10.18653/v1/S16-1169. https://www.aclweb.org/anthology/S16-1169.

[4] 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. doi:10.18653/v1/S16-1210. https://www.aclweb.org/anthology/S16-1210.

[5] 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. doi:10.18653/v1/S16-1208. https://www.aclweb.org/anthology/S16-1208.

[6] 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.

[7] 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, University of South Africa (UNISA), 2021, pp. 126–136. https://www.aclweb.org/anthology/2021.gwc-1.15.

[8] M. Tikhomirov and N. Loukachevitch, Meta-Embeddings in Taxonomy Enrichment Task, in: *Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue"*, 2021.

[9] 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.

[10] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, A Review of Relational Machine Learning for Knowledge Graphs, *Proceedings of the IEEE* **104** (2015). doi:10.1109/JPROC.2015.2483592.

[11] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic web* **8**(3) (2017), 489–508.

[12] 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–.

[13] 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–.

[14] 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 & Data Mining*, KDD '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 2663–2671–. ISBN 9781450355520. doi:10.1145/3219819.3220000.

[15] 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.

[16] 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.

[17] 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.

[18] 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. doi:10.18653/v1/D19-1522. https://www.aclweb.org/anthology/D19-1522.

[19] 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.

[20] 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.

[21] 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$_2$1.

[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. doi:https://doi.org/10.1016/j.future.2020.10.026. https://www.sciencedirect.com/science/article/pii/S0167739X2033003X.

[23] 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$_2$4.

[24] 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. doi:10.18653/v1/S18-1115. https://www.aclweb.org/anthology/S18-1115.

[25] 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. doi:10.18653/v1/S15-2151. https://www.aclweb.org/anthology/S15-2151.

[26] 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. doi:10.18653/v1/S16-1168. https://www.aclweb.org/anthology/S16-1168.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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. doi:10.18653/v1/S18-1116. https://www.aclweb.org/anthology/S18-1116.

[34] 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. doi:10.18653/v1/S18-1152. https://www.aclweb.org/anthology/S18-1152.

[35] 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. doi:10.18653/v1/S18-1151. https://www.aclweb.org/anthology/S18-1151.

[36] 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. doi:10.3115/v1/D14-1162. https://www.aclweb.org/anthology/D14-1162.

[37] 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. doi:10.18653/v1/P16-1226. https://www.aclweb.org/anthology/P16-1226.

[38] 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. doi:10.18653/v1/S16-1202. https://www.aclweb.org/anthology/S16-1202.

[39] 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. doi:10.18653/v1/P19-1474. https://www.aclweb.org/anthology/P19-1474.

[40] 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. doi:10.18653/v1/2020.acl-main.747. https://www.aclweb.org/anthology/2020.acl-main.747.

[41] 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.

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

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

[44] 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.

[45] 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.

[46] J.O. Neill and D. Bollegala, Meta-embedding as auxiliary task regularization, *arXiv preprint arXiv:1809.05886* (2018).

[47] 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.

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

[49] 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.

[50] 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.

[51] 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. doi:10.18653/v1/D18-1360. https://www.aclweb.org/anthology/D18-1360.

[52] 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, Volume 1 (Long Papers)*, Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 2227–2237. doi:10.18653/v1/N18-1202. https://www.aclweb.org/anthology/N18-1202.

[53] 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.

[54] T.N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907* (2016).

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

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

[57] D. Pujary, C. Thorne and W. Aziz, Disease Normalization with Graph Embeddings, in: *Proceedings of SAI Intelligent Systems Conference*, Springer, 2020, pp. 209–217.

[58] 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.

[59] T.N. Kipf and M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, in: *International Conference on Learning Representations (ICLR)*, 2017.

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

[61] G.A. Miller, WordNet: a lexical database for English, *Communications of the ACM* **38**(11) (1995), 39–41.

[62] 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 (online), 2020, pp. 1–23. https://www.aclweb.org/anthology/2020.semeval-1.1.

[63] 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.

[64] 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. doi:10.18653/v1/2020.coling-main.268. https://www.aclweb.org/anthology/2020.coling-main.268.

[65] 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.

[66] 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. doi:10.18653/v1/D15-1166. https://www.aclweb.org/anthology/D15-1166.

[67] 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*, Vol. 30, I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds, Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] M. Straka and J. Straková, Tokenizing, POS 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.

[73] 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.

[74] 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, *arXiv preprint arXiv:1805.09786* (2019).

[75] 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, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, eds, 2013. http://arxiv.org/abs/1301.3781.

[76] 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.

[77] 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*, Vol. 30, I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds, Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[78] 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.

[79] 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.

[80] 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.

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

[82] R. Goel, S.M. Kazemi, M. Brubaker and P. Poupart, Diachronic Embedding for Temporal Knowledge Graph Completion, *Proceedings of the AAAI Conference on Artificial Intelligence* **34**(04) (2020), 3988–3995. doi:10.1609/aaai.v34i04.5815. https://ojs.aaai.org/index.php/AAAI/article/view/5815.

[83] H. Hanke and D. Knees, A phase-field damage model based on evolving microstructure, *Asymptotic Analysis* **101** (2017), 149–180.

[84] E. Lefever, A hybrid approach to domain-independent taxonomy learning, *Applied Ontology* **11**(3) (2016), 255–278.

[85] P.S. Meltzer, A. Kallioniemi and J.M. Trent, Chromosome alterations in human solid tumors, in: *The Genetic Basis of Human Cancer*, B. Vogelstein and K.W. Kinzler, eds, McGraw-Hill, New York, 2002, pp. 93–113.

[86] P.R. Murray, K.S. Rosenthal, G.S. Kobayashi and M.A. Pfaller, *Medical Microbiology*, 4th edn, Mosby, St. Louis, 2002.

[87] E. Wilson, Active vibration analysis of thin-walled beams, PhD thesis, University of Virginia, 1991.

[88] A.V. Aho and J.D. Ullman, *The Theory of Parsing, Translation and Compiling*, Vol. 1, Prentice-Hall, Englewood Cliffs, NJ, 1972.

[89] G. Andrew and J. Gao, Scalable training of L1-regularized log-linear models, in: *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 33–40.

[90] M.S. Rasooli and J.R. Tetreault, Yara Parser: A Fast and Accurate Dependency Parser, *Computing Research Repository* **arXiv:1503.06733** (2015), version 2. http://arxiv.org/abs/1503.06733.

[91] R.K. Ando and T. Zhang, A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data, *Journal of Machine Learning Research* **6** (2005), 1817–1853.

[92] 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.

[93] 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. doi:10.18653/v1/N19-1423. https://www.aclweb.org/anthology/N19-1423.

[94] M. Schlichtkrull and H. Martínez Alonso, MSejrKu at SemEval-2016 Task 14: Taxonomy Enrichment by Evidence Ranking, in: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, Association for Computational Linguistics, San Diego, California, 2016, pp. 1337–1341. doi:10.18653/v1/S16-1209. https://www.aclweb.org/anthology/S16-1209.

[95] Z.S. Harris, Distributional structure, *Word* **10**(2–3) (1954), 146–162.

[96] G.A. Miller, *WordNet: An electronic lexical database*, MIT press, 1998.

[97] X. Cai, Y. Luo, Y. Zhang and X. Yuan, Improving Word Embeddings by Emphasizing Co-hyponyms, in: *International Conference on Web Information Systems and Applications*, Springer, 2018, pp. 215–227.

[98] G.A. Miller, WORDNET: A LEXICAL DATABASE FOR ENGLISH, in: *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992. https://www.aclweb.org/anthology/H92-1116.

[99] A. Moro and R. Navigli, SemEval-2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking, in: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Association for Computational Linguistics, Denver, Colorado, 2015, pp. 288–297. doi:10.18653/v1/S15-2049. https://www.aclweb.org/anthology/S15-2049.

[100] M. Negri and B. Magnini, Using wordnet predicates for multilingual named entity recognition, in: *Proceedings of The Second Global Wordnet Conference*, 2004, pp. 169–174.
[101] J. Herrera, A. Penas and F. Verdejo, Textual entailment recognition based on dependency analysis and wordnet, in: *Machine Learning Challenges Workshop*, Springer, 2005, pp. 231–239.
[102] S.P. Ponzetto and M. Strube, Exploiting Semantic Role Labeling, WordNet and Wikipedia for Coreference Resolution, in: *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, Association for Computational Linguistics, New York City, USA, 2006, pp. 192–199. https://www.aclweb.org/anthology/N06-1025.
[103] Y. Kuratov and M. Arkhipov, Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language, *arXiv preprint arXiv:1905.07213* (2019).
[104] B. Shi and T. Weninger, ProjE: Embedding projection for knowledge graph completion, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31, 2017.
[105] Y. Lin, Z. Liu, M. Sun, Y. Liu and X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29, 2015.
[106] M. Maziarz, M. Piasecki, E. Rudnicka and S. Szpakowicz, plWordNet as the Cornerstone of a Toolkit of Lexico-semantic Resources, in: *Proceedings of the Seventh Global Wordnet Conference*, University of Tartu Press, Tartu, Estonia, 2014, pp. 304–312. https://www.aclweb.org/anthology/W14-0142.
[107] M. Asgari, A. Hadian and B. Minaei-Bidgoli, Farsbase: The persian knowledge graph, *Semantic Web–Interoperability, Usability, Applicability* (2018).
[108] N. Jain, Domain-Specific Knowledge Graph Construction for Semantic Analysis, in: *European Semantic Web Conference*, Springer, 2020, pp. 250–260.
[109] P.A. Bonatti, S. Decker, A. Polleres and V. Presutti, Knowledge graphs: New directions for knowledge representation on the semantic web (dagstuhl seminar 18371), in: *Dagstuhl Reports*, Vol. 8, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
[110] J. McCrae, C. Fellbaum and P. Cimiano, Publishing and Linking WordNet using lemon and RDF, in: *Proceedings of the 3rd Workshop on Linked Data in Linguistics*, 2014.
[111] S.M. Harabagiu and D.I. Moldovan, Enriching the wordnet taxonomy with contextual knowledge acquired from text, *Natural language processing and knowledge representation: language for knowledge and knowledge for language* (2000), 301–333.
[112] R. Snow, D. Jurafsky and A.Y. Ng, Semantic taxonomy induction from heterogenous evidence, in: *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 2006, pp. 801–808.
[113] D. Jurgens and M.T. Pilehvar, Reserating the awesometastic: An automatic extension of the WordNet taxonomy for novel terms, in: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1459–1465.