Taxonomy Enrichment with Text and Graph Vector Representations[1]

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 ("is-a") relationship. Hypo-hypernymy is presented in almost every knowledge base and is used to describe the order of thing we live by. 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. Deep representations of graph structures like GCN autoencoder, Poincaré embeddings, node2vec emerged and have recently demonstrated very promising results on various NLP tasks. Our approach is a comprehensive study of the existing approaches to taxonomy enrichment based on word and graph vector representations. We also explore the ways of using deep learning architectures to extend taxonomic backbones of knowledge graphs. We achieve state-of-the-art results across different datasets.

Keywords: taxonomy enrichment, knowledge graph completion, graph vector representations, word embeddings, graph convolutional auto-encoder, sequence-to-sequence architecture

1. Introduction

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 ("is-a") relationship. Hypo-hypernymy is presented in almost every knowledge base and is used to describe the world structure. Consequently, many lexical databases such as WordNet became widespread across various NLP applications, e.g. lexical entailment [1], entity linking [2], named entity recognition [3], coreference resolution [4].

To keep up with the global development, taxonomies, as well as other knowledge graph structures, need to be constantly updated. With the rapid growth of lexical resources for specific domains, 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 annotation process. There already exist several initiatives on WordNet extension, for example, the Open English WordNet[2] with thousands of new manually added entries or plWordNet [5] which includes a mapping to an enlarged Princeton WordNet. However, the manual annotation process is too costly: it is time-consuming and requires language or domain experts.

[1] This article is based on previous publications "Studying Taxonomy Enrichment on Diachronic WordNet Versions" (COLING-2020) and "Exploring Graph-based Representations for Taxonomy Enrichment" (GWC-2021) featuring more detailed analysis, more graph-based experiments and SOTA baselines, and a new unpublished block of experiments featuring meta-embeddings.

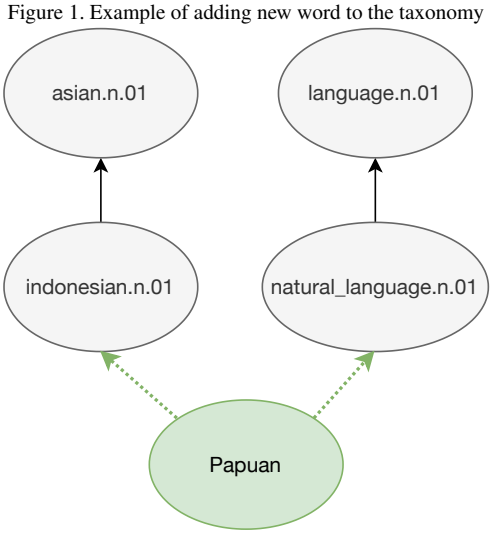
[*] Corresponding author. E-mail: irina.nikishina@skoltech.ru.

[2] https://en-word.net

We suggest that it would be beneficial to assist manual work by introducing automatic annotation systems to keep valuable lexical resources up-to-date. In this paper, we analyse the approaches to automatic enrichment of knowledge bases

Taxonomy enrichment is closely related to knowledge graph completion (KGC) task or link prediction, whose aim is to enrich existing graphs with new 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. In the case of taxonomies, 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 ("is-a") relation. Figure 1 demonstrates the subgraph retrieved from the WordNet taxonomy. The word "Papuan" is attached to the synsets "indonesian.n.01" and "natural_language.n.01" as it can mean both population and language.

Figure 1. Example of adding new word to the taxonomy



Specifically, taxonomy enrichment aims at identifying appropriate hypernyms from the taxonomy for new words not included in it (*orphan words*). The task of finding a single suitable synset is difficult for a machine, and 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. If we relax the requirement of uniqueness and ask instead to provide N (for example, 10 or 15) most suitable candidates, this list can contain correct synsets with higher probability. This setting is also consistent with manual annotation. Presenting an annotator with a small list of candidates will facilitate the process: annotators will not have to look through all synsets of the taxonomy. Thus, we formulate the task as the soft ranking problem, where the synsets are ranked according to their suitability for a given word.

SemEval-2016 task 14 [6] was the first effort to evaluate this task in a controlled environment, but it provided an unrealistic task setting. Namely, the participants were given definitions of words to be added to the taxonomy, which are often unavailable in real-world scenarios. The majority of the presented methods heavily depended on those definitions [7, 8]. In contrast to that, we present a resource-poor scenario and solutions which conform to it.

Most existing approaches for completing knowledge bases rely on graph structures or graph representations [9, 10], whereas the approaches for extending taxonomies mostly rely on word embeddings [6]. We assume that graph-based representations are complementary to the distributional word embeddings, as they capture the hypo-hypernym relations from graphs. We expect that models based on graph representations or their combinations with distributed word vector representations could be beneficial for the taxonomy enrichment task.

On the other hand, deep representations of graph structures like GCN autoencoder [11, 12] or sequence-to-sequence architecture for the taxonomy enrichment task [13] have recently demonstrated very promising results. Therefore, in this work, we also explore how deep learning approaches can be used to extend taxonomic backbones of knowledge graphs.

We check our hypothesis on several models which make use of graph structures (node2vec [14], Poincaré embeddings [15], GCN autoencoder [11], TaxoExpan [12]) on our datasets and compare them with the approaches which apply word vector representations and their combinations as meta-embeddings [16], features from Wiktionary [17], and sequence-to-sequence architecture [13].

More specifically, our contributions are as follows:

- We present a computational study of various approaches to taxonomy enrichment including recent state-of-the-art results, e.g. [12, 13].
- We present datasets for studying diachronic evolution of wordnets for English and Russian, extending the monolingual setup of the RUSSE'2020

shared task [18] with a larger Russian dataset and similar English versions.[3]
– We explore the use of meta-embeddings for the task of taxonomy enrichment.
– We present several efficient methods for taxonomy enrichment reaching new state-of-the-art results on the diachronic WordNets datasets for both English and Russian.
– We explore the benefits of graph-based representations for the taxonomy enrichment task and their combinations with the word distributed representations.

2. Related Work

There exist numerous approaches to knowledge base completion, which make use of various neural network architectures described in review papers [19, 20]. Most of them apply low-dimensional graph embeddings [21, 22], deep learning architectures like autoencoders [23] or graph convolutional networks [24–26]. Another group of approaches makes use of tensor decomposition approaches: Tucker decomposition [27] and Canonical Polyadic decomposition (CANDECOMP/PARAFAC) [28, 29]. 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 ontoligies and knowledge bases posess 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 [30]: 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 [31–33], 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.

2.1. 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 [7] 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 [8] 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 [18], 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 [34] 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 [35] or ranked all hypernyms by suitability for a particular word [36]. They also used a range of additional resources, such as Wiktionary, dictionaries, additional corpora [37]. Interestingly, only one of the well-performing models [38] used context-informed embeddings (BERT) or external tools such as online Ma-

[3] Code & datasets available at https://github.com/skoltech-nlp/diachronic-wordnets

chine 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).

2.2. Approaches Based on Word Vector Representations

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 [30] most of participants use word embeddings. For intance, Bernier-Colborne and Barriere [39] predict the likelihood of the relationship between an input word and a candidate using word2vec embeddings. Berend et al. [40] use Word2vec vectors as features of a logistic regression classifier. Maldonado and Klubicka [41] simply consider top-10 closest associates from the Skip-gram word2vec model as hypernym candidates. Pre-trained GloVe embeddings [42] are also used to initialize embeddings for an LSTM-based Hypernymy Detection model [43].

Participants also solve the SemEval-2016 Task 13 on taxonomy induction with word embeddings [44]: 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 [45] 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 [6] and RUSSE'2020 [18] 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 [7]. Another option is to train word2vec embeddings from scratch and cast the task as a classification problem [35]. Some participants compare the approach based on XLM-R model [46] with the word2vec "hypernyms of co-hyponyms" method [37]. 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 [18, 30].

2.3. 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 [47], and in medical domain [48].

Coates et al. [49] have shown 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 [50]. Autoencoders [16] became a further development of 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 backward.

In 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 meta-embedding dimension.

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 [51] the authors investigated the performance of the autoencoders depending on the loss function. They found that there is no evident winner across tasks and that different loss functions should be chosen for different tasks.

Meta-embeddings were used earlier in such tasks as dependency parsing [47], classification in healthcare [48], named-entity recognition [51, 52], sentiment analysis [51], word similarity and analogy tasks [16, 49, 50]. To the best of our knowledge, meta-embeddings have not been applied to the taxonomy enrichment task.

2.4. Graph-based Representations for Taxonomies

Taxonomies can be represented as graphs and there exist various approaches to learn graph-based repre-

sentations. Most of them have been tested on tasks related to the taxonomy enrichment.

For instance, node2vec embeddings [14] are used for taxonomy induction among other network embeddings [53]. In [45], the authors perform the same task. They use hyperbolic Poincaré embeddings to enhance automatically created taxonomies. The SemEval-2016 subtask of reattaching orphan 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, [45] use Hearst Patterns to discover hyponym-hypernym relationships. This technique operates on words, and cannot be transferred to word-synset relations without extra manipulation.

Graph convolutional networks (GCNs) [11] as well as graph autoencoders [54] are mostly applied to the link prediction task on large knowledge bases. For example, in [55] 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 [56]. As for the taxonomy enrichment task, we are only aware of a recent approach TaxoExpan [12] which applies position-enhanced graph neural networks (GCN [57] and GAT [58]) that we also evaluate on our datasets[4].

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.

3. 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 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 [59] and RuWordNet taxonomies. It is important to mention that by using the word "diacronic" we do not imply lexical diachrony, e.g., semantic shifts [60], 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.

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

3.1. English Dataset

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

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

As gold standard hypernyms, we use not only the immediate hypernyms of each lemma 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 obser-

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

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

vations, 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 created for this research and the code for their construction are publicly available [5].

3.2. Russian Datasets

In order to create an analogous version to English dataset for Russian, we use the RuWordNet taxonomy [34]. As a taxonomy, RuWordNet possesses *synsets* — sets of synonyms expressing a particular concepts. 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 [18] 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; (word of the feminine gender, denoting profession, job, specialization, classes, etc.),
- 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.

[5] https://zenodo.org/record/4279821

3.3. 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 [30]. 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 3, the gold-standard hypernym list is extended with second-order hypernyms (parents of parents). This extension can distort MAP. If we consider all gold standard answers as compulsory for the maximum score, it means that we demand models to find both direct and second-order hypernyms. This disagrees with the original motivation of including second-order hypernyms to the gold standard — it was intended to make the task easier by allowing a model to guess a direct *or* a second-order hypernym.

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

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

While the case 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 [18], 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.

4. Taxonomy Enrichment Methods Based on Word Representations

We derive our methods from the baseline distributional model from RUSSE-2020 shared task. We extend it with ranking of synset candidates using the information from Wiktionary and various types of embeddings.

4.1. Hypernym of Co-Hyponyms (HCH) Baseline

According to [61] and [45], co-hyponyms (words or phrases that share a hypernym) usually have similar contexts. On the other hand, the distributional hypothesis states that words that occur in similar context tend to have similar meanings [62]. Our approach combines the two hypothesis. We take top $k = 10$ nearest neighbours of the input word from the pre-trained embedding model (according to the above considerations they should be co-hyponyms). Subsequently, hypernyms of those co-hyponyms are extracted from the taxonomy. These hypernyms are also considered hypernyms of the input word.

There is no one-to-one mapping between a word and a synset. On one hand, several hypernyms can belong to one synset, on the other hand, one word can occur in multiple synsets. Thus, all synsets associated with the list of extracted hypernyms are extracted. Then, vector representation of a synset is computed by averaging embeddings of all lemmas belonging to the synset. We pre-computed embeddings for all synsets and are searching for the closest synsets when generating top k nearest neighbours instead of words. Despite its simplicity, this method turned out to be a strong baseline.

4.2. Similarity-Based Hypernym Ranking

This baseline has a shortcoming: it lacks sorting operation on the extracted candidates. The rank of synsets is defined only by the rank of a corresponding nearest neighbour.

We improve the HCH approach by ranking the generated synset candidates. In addition to that, we extend the list of candidates with second-order hypernyms (hypernyms of each hypernym). The direct hypernyms of the word's nearest neighbours can be too specific, whereas second-order hypernyms are likely to be more abstract concepts, which the input word and its neighbours have in common. After forming a list

of candidates, we score each of them according to the following equation:

$$score_{h_i} = n \cdot sim(v_o, v_{h_i}), \quad (2)$$

where v_{h_i} is vector representation of a hypernym h_i, n is the number of occurrences of this hypernym in the merged list, $sim(v_o, v_{h_i})$ is the cosine similarity of the vector of the orphan word o and hypernym vector h_i. By computing the score, we assume that the most frequent and the most similar candidates are the true hypernyms of the word. We sort the hypernyms by this score and return top k.

4.3. Wiktionary-based Features

One of the promising multilingual resources which the taxonomy enrichment models could benefit from is Wiktionary[6]. We choose it as Wiktionary is the only large web-based free content dictionary existing for 175 languages, including English (6,334,384 entries) and Russian (1,076,156 entries). More importantly, each Wiktionary page usually comprises a definition and lists of hypernyms, hyponyms and synonyms, which could be useful for our task. We implement the following Wiktionary features:

- the candidate is present in the Wiktionary hypernyms list for the input word (binary feature),
- the candidate is present in the Wiktionary synonyms list (binary feature),
- the candidate is present in the Wiktionary definition (binary feature),
- average cosine similarity between the candidate and the Wiktionary hypernyms of the input word.

We do not use the definitions directly, as their texts are too noisy. They often include example usages of words which cannot be separated from the definitions and can distort their vector representations.

We extract lists of hypernym synset candidates using the baseline procedure and compute the four Wiktionary features for them. In addition to that, we use the score from the previous approach as a feature. To define the feature weights, we train a Logistic Regression model with L_2 regularisation on a training dataset which we construct from the older (known) versions of WordNet. This dataset is constructed analogously to the datasets for evaluation which we described in Section 3 using all leaf synsets from the older WordNet. For each lemma of such synsets, we extract their gold standard hypernym synsets. As a result, our dataset comprised 79,000 positive and 79,000 negative examples of *word-candidate* pairs for both nouns and verbs for English dataset and 59,914 and 59,914 for Russian.

In order to understand the contribution of the Wiktionary features to the final score we compute the number of orphans encountered in Wiktionary (97% to 100%) and the number of orphans containing at least one hypernym in Wiktionary fields (2–18% for "hypernyms" field, 1–2% for "synonyms" field and 26–35% in the definition). From those number we expect that the impact of the Wiktionary features would not be very significant.

4.4. Meta-Embeddings for Hypernym Prediction

In our work we compare simple meta-embeddings such as the concatenation of source embeddings and SVD over the concatenation and two varians of autoencoders for the generation of meta-embeddings: Concatenated Autoencoded Meta-Embeddings (CAEME) and Averaged Autoencoded Meta-Embeddings (AAEME), which have shown good results in previous works [16].

Let us consider two source embeddings $s_1(w)$ and $s_2(w)$, their encoders $E_1(w)$ and $E_2(w)$ and their decoders $D_1(w)$ and $D_2(w)$. Meta-embedding $m(w)$ in CAEME is constructed as the L_2-normalised concatenation of two encoded source embeddings $E_1(s_1(w))$ and $E_2(s_2(w))$:

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

where $\oplus$ is the concatenation operation.

In CAEME, the dimensionality of the meta-embedding space is the sum of the dimensions of the source embeddings. 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 (3) instead of their concatenation. Averaging gives the possibility to avoid increasing the dimensionality of the meta-embedding.

The AAEME encoder computes the 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 two encoded

[6] https://wiktionary.org

versions of the source embeddings $E_1(s_1(w))$ and $E_2(s_2(w))$.

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

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

$$\begin{aligned} \hat{s}_1(w) &= D_1(m(w)) \\ \hat{s}_2(w) &= D_2(m(w)) \end{aligned} \quad (5)$$

In training autoencoders any distance or similarity measure as MSE, KL-divergence, or cosine distance can be used.

To obtain meta-embedding representations after training, only the encoders are applied, which convert the input source embeddings into a meta representation. Further, these meta-embedding vectors are used as vector representations of words.

More specifically, in our approach the embeddings are used to generate a list of most similar taxonomy entries to the target word according to cosine similarity. The term "taxonomy entry" comprises words or phrases described in the taxonomy. For each target word, the top 20 taxonomy entries are considered. The number of elements for consideration was chosen experimentally. For each entry in the similarity list, all synsets, their direct and second-order hypernyms are extracted from the taxonomy. They are considered as candidate synsets to be hypernyms of the target word. For candidate hypernyms synsets, several features are calculated. Logistic regression is used to predict the probability of a candidate to be a hypernym of the target word. The calculated features are as follows:

- maximal, minimal and average similarity between the query word and synonyms in a candidate synset;
- similarity values between the query word and synonyms in hyponym synsets of the candidate synset. At first, maximal, minimal and average similarity values are calculated for the query and synonyms in each hyponym synset. Then maximal, minimal and average similarity values are calculated over all hyponym synsets for a given candidate synset;
- positional feature (0, 1, 2): if candidate is one of the synset of a taxonomy entry or it is its direct or second-order hypernym.
- the number of occurrences of the synset in the candidate list.

In total, 17 features were calculated. Training data is generated randomly and automatically from RuWordNet and WordNet-1.6 and summarized in Table 3.

Table 3
Training datasets for meta-embeddings.

Language	**Part of speech**	
	Nouns	Verbs
English	2931	1532
Russian	2990	814

To study combined techniques of meta-embeddings with the Wiktionary word sense description (definitions, synonyms, hypernyms, hyponyms) corresponding features were added into logistic regression, which gives 23 features for each candidate.

4.5. Leveraging Web-based Features

In this section, we describe a method that reached top 1 rank at the RUSSE'2020 shared task on taxonomy enrichment for the Russian language for nouns. This approach leverages the power of the existing general-purpose services, we present an approach which makes use of the two famous search engines: Google (for both English and Russian datasets) and Yandex[7] (for Russian only). According to our hypothesis, the search results are likely to contain 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, we google it and see the 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 orphan words to a rich-resource language (e.g. English) in order to use an existing taxonomy (e.g. English Princeton WordNet). In this study

[7]https://yandex.com

we use Yandex Machine Translation system[8] to translate query words into English and then translate found hypernyms back into Russian.

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

Similarly to the approaches described above, here we also make use of Wiktionary and fastText embeddings cosine similarity, however, we treat word synsets and lemmas in a different way. In the previously described approaches we computed embeddings for multiword lemmas by averaging word embeddings of individual words in these lemmas. Here we treat them as sentences — we compute their embeddings using `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 lemma vectors into a synset vector but operate with lemma embeddings directly.

The algorithm consists of two steps: candidate generation and ranking. Our candidate list is formed of the following synsets:

- synsets which contain the top-10 nearest neighbours of the query word;
- hypernyms and second-order hypernyms of those synsets;
- synsets and hypernyms of the lemmas that match the hypernyms of the query word found in Wiktionary;
- cross-lingual synsets and hypernyms of the lemmas that match the hypernyms of the query word (for the Russian language only).

All candidates are then ranked by a logistic regression model which uses the following features:

- candidate is present in the synsets of one of the most similar lemmas (nearest neighbours);
- candidate is a hypernym of one of the most similar lemmas;
- candidate is a second-order hypernym of one of the most similar lemmas;
- candidate is present in the synsets of the lemmas that match hypernyms of the query word found in Wiktionary;
- candidate is a hypernym of one of the lemmas that match hypernyms of the query word found in Wiktionary;
- lemmatised candidate is present in the query word definition in Wiktionary;
- candidate is present in WordNet synset candidates (for the Russian language only);
- candidate is present in hypernyms of the WordNet candidates (for the Russian language only);
- lemmatised candidate is present on the Google results page;
- lemmatised candidate is present on the Yandex results page (for the Russian language only).

The training set for the logistic regression model is formed as follows. For each query word in the list of orphan 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 was presented at the RUSSE'2020 Taxonomy Enrichment task for the Russian Language and achieved the best result on the nouns track.

4.6. Sequence-to-Sequence Architecture for Hypernym Prediction

A completely different approach to make use of fastText embeddings is the one from "Leveraging WordNet Paths for Neural Hypernym Prediction" [13]. 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 [63] with Luong attention [64]. First, they average fastText embeddings for the input synset or phrase and put it through the encoder. Then the decoder sequentially generates synsets from the encoder hidden state, conditioned on the previously generated ones. The authors consider two different dataset varia-

[8] https://translate.yandex.ru/

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

tions to train their model on: *hypo2path* (which starts from root and goes down the taxonomy to the closest hypernym) and *hypo2path rev* (where target sequence of synsets starts with the closest hypernym up to the root entity).

To be able to apply their sequence-to-sequence architecture on our data, we build new datasets similar to the ones described in [13]: 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 right to left or the *hypo2path rev* from left to right and get the first 10 synsets.

5. Taxonomy Enrichment Methods Based on Graph Representations

This section describes methods using graph-based representation alone or along with word vectors for the taxonomy enrichment task.

5.1. Candidate Generation Using 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" [15]. 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 [45].

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 [15].

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 4.1) and then aggregate embeddings in hyperbolic space using Einstein midpoint, following [65]. The resulting vector is considered as an embedding of the input word in the Poincaré space.

Then, we search for the word's top-10 Poincaré nearest neighbours and consider them as candidates. We also try to extend the candidate list with the hypernyms of each Poincaré associate and rank them according to their frequency and similarity to the input word.

5.2. Candidate Generation Using 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 [14] approach which represents a "random walk of fixed length l" and "two parameters p and q which guide the walk in breadth of in depth". Node2vec randomly samples sequences of nodes and then applies a Skip-gram model 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 technique 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.

We also use an alternative approach to compute out-of-vocabulary node2vec embeddings. Namely, we apply linear transformation from the source fastText to the target node2vec embeddings. For this purpose we train a matrix which is used to project fastText embeddings of the input words to the target node2vec space.

5.3. Link Prediction Using GCN Autoencoder

The models described above have a major shortcoming: the resulting vectors for the input words heavily depend on their representations in 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 Poincaré model and result in inaccurate output even if the Poincaré model is of high quality.

Therefore, we test graph convolutional network architecture [11] which makes use of both fastText embeddings and the graph structure of the taxonomy. In particular, we use graph autoencoder model [54] where encoder is a graph convolutional network architecture. This model learns vector representations in a completely unsupervised way: it encodes the nodes in the network in a low-dimensional space in such a way that the embeddings can be decoded into a reconstruction of the original network. FastText embeddings are used as input node features. Even though new words are not connected to the taxonomy, it is still possible to compute their embeddings according to their input node features.

For each new node we get its vector representation from the encoder and then predict the probability of the link between the new node and all other nodes in the graph. The top-10 synsets from the existing taxonomy with highest probabilities are considered as final candidates.

5.4. Combining Word and Graph Representations

Additionally, we extend the approach using Logistic Regression model with features based on node2vec and Poincaré embeddings. Namely, we use two extra features: cosine similarity between the candidate and the input word in node2vec vector space and similarity between the candidate and the input word in Poincaré ball model. The overall formula is the following:

$$score_{h_j} = w \cdot m = \sum_{i=1}^{n} w_i m_i \quad (6)$$

Feature weights from the Logistic Regression model are denoted as vector w, m is the feature vector.

6. Experiments

In this section, we report the performance of our models on the Taxonomy Enrichment task and discuss reasons of low performance of methods exploiting hierarchical structure of the taxonomy.

Table 4
Experiments with pre-trained embeddings.

Method	English		Russian	
	Nouns	Verbs	Nouns	Verbs
	fastText			
HCH	0.245	0.197	0.421	0.334
Ranking	0.250	0.200	0.507	0.336
Ranking + Wiki	**0.344**	**0.237**	**0.552**	0.391
	word2vec			
HCH	0.130	0.208	0.357	0.379
Ranking	0.147	0.195	0.410	0.391
Ranking + Wiki	0.164	0.212	0.443	**0.451**
	BERT			
HCH	0.238	0.097	0.138	0.121
Ranking	0.237	0.107	0.185	0.119
Ranking + Wiki	0.253	0.120	0.218	0.162

6.1. Approaches Based on Word Vector Representations

We test our baseline approaches applying different types of embeddings: non-contextualised fastText [66] and word2vec [67] embeddings and contextualised BERT [68] embeddings. We use the fastText embeddings from the official website[10] for both English and Russian, trained on Common Crawl from 2019 and Wikipedia CC including lexicon from the previous periods as well. For word2vec we use models from [69, 70] for both English[11] and Russian [12]. We lemmatise words and synsets for both languages with the same UDPipe [71] model which was used while training the representations. For the out-of-vocabulary words we find all words in the vocabulary with the longest prefix matching this word and average their embeddings like in [36].

While fastText and word2vec embeddings can be generated for individual words, BERT requires a context for a word (i.e. a sentence containing it) to generate its embedding. For experiments with English datasets, we extract contexts from Wikipedia. For the experiments with Russian, we use a news corpus provided by the organisers of RUSSE'2020,[13] which con-

[10] https://fasttext.cc/docs/en/crawl-vectors.html
[11] http://vectors.nlpl.eu/repository/20/29.zip
[12] http://vectors.nlpl.eu/repository/20/185.zip
[13] https://github.com/dialogue-evaluation/taxonomy-enrichment

tains at least 50 occurrences for each word in the dataset.

We use the pre-trained BERT-base model for English from [68]. For Russian, we utilize RuBERT model from [72], which proved to outperform the Multilingual BERT from the original paper. To compute BERT embeddings for orphans and synsets, we extract sentences containing them from the corresponding corpora. If the words are absent in the corpora, we computed the average of lemmas without context for synsets and the embedding of the input word without context. We also averaged word-pieces for the out-of-vocabulary words. We lemmatise corpora with UDPipe [71] to be able to find not only exact word matches but also their grammatical forms. We rely on UDPipe as it supports many languages and shows reasonable performance on our data. In case of multiple occurrences of the same orphan, we average the retrieved contextualised embeddings.

Table 4 demonstrates the results for the HCH baseline and its enhanced versions on each type of embeddings and both languages. We see that our methods consistently improve the hypernym detection baseline for both nouns and verbs across different datasets. Extending a list of hypernym candidates with second-order hypernyms and ranking them increases MAP by a large margin, especially for nouns. Adding Wiktionary features further boosts the performance of models. However, the use of contextualised word embeddings demonstrated in Table 4 does not guarantee high results in this task. The models which used BERT vector representations perform worse than the same approaches using fastText. This also holds for all datasets and parts of speech. Apparently, non-contextualised embeddings are good at modelling common and the most popular word senses, whereas BERT embeddings aggregate sense from different contexts, which results in mixing different senses and confusing word representations. For example, for the word "смайлик" (emoji) the predicted candidates are completely unsuitable (person, device, flatterer, hypocrite, visual materials) in comparison with the fastText prediction and correct hypernyms (graphical sign, image, symbol). Therefore, the contexts (in a broad sense) extracted from the pre-trained non-contextualised embeddings are sufficient to attach new words to the taxonomy.

When comparing results for word2vec and fastText embeddings, we can see that fastText performs better across all datasets except for Russian verbs. This phenomena can be explained by the rich morphology of the Russian language: new (orphan) verbs are given as infinitives, whereas verb infinitives are much less widespread in corpora that other verb forms. If we normalise corpus for training, then the word embeddings for verbs will be of much better quality.

Interestingly, 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.

6.2. Results on Meta-Embeddings

To evaluate our meta-embedding approach, we apply different source vector representations: fastText[14], word2vec[15], GloVe[16]. We investigate different meta-embedding approaches: concatenation, SVD over concatenation, CAEME, AAEME. As standard loss function for AEME approaches we use cosine distance loss, but also experimented with MSE loss, KL divergence loss and cosine distance loss.

Another scope of experiments relates to the additional restrictions to the generated meta-embeddings in the AEME algorithms such as triplet loss.We restrict the word to be closer to its semantically related words according to the taxonomy than to a randomly chosen word. 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 from synonyms, hyponyms and hypernyms;
2. on each epoch, we randomly select K positive words from this related set and K negative words from the vocabulary;
3. if the word is not presented in the taxonomy, then the noisy variations of the original vector are positive vectors;
4. next, triplet margin loss is calculated: the triplet loss is combined with the original loss as $\alpha * loss + (1 - \alpha) * triplet\_loss$, we used $\alpha = 0.5$.

The results can be seen in Tables 5 and 6. The low results for GloVe and word2vec are explained by the data coverage: some query words can be ab-

[14]Common Crawl English and Russian versions from https://fasttext.cc/docs/en/crawl-vectors.html

[15]Araneum for Russian and Gigaword for English from http://vectors.nlpl.eu/repository/

[16]Common Crawl 840b tokens from https://nlp.stanford.edu/projects/glove/

Table 5
MAP scores for the taxonomy enrichment methods for the English datasets. Results based on meta-embeddings

method	nouns			verbs			vector
	1.6-3.0	1.7-3.0	2.0-3.0	1.6-3.0	1.7-3.0	2.0-3.0	dim
fastText	0.300	0.346	0.396	**0.290**	**0.224**	0.280	300
word2vec	0.065	0.065	0.077	0.077	0.102	0.151	300
GloVe	0.099	0.109	0.132	0.165	0.128	0.187	300
concat	0.296	0.337	0.390	0.264	0.191	0.234	900
SVD	0.300	0.349	0.404	0.283	0.218	0.286	600
CAEME	0.288	0.330	0.379	0.264	0.196	0.252	900
CAEME triplet loss	0.309	0.362	0.409	0.281	0.219	0.281	900
AAEME	0.314	0.364	0.417	0.274	0.203	0.265	600
AAEME triplet loss	0.317	0.368	0.421	0.278	0.216	0.265	600
SVD + wiki	0.313	0.366	0.427	0.284	0.217	**0.289**	600
AAEME + wiki	**0.327**	0.383	**0.445**	0.280	0.202	0.265	600
AAEME triplet loss + wiki	**0.328**	**0.386**	**0.444**	0.279	0.218	0.268	600

Table 6
MAP scores for the taxonomy enrichment methods for the Russian datasets. Results based on meta-embeddings

method	nouns		verbs		vector
	non-restricted	restricted	non-restricted	restricted	dim
fastText	0.416	0.537	0.318	0.418	300
word2vec	0.276	0.526	0.231	0.272	600
concat	0.401	0.563	0.337	0.423	900
SVD	0.435	0.579	0.382	0.442	600
CAEME	0.468	0.579	0.352	0.433	900
CAEME triplet loss	0.470	0.580	0.350	0.420	900
AAEME	0.466	0.577	0.350	0.432	600
AAEME triplet loss	0.474	0.581	0.352	0.437	600
SVD + wiki	0.452	**0.612**	**0.423**	0.458	600
AAEME + wiki	0.484	**0.611**	0.403	**0.486**	600
AAEME triplet loss + wiki	**0.490**	**0.611**	0.400	0.477	600

sent in word2vec or GloVe fixed vocabulary, whereas fastText allows generating vectors even for the out-of-vocabulary words. It can be seen that in Russian datasets the performance of word2vec is much higher than for English datasets. This phenomenon can be explained by the specificity of the datasets: the Russian datasets include only single words as queries, whereas English datasets contain a large number of multi-word expressions, which possess no embeddings in the pre-trained GloVe and word2vec models.

Concatenation as a meta-embedding method did not improved the results for any English dataset, as word2vec and GloVe embeddings are not able to share useful information. For the Russian dataset with better embeddings coverage, the concatenation of fastText and word2vec embeddings improves the prediction for most datasets except for the non-restricted nouns.

Singular value decomposition (SVD) applied to the concatenation of initial embeddings improves the results of concatenation for both languages. However,

Table 7
MAP scores for the graph-based methods for the English datasets.

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
Poincaré embeddings	0.059	0.066	0.101	0.126	0.066	0.109
node2vec	0.194	0.219	0.155	0.151	0.109	0.147
node2vec (projection)	0.040	0.027	0.022	0.104	0.052	0.038
GCN autoencoder	0.157	0.175	0.168	0.109	0.094	0.117

Table 8
MAP scores for graph-based methods for the Russian datasets.

method	nouns		verbs	
	non-restricted	restricted	non-restricted	restricted
Poincaré embeddings	0.143	0.252	0.105	0.140
node2vec	0.266	0.366	0.168	0.252
node2vec (projection)	0.185	0.253	0.180	0.253
GCN autoencoder	0.183	0.261	0.095	0.141

for English SVD is still not able to handle with the pre-trained embeddings of low quality. As for the Russian verb datasets, SVD significantly improves the results achieving the highest score among all meta-embeddings without external sources on the Russian verb datasets.

The AAEME autoencoder with the cosine loss is comparable to the CAEME autoencoder for the Russian datasets. For English it manages to slightly improve the results for nouns. Among meta-embeddings based on autoencoders, the highest results are obtained by the AAEME autoencoder with the triplet loss. This approach improves the results of hypernym prediction for almost all datasets compared to the initial fastText embeddings.

Moreover, we see that experiments with the additional Wiktionary features also improves the results for both languages (Tables 5 and 6).

6.3. Results for Graph-based Approaches

We test the models suggested in Section 5 on both English (Table 7) and Russian (Table 8) datasets. It is clearly seen that distributed word vector representations outperform graph-based approaches 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 nearest neighbours retrieved from fastText 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.

Likewise, node2vec model also possesses a non-extensible set of embeddings for the taxonomic synsets and uses averaging of fastText associates for representing the new words which negatively affects the results. However, the approach which uses node2vec embeddings and averages top-5 fastText associates is the best-performing approach across methods with graph representations. Moreover, node2vec embeddings perform much better than the Poincaré embeddings. Einstein midpoint aggregation used in our Poincaré-based model makes generalisation of the associate synsets, which results in too abstract synset candidates. On the other hand, averaging node2vec vectors does not have such an effect. The differences between the two models are illustrated by the examples in Tables 11 and 12.

However, node2vec embeddings still rely on the fastText similarities of the closest embeddings to the input word vector and propagate the fastText inaccuracies. Linear projection which is an alternative option for the computation of node2vec vectors for out-of-vocabulary words, does not solve the problem either.

Table 9
MAP scores for the taxonomy enrichment methods for the English datasets.

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
node2vec (top-5 fastText associates)	0.194	0.219	0.155	0.151	0.109	0.147
Ranking + wiki [17]	**0.337**	0.380	0.344	0.270	0.200	0.237
Ranking + wiki [17] + node2vec	0.313	0.380	0.340	0.259	0.195	0.200
Ranking + wiki [17] + node2vec + Poincaré	0.311	0.350	0.300	0.251	0.177	0.248
Ranking + wiki (word2vec)	0.142	0.180	0.164	0.229	0.156	0.212
hypo2path rev [13]	0.260	0.284	0.237	0.171	0.106	0.119
hypo2path [13]	0.252	0.263	0.207	0.161	0.093	0.068
TaxoExpan [12]	*0.003*	*0.007*	*0.006*	*0.001*	*0.000*	*0.000*
Top-1 RUSSE'2020 for nouns	0.332	**0.393**	0.436	0.252	0.208	0.250
Top-1 RUSSE'2020 for nouns, no search engine features	0.251	0.309	0.344	0.230	0.180	0.221
Meta-embeddings SVD	0.300	0.350	0.404	**0.283**	**0.218**	0.286
Meta-embeddings AAEME triplet loss	0.317	0.368	0.421	0.278	**0.216**	0.265
Meta-embeddings SVD + wiki	0.313	0.366	0.427	**0.284**	**0.217**	**0.289**
Meta-embeddings AAEME triplet loss + wiki	0.328	0.386	**0.444**	0.279	**0.218**	0.268

As it can be seen in Table 11 and 12, candidates generated using node2vec with the linear projection come from completely irrelevant domains.

GCN autoencoder does not outperform the majority of the approaches for neither of languages despite being a holistic and self-sufficient approach aimed at combining word representations with the graph structure of taxonomy. The model assigns high probabilities to all synsets in the word's neighbourhood in the graph, whereas only direct and second-order hypernyms are the correct answer. Taxonomic "uncles", "siblings", "cousins", and other distant "relatives" are not welcome.

6.4. Overall Comparison

As it can be seen from Tables 9 and 10 meta-embeddings approaches are the unconditional winners for English nouns and for both Russian datasets. Moreover, they even obtain the best results among all approaches, which do not apply information from Wiktionary. The AAEME autoencoder with triplet loss was the best among meta-embedding methods under consideration. The features from Wiktionary further improved the results.

As for the *nouns 1.6-3.0* and *nouns 1.7-3.0*, we have a tie for first place between the approach combining search engines result and ranking+wiki approach (for *nouns 1.6-3.0*). Apparently, zero embeddings in word2vec and GloVe heavily distract model from predicting correct candidates.

The best approach from graph-based approaches *node2vec* (see Tables 7 and 8 for comparison) does not demonstrate decent performance on the task. All word methods using word embeddings (except for word2vec for nouns) outperform graph methods with a large margin. The approach combining fastText *Ranking + wiki*, *node2vec* and *Poincaré* is not promising either: incorporating graph-based features leads to an increase in scores for the Russian nouns and verbs datasets, whereas for the English dataset the approach does not yield any improvement except for the WordNet 2.0-3.0 dataset. Nevertheless, the combined method performs on par with the best RUSSE'2020 system. 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 *hypo2path rev* are not promising either. They perform more or less equal to the graph-based embeddings approaches like node2vec, Poincare and

Table 10
MAP scores for the taxonomy enrichment methods for the Russian datasets.

method	nouns		verbs	
	non-restricted	restricted	non-restricted	restricted
node2vec (top-5 fastText associates)	0.266	0.366	0.168	0.252
Ranking + wiki [17]	0.413	0.552	0.297	0.389
Ranking + wiki [17] + node2vec	0.410	0.558	0.293	0.383
Ranking + wiki [17] + node2vec + Poincaré	0.414	0.559	0.306	0.391
Ranking + wiki (word2vec)	0.266	0.443	0.348	0.451
hypo2path rev [13]	0.246	0.342	0.175	0.192
hypo2path [13]	0.061	0.097	0.160	0.172
TaxoExpan [12]	*0.007*	*0.006*	*0.009*	*0.009*
Top-1 RUSSE'2020 for nouns	0.393	0.552	0.293	0.436
Top-1 RUSSE'2020 for nouns, no search engine features	0.369	0.507	0.267	0.389
Top-1 RUSSE'2020 for verbs: [36]	0.288	0.418	0.340	0.448
Meta-embeddings SVD	0.435	0.579	0.382	0.442
Meta-embeddings AAEME triplet loss	0.474	0.581	0.352	0.437
Meta-embeddings SVD + wiki	0.452	**0.612**	**0.423**	0.458
Meta-embeddings AAEME + wiki triplet loss	**0.490**	**0.611**	0.400	**0.477**

GCN autoencoder. While looking through the candidates provided by the model we discovered the following limitations that could probably affect the model performance:

- if the hypo2path rev model incorrectly predicts the first direct hypernym the whole chain of hypernyms used as candidates can also be irrelevant;
- 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 [12] for some reason are significantly lower than the baseline. We do not consider those results credible and provided them in italics. According to our viewpoint, those results can be explained of the incorrect launching of the model from their GitHub repository[17].

In addition to the presented results we noticed an interesting outcome about the models comparison — their performance of different part-of-speech datasets. According to the Tables 9 and 10, all models except for *Ranking + wiki (word2vec)* provide better results for nouns rather than for verbs. We assume that it can be explained by the lemmatisation step applied before the word2vec training procedure. Apparently, it is beneficial for the representations of the verb infinitives.

7. Error Analysis

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

7.1. Comparison of Graph-based Approaches with Word Vector Baselines

In order to better understand the difference in systems performance and their main difficulties, we per-

[17] https://github.com/mickeystroller/TaxoExpan

Table 11
Prediction noun examples from the English v 1.6-3.0 dataset.

Francis_Joseph_I emperor.n.01, sovereign.n.01			
Poincaré	node2vec	node2vec projection	hypo2path rev
person.n.01	king.n.01	fish_genus.n.01	pope.n.01
entity.n.01	edward.n.02	genus.n.02	spiritual_leader.n.01
life_form.n.01	herod.n.01	mammal_genus.n.01	leader.n.01
causal_agent.n.01	arthur.n.02	city.n.01	person.n.01
worker.n.01	messiah.n.03	municipality.n.01	causal_agent.n.01
european.n.01	louis_xiii.n.01	arthropod_genus.n.01	entity.n.01
leader.n.01	louis_xiv.n.01	dicot_genus.n.01	
object.n.01	frederick_ii.n.01	asterid_dicot_genus.n.01	
ruler.n.01	belshazzar.n.01	animal_order.n.01	
animal.n.01	pyrrhus.n.01	asterid_dicot_genus.n.01	
GCN	fastText	combined (best)	AAEME tr loss
day.n.04	king_of_england.n.01	king_of_england.n.01	king_of_england.n.01
metallic_element.n.01	king.n.01	king.n.01	king.n.01
large_integer.n.01	pope.n.01	holy_roman_emperor.n.01	pope.n.01
semitic_deity.n.01	islamic_calendar_month.n.01	pope.n.01	composer.n.01
hindu_deity.n.01	holy_roman_emperor.n.01	deliberation.n.02	christian_holy_day.n.01
hindu_calendar_month.n.01	general.n.01	islamic_calendar_month.n.01	catholic.n.01
month.n.02	calendar_month.n.01	**emperor.n.01**	musician.n.02
anomalistic_month.n.01	**emperor.n.01**	missionary.n.02	saint.n.01
chemical_element.n.01	frank.n.01	frank.n.01	latter-day_saint.n.01
religionist.n.01	jew.n.01	gravida.n.01	spiritual_leader.n.01

formed quantitative and qualitative analysis of the results on the English nouns subset.

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. We list the candidate synsets predicted by different methods in Table 11. Underlined bold text denotes predictions of the model from the ground truth. They demonstrate the main features of the tested approaches. As we can see, the Poincaré embeddings retrieved by aggregating words from fastText provide too broad concepts which are clearly too far from the correct answers (“object”, “person”, “element”). GCN is too far from the correct answers in general, whereas node2vec results depend on the fastText embeddings and are semantically close to the ground truth synsets.

The candidates provided by fastText model combined with graph-based models features are quite sim-

Table 12
Prediction verb examples from the English v 1.6-3.0 dataset.

overreact			
react.v.01, act.v.01			
Poincaré	node2vec	node2vec projection	hypo2path rev
change.v.01	react.v.02	play.v.01	**act.v.01**
act.v.01	**react.v.01**	compete.v.01	
touch.v.01	pursue.v.04	utter.v.02	
judge.v.02	**act.v.01**	change.v.01	
change_magnitude.v.01	run_down.v.01	shape.v.03	
interact.v.01	backfire.v.01	compete.v.01	
think.v.03	buck.v.02	adjust.v.01	
affect.v.01	marry.v.02	correct.v.01	
tell.v.02	answer.v.02	fast.v.02	
participate.v.01	wrench.v.01	travel.v.01	
GCN	fastText	combined (best)	AAEME tr loss
retaliate.v.02	**act.v.01**	**act.v.01**	**act.v.01**
exacerbate.v.02	react.v.02	**react.v.01**	interpret.v.01
cramp.v.01	**react.v.01**	react.v.02	understand.v.01
respond.v.03	change.v.01	make.v.01	**react.v.01**
upset.v.01	affect.v.05	change.v.01	displease.v.01
upset.v.06	make.v.01	fear.v.02	change.v.01
dictate.v.02	dramatize.v.02	terrify.v.01	err.v.01
irritate.v.02	misjudge.v.01	take.v.06	act.v.09
hurt.v.04	change_state.v.01	misjudge.v.01	make.v.01
sedate.v.01	right.v.01	burn.v.01	make.v.03

Table 13
MAP scores for different categories of words for English and Russian datasets.

Method	**Nouns**				**Verbs**		
	NE	Short	Other	All	Short	Other	All
% in the data	38%	6%	61%	–	5%	95%	–
	WordNet 2.0 — WordNet 3.0 (fastText)						
Baseline	0.328	0.319	0.233	0.291	**0.444**	0.191	0.205
Ranking	0.424	0.348	0.288	0.339	0.296	0.208	0.213
Ranking + Wiki	**0.437**	**0.360**	**0.332**	**0.372**	0.411	**0.263**	**0.271**
	RuWordNet 1.0 — RuWordNet 2.0 (fastText)						
% in the data	25%	7%	70.7%	–	0%	100%	–
Baseline	0.251	0.165	0.337	0.309	-	0.232	0.232
Ranking	0.417	0.218	0.381	0.384	-	0.252	0.252
Ranking + Wiki	**0.436**	**0.230**	**0.416**	**0.414**	-	**0.295**	**0.295**

ilar to those generated by the fastText model without additional features. Therefore, it is reasonable that the difference in scores is minor. However, for some cases (like "emperor.n.01" and "react.v.01" in Table 11) graph vector representations slightly improve the ranking.

7.2. Performance on Different Classes of Words

We noticed that for certain words hypernym discovery is an easier task. In particular, named entities and some other categories of words seem less challenging for our models. To test that, we divide our datasets into several parts: named entities, short words (less than 4 letters) and the rest. We compute MAP separately for each of these groups (see Table 13).

The MAP scores vary significantly across groups. Since MAP for a dataset is an average of MAPs for individual words, we can directly compare scores for different subsets. Thus, we see that for both languages named entities are easier to find hypernyms for. This happens because their hypernyms often contain the word from the same named entity. For instance, the named entity "Massif Central" has "massif.n.01" as one of the true hypernyms. The performance on short nouns and short verbs also differs. Whereas short nouns are often polysemous (hence more challenging), short verbs have one sense, are uncommon, and their sense is sometimes deduced from their form (e.g. "to aah" — "to produce an 'aah' sound").

Finally, the performance on all other nouns and verbs which have no such lexical cues is lower than on the whole list of words. This trend is particularly marked for nouns where less challenging groups (NE) constitute less than two fifths in both datasets. Thus, in order to evaluate taxonomy enrichment models, we should check their quality on different groups of words.

7.3. 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 2 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.

7.4. 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;

Figure 2. Distribution of words over the number of senses.
(a) Russian dataset (nouns and verbs) (b) English dataset (nouns and verbs)

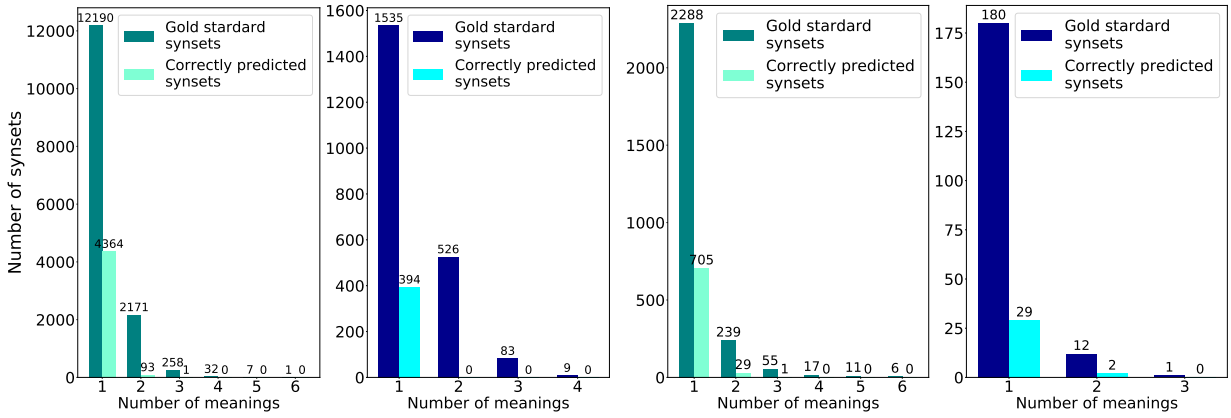


Figure 3. Manual datasets evaluation results: Precision@10.
(a) Russian dataset (b) English dataset

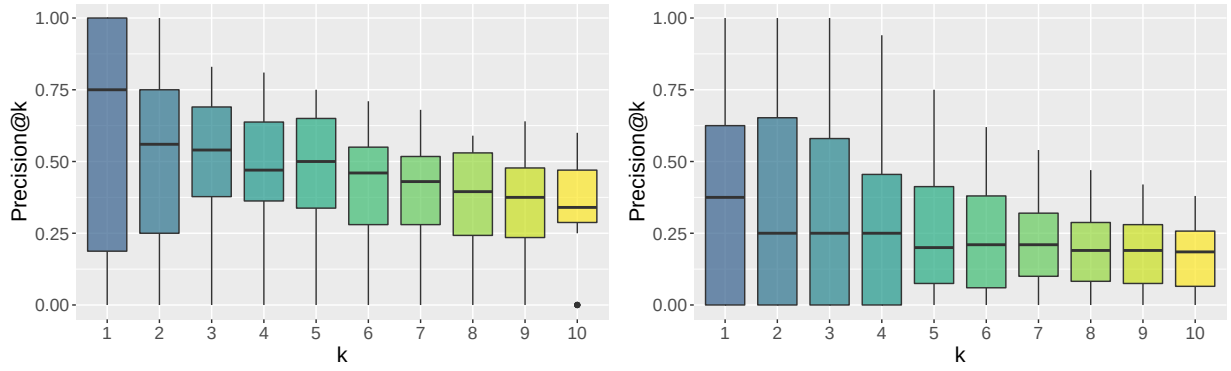


Table 14
Words selected for the manual evaluation.

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

– 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 not semantically close:

– 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).

The above mentioned mistakes and inaccuracies may dramatically decrease the scores of automatic metrics. 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 14). 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 3. We can see that even for words with MAP below 1 our model manages to extract useful hypernyms.

8. 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 in the future 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 4 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.

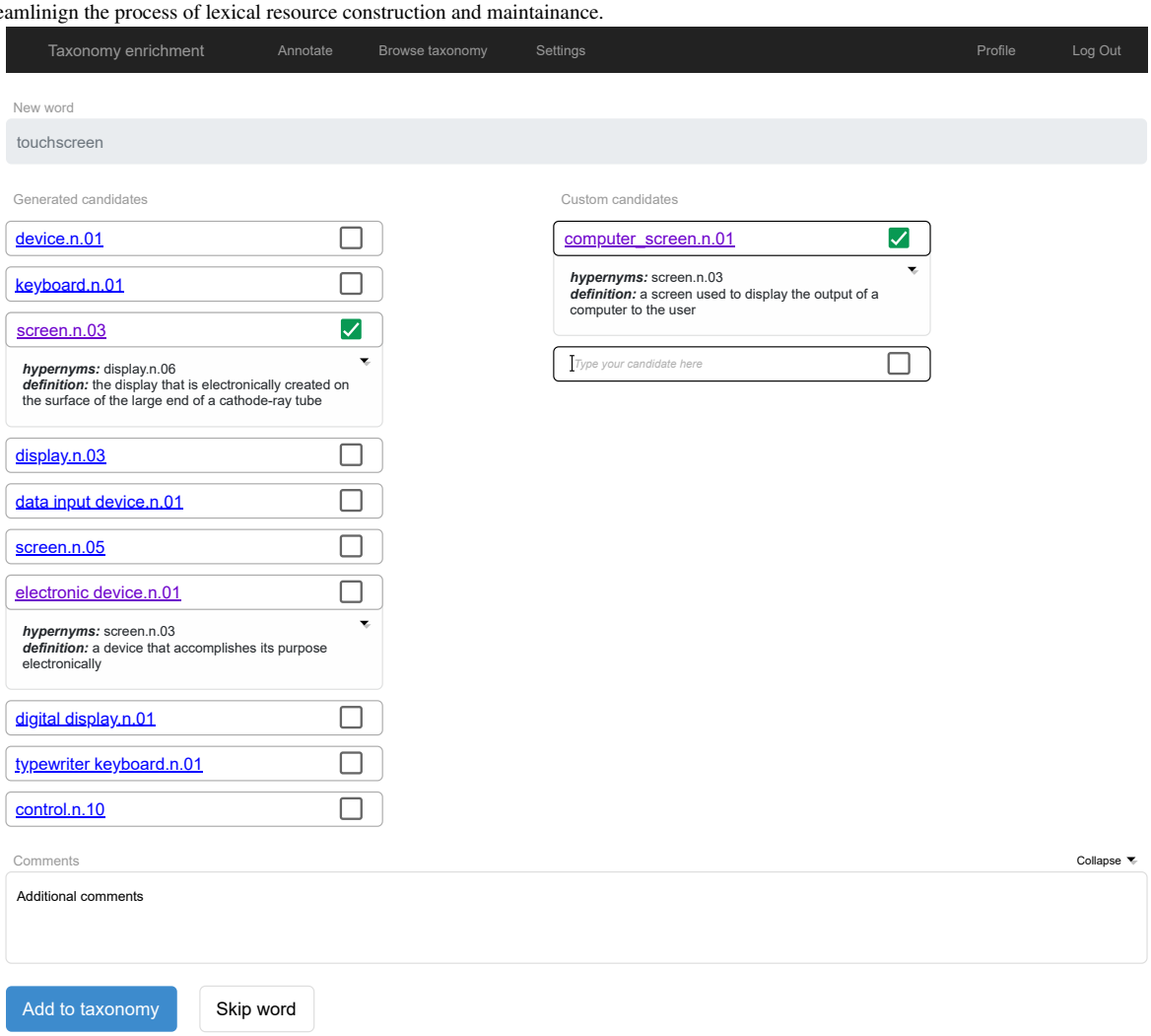
The described automatic methods and the presented interface allow much faster adaptation of existing taxonomic resources to new domains or new texts.

9. Conclusion

In this work, we performed comprehensive study of the methods for taxonomy enrichment using word-based and graph-based representations showing that word vectors achieve state-of-the-art results on all datasets. We tested approaches based on Poincaré and node2vec embeddings along with the approach based on graph autoencoder, sequence-to-sequence approach, meta-embeddings and search engine features to predict hypernym synsets for the input word.

Our results show that the use of word vector representations is much more efficient than any of the tested graph-based approaches. Moreover, our baseline method (candidates retrieved from fastText nearest neighbour list and ranked with features extracted from Wiktionary) does not benefit from graph-based methods. Namely, combining the baseline scoring function with Poincaré and node2vec similarities results in marginal improvements for some datasets, but this does not hold for all of them.

Figure 4. 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 streamlinign the process of lexical resource construction and maintainance.



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.

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 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 compelmentary information to the distributional signal contained in text corpora. One way to improve their performance, may be to use more sophisticated non-linear projection transofrmations from word to graph embeddings. Another promising way in our opinion is to explore the use of meta-embeddings to mix word and graph signals, e.g. GraphGlove [73]. Moreover, we find it promising to experiment with temporal embeddings such of those of [74] for the taxonomy enrichment task.

Last but not least, we find it promising 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 challanging problem in its own.

Acknowledgments

The work of Natalia Loukachevitch in the current study (preparation of data for the experiments) is supported by the Russian Science Foundation (project 20-11-20166).

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] B. Shi and T. Weninger, ProjE: Embedding projection for knowledge graph completion, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31, 2017.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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–.

[22] 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–.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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).

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

[50] 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.

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

[52] 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.

[53] 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.

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

[55] 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).

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

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

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

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

[60] 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.

[61] 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.

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

[63] 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.

[64] 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.

[65] 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).

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] Y. Kuratov and M. Arkhipov, Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language, *arXiv preprint arXiv:1905.07213* (2019).

[73] 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).

[74] 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.

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

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

[77] 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.

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

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

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

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

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

[88] 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.