Skolkovo Institute of Science and Technology

Learning Linguistic Tree Structures with Text and Graph Methods

Doctoral Thesis
by

Irina Nikishina

Doctoral Program in Computational and Data Science and Engineering

Supervisor
Assistant Professor, Alexander Panchenko

Moscow — 2022

I hereby declare that the work presented in this thesis was carried out by myself at Skolkovo Institute of Science and Technology, Moscow, except where due acknowledgement is made, and has not been submitted for any other degree.

Candidate (Irina Nikishina)

Supervisor (Prof. Alexander Panchenko)

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 the hypo-hypernym ("class-subclass") relationship. With the rapid growth of lexical resources for specific domains, the problem of automatic extension of the existing knowledge bases with new words is becoming more and more widespread. In this thesis, we address the problem of *Taxonomy Enrichment* which aims at adding new words to the existing taxonomy.

We formulate two task settings on the automatic taxonomy extension. The first one aims at predicting hypernyms ("parents, words with broader meanings") from taxonomy given a predefined list of new words with no definition. The second task setting considers taxonomy enrichment with no predefined candidates. We assume that the the compressed information from pre-trained language models like BERT can be leveraged to predict new words missing in taxonomic resources.

We suggest multiple approaches and datasets for each task setting. First, we present a new method called DWRank which allows achieving high results on this task with little effort. It uses the resources which exist for the majority of languages, making the method universal. We extend our method by incorporating deep representations of graph structures like node2vec, Poincaré embeddings, GCN and many more, which have recently demonstrated promising results on various NLP tasks. Furthermore, combining these representations with word embeddings allows us to achieve the state of the art results.

Secondly, propose the Cross-modal Contextualized Hidden State Projection (CHSP) method that combines graph-, and text-based contextualized representations from transformer networks to predict new entries to the taxonomy. We have evaluated the method suggested for this task against text-only baselines based on BERT and fastText representations. The results demonstrate that the incorporation of graph embeddings is beneficial for the task of hyponym ("children, narrower words") prediction using contextualized models.

Additionally, we conduct a comprehensive study of the existing approaches to taxonomy enrichment based on word and graph vector representations and their fusion approaches. We also explore the ways of using deep learning architectures to extend the taxonomic backbones of knowledge graphs. We create several datasets for taxonomy extension for English and Russian. We achieve state-of-the-art results across different datasets and provide an in-depth error analysis of mistakes. We hope the new challenging tasks presented in this work will foster further research in automatic taxonomy construction methods.

Publications

During my PhD studies, I was involved in the following publications. The main body of the dissertation is based on the publications listed only as "main author".

Main author

This section presents the main publications of the author related to the PhD dissertation.

1. **Irina Nikishina**, Mikhail Tikhomirov, Varvara Logacheva, Yuriy Nazarov, Alexander Panchenko, and Natalia Loukachevitch. Taxonomy enrichment with text and graph vector representations. *Semantic Web*, 13(6):441–475, 02 2022. doi:10.3233/SW-212955. URL https://content.iospress.com/download/semantic-web/sw212955?id=semantic-web%2Fsw212955 [Q1]

2. **Irina Nikishina**, Natalia Loukachevitch, Varvara Logacheva, and Alexander Panchenko. Evaluation of taxonomy enrichment on diachronic WordNet versions. In *Proceedings of the 11th Global Wordnet Conference*, pages 126–136, University of South Africa (UNISA), January 2021. Global Wordnet Association. URL https://aclanthology.org/2021.gwc-1.15 [SCOPUS]

3. **Irina Nikishina**, Varvara Logacheva, Alexander Panchenko, and Natalia Loukachevitch. Studying taxonomy enrichment on diachronic WordNet versions. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3095–3106, Barcelona, Spain (Online), December 2020b. International Committee on Computational Linguistics. doi:10.18653/v1/2020.coling-main.276. URL https://aclanthology.org/2020.coling-main.276 [CORE A]

4. **Irina Nikishina**, Varvara Logacheva, Alexander Panchenko, and Natalia 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"*, 2020a. URL https://www.dialog-21.ru/media/5111/nikishinaiplusetal-160.pdf [SCOPUS]

Co-author

This section contains publications prepared during graduate studies (not directly related to the PhD dissertation).

5. Varvara Logacheva, Daryna Dementieva, Irina Krotova, Alena Fenogenova, **Irina Nikishina**, Tatiana Shavrina, and Alexander Panchenko. A study on manual and automatic evaluation for text style transfer: The case of detoxification. In *Proceedings of the 2nd Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 90–101, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.humeval-1.8. URL https://aclanthology.org/2022.humeval-1.8 [SCOPUS]

6. Daryna Dementieva, **Irina Nikishina**, Varvara Logacheva, Alena Fenogenova, David Dale, Irina Krotova, Nikita Semenov, Tatiana Shavrina, and Alexander Panchenko. Russe-2022: Findings of the first russian detoxification task based on parallel corpora. In *Computational Linguistics and Intellectual Technologies*, 2022. URL https://www.dialog-21.ru/media/5755/dementievadplusetal105.pdf [SCOPUS]

7. Evgeny Kotelnikov, Natalia Loukachevitch, **Irina Nikishina**, and Alexander Panchenko. RuArg-2022: Argument Mining Evaluation. In *Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue"*, 2022. URL https://www.dialog-21.ru/media/5773/kotelnikoveplusetal111.pdf [SCOPUS]

8. Anna Safaryan, Petr Filchenkov, Weijia Yan, Andrey Kutuzov, and **Irina Nikishina**. Semantic recommendation system for bilingual corpus of academic papers. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 22–36. Springer, 2020. URL https://link.springer.com/chapter/10.1007/978-3-030-71214-3_3 [SCOPUS]

9. Andrey Kutuzov and **Irina Nikishina**. Double-blind peer-reviewing and inclusiveness in russian nlp conferences. In *International Conference on Analysis*

of Images, Social Networks and Texts, pages 3–8. Springer, 2019. URL https://link.springer.com/chapter/10.1007/978-3-030-37334-4_1 [SCOPUS]

Dedicated to my dog named SOTA.

Acknowledgments

First and foremost, I am enormously grateful to my supervisor Professor Dr. Alexander Panchenko for believing in a young computational linguist from the faculty of Humanities and accepting my candidature as a PhD student at Skoltech. It was an honor to work under your supervision and to gain from your vast experience. You were more than a supervisor, but a "spiritual mentor" who guided me on the thorny way of a PhD candidate. You helped me to overcome my impostor syndrome and even entrusted me with helping to organize the AIST-2020 and AIST-2021 conferences. And many thanks to Andrey Kutuzov who presented me to Alexander Panchenko at the AIST conference in 2017.

I am also deeply thankful for my "'PhD mom' — Varvara Logacheva, previously a postdoc at Skoltech — for being always aside, and for her gentle supervision. I am still excited by the way you masterfully edited our papers and your immense moral support during these three years. You were one of those who helped me to assuage my doubts about PhD and not quit in the middle of my studies.

I also express my gratitude to Natalia Loukashevitch, Mikhail Tikhomirov, and Elena Tutubalina for being part of the research presented in the current thesis. This dissertation would not be possible without them and their contribution. I am deeply grateful that we managed to establish a strong connection. I am hopeful that we can continue to collaborate.

During the last three years, Skoltech became my second home and an asylum, where I met wonderful and exceptional people. I want to thank all members of the SkoltechNLP group: Daryna Dementieva, Nikolay Babakov, Viktoria Chekalina, Anton Razzhigaev, and David Dale for spending three fantastic years together. I will remember our coffee breaks (especially the discussions in the kitchen) and lunches together, several Secret Santa rounds and group meetings.

Special thanks go to the neighbouring lab of Maxim Panov and especially to Kirill Fedyanin. Thank you for being my guiding light during the hardest year of my life and for motivating me to write the text of this thesis on a dare.

I could not forget to express my gratitude to the best master's student and the

sweetest girl at Skoltech – Alsu Vakhitova. She was the first student I helped to supervise and we worked a lot on one of the topics of this thesis. This dissertation would not be possible without her and her contribution. And my life is not possible now without being good friends with Alsu.

I am also grateful to Chris Biemann and Anh-Huy Phan for being part of my PhD studies as members of the Individual Committee. I also want to thank Chris for his support during the last month of my work on the thesis.

During my PhD studies, I realized that I have an enormous number of friends and terrific people around, with who I spend many great moments during the last three years. I know that I can rely on them at any moment. They are always ready to help me even with the proofreading of my thesis. Many thanks to Natasha, Daryna, Yulya, Nastya, Katya, Ira, Olya, Alyona, Alsu, Kirill, Danya, Masha, David, Anton, Dasha, Alina, Mikhail, Amir, Evgenii, Nikita, Oleg, Anastasia, Ivan, Vlad, Özge, Irina, Nikolay, Betty, Simon, Adrien, Henadzi, Kate, and many other beautiful people (so sorry if I forgot to mention you here) for the proofreading and for your support. Especially, I want to thank Sasha for believing in me as a PhD student and a researcher, which motivated me a lot to continue. Dear Natasha, you are my best friend and the love of my life, I would not be able to write the thesis without you.

I would never get to this point without my family: my parents Marina and Alexander, my brother Dmitry and my husband Alexey.

Finally, I would like to thank everyone who keeps fighting for justice at this dark period in the worldwide history.

Contents

List of Figures

List of Tables

Glossary

AAEME Averaged Autoencoded Meta-Embeddings. 50, 79–81, 92, 95, 153–155, 158, 159

AI Artificial Intelligence. 19

API Application Programming Interface. 126

BERT Bidirectional Encoder Representations from Transformers. 13, 14, 39, 44, 45, 58, 59, 97, 101–106, 108–111, 113, 115–122, 124, 125, 133

CAEME Concatenated Autoencoded Meta-Embeddings. 50, 79, 80, 92, 152, 158, 159

CC Common Crawl. 20, 85, 115

CHSP Cross-modal Contextualized Hidden State Projection. 3, 106, 110, 116–118, 123

DWRank Distributional Wiktionary-based synset Ranking. 3, 57, 82, 85, 86, 88, 90–95, 115, 131

ELMo Embeddings from Language Model. 44, 101

FFNN Feed-Forward Neural Network. 39, 40

GAT Graph Attention Network. 58, 88, 131

GCN Graph Convolutional Network. 52, 56, 57, 88, 89, 131

GNN Graph Neural Network. 52, 56, 58, 59, 88

GPT Generative Pre-trained Transformer. 44, 101, 133

Graph-BERT Graph-based BERT. 58, 59

GraphSAGE GraphSAGE (SAmple and aggreGatE). 88, 89

HD Hypernym Discovery. 38, 39, 45, 67, 116

HOPE High-Order Proximity preserved Embeddings. 52, 57, 131

ILI Inter-Lingual Index. 26, 27, 65, 70

KB Knowledge Base. 19–21, 24, 25, 37

KG Knowledge Graph. 22, 24, 51, 52

LOD Linked Open Data. 26, 27

LogReg Logistic Regression. 33, 45, 77, 78, 83, 84

LSTM Long Short-term Memory. 36, 46

MAP Mean Average Precision. 67, 69, 90, 91, 94, 95, 99

MLM Masked Language Modelling. 45

MLP Multi-layer Preceptron. 40, 108

NLP Natural Language Processing. 19, 45

NSP Next Sentence Prediction. 45

PPMI Parkinson's Progression Markers Initiative. 74

RNN Recurrent Neural Network. 41

SKOS Knowledge Organization Systems. 20

SLP Single-layer Preceptron. 40

SOTA state-of-the-art. 21, 22, 26, 40, 45, 57, 60, 78, 90, 91, 93, 100, 133

SVD Singular Value Decomposition. 50, 74, 79, 158, 159

SVM Support Vector Machine. 32, 33

TADW Text-Associated Deep Walk. 52, 53, 89, 92, 95, 131–133

TE Taxonomy Enrichment. 21–29, 37–40, 51, 52, 57, 60, 66, 67, 70, 71, 78, 79, 82, 86, 88, 90, 94, 101–103, 107, 116, 118, 122, 124, 131–135

TI Taxonomy Induction. 27, 37, 46, 51, 86

WBSR Web-based Synset Ranking. 76, 93, 158, 159

Chapter 1

Introduction

"You shall know a word by the company it keeps"

John Rupert Firth, linguist, (1957)

Natural Language Processing (NLP) is a vast subfield of Artificial Intelligence (AI) which aims at understanding and generating natural language. Even though modern intelligent systems and large-scale models are making good progress in imitating natural speech and demonstrating good results in Turing's test [French, 2000], we still cannot conclude that Artificial Intelligence (AI) has evolved. Moreover, it is expected that AI will not only be able to understand and generate phrases in natural language, but also have larger knowledge about the world than people do.

That is why people often try to build up words and phrases into some structure like Knowledge Base, (e.g., ontology or taxonomy). Such structures are aimed at storing world knowledge (relationships between objects) in a machine-readable format. Knowledge Bases, in particular ontologies and taxonomies, are widely used in multiple NLP tasks as the main source of knowledge. It is expected that the machine (and the Artificial Intelligence one day) will use such structures for understanding relationships between objects, searching for information for Question Answering and storing the acquired knowledge as the background for language understanding and generation.

The amount of data that can be used to construct such objects are usually quite large. For instance, it could be Wikipedia where the nodes are the Wikipedia pages

and the relations between them are links between pages. Moreover, we can think of all words in a language that should be organised in a hierarchical structure starting from the most abstract to the most concrete entities. The last but not the least is the Common Crawl that could also be organised in a machine-interpretable structure which is the central idea of the Semantic Web. For that, web pages should be linked to *ontologies* [Berners-Lee et al., 2001, Gómez-Pérez and Corcho, 2002] — databases which contain information on classes of objects, their properties, and relations between the classes. The relations between objects are particularly important in an ontology, as they form its structure. These relations can be of different types corresponding to different types of relationships between real-world objects. One of the most important relationships is the class-subclass relation. It allows the organization of entities into a *taxonomy* — a tree structure where entities are represented as nodes and the edges between are denoted as *subclass-of* or *instance-of* relationships. The class-subclass relations and taxonomies built from them are crucial for understanding the place, the purpose of an object or a concept in the world. This relation and the hierarchical structure also constitute a basis of many Knowledge Bases and *Knowledge Graphs* — a particular type of Knowledge Base in which objects are organised in a graph structure. There, the nodes of a graph are objects, and the edges of a graph are relations between the objects.

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

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

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

example, it can include part-whole relations.

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

Taxonomy is a graph structure where words are nodes and the edges between them are the hypo-hypernym relations between them. We choose taxonomies as the topic of our research because they are still not fully investigated. There exist papers on extending KBs, but very prior works implement or test graph methods on taxonomic structures. The specificity of taxonomies has been described in Carper and Snizek [1980] which implies that we might contribute to the field by proposing taxonomy-specific methods for graph extension.

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

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

First, we propose the **DWRank** method. Analogously to many existing methods, it uses only distributional information. Then, we enable this method to incorporate the different sources of graph information. We compare various approaches of getting the information from a Knowledge Graph. Finally, we present another enhancement of our method which allows for the successful combination of information from different sources, beating the SOTA.

To place our models in the context of the research on Taxonomy Enrichment, we compare them with several state-of-the-art models. To the best of our knowledge, this is the first large-scale evaluation of Taxonomy Enrichment methods. We are also the first to evaluate the methods on datasets of different sizes and in different languages (English and Russian).

1.1 Task Formulation

In this thesis, we focus on the idea of keeping valuable resources up-to-date and extending the existing taxonomies with automatic methods. In order to have a better understanding of the task, let us consider an example of taxonomy. Figure 1-1 demonstrates a subgraph for the word "Papuan" retrieved from WordNet. There, each concept has one or more *parents* (concepts which it is derived from). The parents in their turn have their own parents, and one can trace the word affiliation to the most abstract *root* concepts of the taxonomy. Here, the word "Papuan" is attached to the synsets "Indonesian" and "natural_language" as it can mean both an ethnicity and a language. Note that words can have multiple parents for different reasons. Two or more parents can point to the fact that a word has multiple meanings (as is the case

with “Papuan”). Alternatively, a word meaning can be a combination of meanings of two higher-level concepts.

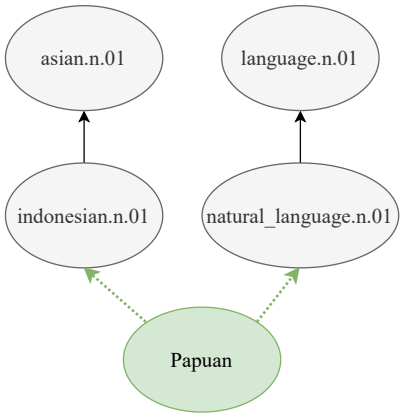


Figure 1-1: Example of adding a new word “Papuan” to the taxonomy

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

The task of finding a single suitable hypernym synset is difficult for a machine because the number of nodes in the existing taxonomy can be very large (e.g., in WordNet the number of noun synsets is around 29,000). Thus, a model trained to solve this task will inevitably return many false answers if asked to provide only one synset candidate. Moreover, as we can see from Figure 1-1, a word may have multiple hypernyms.

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

This is an established approach to tackle this task. The same formulation was also used in research works on Taxonomy Enrichment and in Taxonomy Enrichment shared tasks [Jurgens and Pilehvar, 2016, **Nikishina** et al., 2020a].

1.2 Terminology

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

Knowledge Base (KB) is a data structure used to store knowledge which comprises concepts and relations (known as TBox-es) and individuals, concepts, and relation instantiations (also known as ABox-es) [De Giacomo and Lenzerini, 1996].

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

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

Hypernymy relation is a relation $r \in R$. According to Miller [1998b], hypernymy relation exists between objects X and Y if native speakers accept sentences constructed using such patterns as "An X is a (kind of) Y". Hypernymy is transitive and asymmetrical. Such relations are central organizing relations for nouns and verbs in WordNet [Miller, 1998b]. In fact, hypernymy relations comprise class (or subsumption) relations and instantiation relations considered in ontology studies and description logics [McGuinness and Borgida, 1995].

Taxonomy is a special case of a Knowledge Graph G. It is a tree-based structure where nodes from the set E are connected with the hypernymy relation r ($R = \{r\}$).

Elements included in set E are words or concepts. In taxonomy G, they are arranged in a hierarchical structure from the most abstract concept (*root*) to the most specific concepts (*leaves*).

Query words (new words) — words to be added to the taxonomy T, usually manually collected by experts. In this work, we use the following notation for this set $Q \subseteq \{q_1, ..., q_n\}$. Note that words may be ambiguous: single q_i can correspond to multiple synsets s_i.

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

1.3 Research Questions

This thesis presents the research efforts which share the common topic of extending an existing taxonomy with new words. Together, they demonstrate the diversity of approaches for predicting hypernyms as well as the diversity of perspectives from which the task of Taxonomy Enrichment can be solved. The present thesis addresses several specific research questions which are presented below.

R1: The primary research question of this work asks **whether it is possible to predict hypernym for new words from an existing taxonomy**. We propose a new task setting in which new words come without definitions in comparison to SemEval-2016 task 14. It makes the prediction task more challenging and requires a range of experiments to investigate the problem.

R2: The secondary research question investigates **whether graph-based embeddings are beneficial for attaching new words to the taxonomy**. There exist multiple approaches making use of word embeddings, whereas the advantage of using graph embeddings has not been investigated.

R3: The tertiary research question is about **predicting new nodes in a taxonomy using a pre-trained language model**. We aim at extending the

existing taxonomy without predefined candidates. We want to test, whether the pre-trained language models contain enough pre-trained information to be able to predict new hyponyms at a certain place of the taxonomy.

We elaborate on these research questions in the next section while discussing objectives and contributions and in Conclusion (Chapter 9), where we summarise our answers to them.

1.4 Contributions

This thesis has the following contributions:

1. We present a new open Taxonomy Enrichment method **DWRank** which combines distributional information and the information extracted from Wiktionary (Chapter 5).

2. We present an extension of DWRank called **DWRank-Graph** which uses various graph-based representations via a common interface (Section 5.4).

3. We conduct a large-scale computational study of various approaches to Taxonomy Enrichment, which features multiple methods (including ours as well as state-of-the-art (SOTA) approaches), multiple datasets and languages (Section 6.1).

4. We present open datasets for studying the automatic enhancement of wordnets for English and Russian (Chapter 3).

5. We explore the benefits of combinations of embeddings and graph embeddings for the task of Taxonomy Enrichment (Section 5.5).

6. We provide an in-depth error analysis of different types of errors in the state-of-the-art models (Section 6.2).

7. We provide mappings to WordNet Linked Open Data (LOD) Inter-Lingual Index (ILI) [Bond et al., 2016] from Russian to English synsets. A dataset of

multilingual hypernyms opens possibilities for various use cases for cross-lingual operations on taxonomies (Subsection 3.2.1).

8. We propose a method for incorporating graph information into pretrained language models based on hidden contextualized state projection, aiming to predict candidates in a certain place in the taxonomy (Chapter 7).

1.5 Thesis Outline

The thesis is structured as follows.

Chapter 2 (Related Work) starts with the description of previous approaches to Taxonomy Enrichment in Chapter 2. It also describes the datasets used for the task and their shortcomings in terms of task formulation. Moreover, it sets the scene for our methods and approaches derived from previous experience. Section 2.4 (Background) outlines the current state of the field concerning taxonomies and graph-based approaches. It continues with the overview of the adjacent taxonomy-related tasks, like Taxonomy Induction and hypernym prediction. This chapter both positions our work in the broad academic context and proposes foundations for structuring the field itself by determining the axes along which the research on the topic can be meaningfully compared.

Chapter 3 (Diachronichal Dataset for Evaluation) describes the process of the creation and annotation of our new datasets. We present datasets for studying the evolution of wordnets for English and Russian, extending the monolingual setup of the RUSSE'2020 shared task with a larger Russian dataset and similar English versions. We provide mappings to WordNet LOD ILI from Russian to English synsets.

Chapter 4 (Competing Systems) chapter introduces existing approaches for attaching new words to the existing taxonomy. This chapter considers approaches both developed in parallel with the current research and the approaches proposed by other researchers but yet not tested on our dataset.

Chapter 5 (Distributional Wiktionary-based synset Ranking (DWRank)) explores an approach for attaching new words to the taxonomy based on embedding similarity features. We also describe various DWRank modifications, such as DWRank-Graph and DWRank-Meta, which are tested on a wide range of text and graph embeddings, as well as their combination.

Chapter 6 (DWRank Experiments) describes metrics used for the Taxonomy Enrichment task and the issues of ranking and evaluating predicted candidates amplified by the taxonomic structure. We compare the DWRank performance (as well as DWRank-Graph and DWRank-Meta performances) to the baselines and provide error analysis to evaluate whether the graph structure information is beneficial for the hypernym prediction.

Chapter 7 (Cross-modal Contextualized Hidden State Projection for Candidate-free Taxonomy Enrichment) addresses the problem of Taxonomy Enrichment from another viewpoint. First, it inverts the process of attaching new words to the taxonomy. We assume that the list of predefined candidates is excessive and new words can emerge from the pre-trained language models at a certain place in the taxonomy. Moreover, we provide a set of experiments and baselines and conclude candidate-free Taxonomy Enrichment to be a promising direction for further research.

Chapter 8 (System Description) presents the online service for searching and predicting words and synsets and visualising them in the taxonomic context.

Chapter 9 (Conclusion) is dedicated to a discussion of our results obtained and a detailed description of the contributions of each chapter. Moreover, we outline our future work directions. We also list all publicly available code, trained models and datasets produced in the course of work on the thesis.

Chapter 2

Related Work

This chapter sets the scene for our experiments in the field by describing previous research on Taxonomy Enrichment. We outline the existing task settings, datasets and approaches and demonstrate the limitations of previous setups.

2.1 Taxonomy Enrichment

Until recently, the only dataset for the Taxonomy Enrichment task was created under the scope of SemEval-2016. Moreover, all the existing approaches for the task are developed during the competition.

SemEval-2016 Task 14 [Jurgens and Pilehvar, 2016] is an evaluation framework for automatic Taxonomy Enrichment techniques by measuring the placement of a new concept into an existing taxonomy. Given a new word and its definition, systems were asked to attach or merge the concept into an existing WordNet concept. Five teams submitted 13 systems to the task, all of which were able to improve over the random baseline system. However, only one participating system outperformed the second, more competitive baseline that attaches a new term to the first word in its synset with the appropriate part of speech, which indicates that techniques must be adapted to exploit the structure of synsets.

The goal of Task-14 is to evaluate systems that enrich semantic taxonomies with new word senses drawn from other lexicographic resources. The task provides systems with a set of word senses that are not defined in WordNet. Each word sense

comprises three parts: a lemma, part of speech tag, and definition. For example, the noun "geoscience" is a word sense in our dataset which is associated with the definition "Any of several sciences that deal with the Earth". The word sense is drawn from Wiktionary. For each of these word senses, a system's task is to identify a point in the WordNet's subsumption (i.e., is-a) hierarchy which is the most plausible point for placing the new word sense. In other words, a system's task is to find the most semantically similar WordNet synset to the given new word sense. Once the target synset is identified, a system has to decide how to integrate the new word sense. For a given new word sense s and a target synset S we define two possible operations:

- MERGE: when s refers to the same concept that is conceptualized by the synset S. As a result of this operation s is added to the set of synonymous word senses in S.

- ATTACH: when s refers to a more specific concept than S. In other words, S is a generalization of the new word sense s (i.e., its hypernym). This operation creates a new synset containing the sole word sense s and attaches the new synset as a hyponym of S in the WordNet's subsumption hierarchy.

For each item in our datasets, we provide the source dictionary from which the corresponding word sense (i.e., a word and its definition) is obtained. The participating systems were allowed to use the source dictionary in order to draw additional information or exploit its structural properties. Based on their usage of the source dictionary, we classify the participating systems into two categories:

- Resource-aware: the participating systems could use the URLs provided in the dataset to gather additional information (e.g., hyperlinks, wiki-markup) for performing the integration and may use additional information from any dictionary, including the one from which the target word sense had been obtained, e.g., Wiktionary.

- Constrained: the system might use any resource other than dictionaries.

As the task setting suggests using definitions for the new words to predict the parent synset from the WordNet, all the participating teams made use of definitions

in their approaches. Nevertheless, we describe all of them to get a full picture of previous work done for the task discussing whether they could or could not be applied to the task we formulate in the present thesis.

Deftor team [Tanev and Rotondi, 2016] compute the definition vector for the input word, comparing it with the vector of the candidate definitions from WordNet using cosine similarity. In order to compute the WordNet synset vector, the authors concatenate non-stop words from the definition with lemmas from their glosses. TF-IDF weights calculated on definitions are used as values for such definition vectors.

The synset with the highest similarity is attached as its hyponym if the cosine similarity is higher than a certain threshold. Otherwise, the query word is skipped.

TALN team [Espinosa-Anke et al., 2016] also makes use of the definition by extracting noun and verb phrases for candidate generation. The authors exploit the mapping from BabelNet [Navigli and Ponzetto, 2010] to WordNet to obtain *SensEmbed* [Iacobacci et al., 2015] embeddings. Those embeddings are used to select candidates from WordNet. The following strategy is applied:

1. first, the definition of the query word is pre-processed: noun phrases are allocated with to each other, forming the multiword expression and stop words are excluded.
2. Then, the authors calculate the centroid vector for all found senses by simply averaging and normalizing all vectors.
3. As the first candidate, the closest to the centroid sense is selected (μ^{wn}).
4. As the second candidate, they select the word appearing at the first position of the definition sentence with the correct POS ($firstBest$).
5. The third candidate is the closest sense according to the cosine similarity ($highestBest$).
6. The last candidate is obtained by recalculating centroids, keeping only relevant senses according to the cosine similarity $\{\mu^{wn}_{best}\}$.

As a result, they have the following candidates:

$$C = C \bigcup \{\mu^{wn}, firstBest, highestBest, \mu^{wn}_{best}\}, \tag{2.1}$$

where C contains matching lemmas from synset and definition.

In order to rank candidates, the authors applied the following methods:

- **RunHeads** — they select the first word which has the same POS as the correct answer. If WordNet does not have this word, they select the one which is the closest to the centroid.
- **RunSenses** — selects the best candidate according to the voting system: it may occur that the first word is the most similar, which means that its weight is larger than the one from the similarity of SensEmbed embeddings. At the same time, if all candidates are equal, the closest (defined by cosine similarity) sense is selected.
- **RunHyps** — chooses the most general sense according to the tree structure of the taxonomy among candidates.

The authors did not outperform the first word-based baseline. However, their experiments report the upper bound of 0.7, which means that developed methods still demonstrate decent results.

MSejrKU team [Schlichtkrull and Martínez Alonso, 2016] presents an algorithm based on the Gaussian-kernel SVM and using both lexical and syntactic features. Additionally, the authors have used the following distributional features:

- word position,
- word shape for the query words and the WordNet senses,
- part-of-speech (POS) tag for the query words and the WordNet senses,
- dependency structure for the WordNet senses,
- binary feature representing the description word appearing in the target term,

- some features are used only for direct classification:
  - overlap between the synset and the term description,
  - the length of the shortest path to the description;
- the following pairwise features are used for ranking systems:
  - relative distance and position in the description sentence,
  - relative distance and position in the dependency tree,
  - difference in the number of overlapping lemmas between the description sentence and the senses which appear to be the most similar,
  - cosine distance between the query word (or phrase) and the most similar senses: Mikolov et al. [2013c] train a SkipGram model on a corpus of glosses based on WordNet and the Wikipedia entries for each term and the English part of the PolyGlot corpus [Al-Rfou' et al., 2013]. The embeddings of the multi-word expressions are obtained by averaging the word embeddings from a term or a synset.

MSejrKU team tests two strategies: direct classification and ranking approach. The first strategy provides correct integration for the query term. Then the synsets are sorted according to the prediction probabilities. The top-1 synset is considered as the answer. The authors use Logistic Regression and a Single Layer Feed-Forward Neural Network for classification. The second strategy utilizes the Gaussian kernel SVM to learn the relationships between the query term and words from the description sentence. Using the classifier predictions, we can reorder the candidates and choose the best candidate for the query term.

The results outperform most of the other approaches, as well as the simple baseline. They have the same (or slightly better) performance as the strong baseline.

VCU team [McInnes, 2016] relies on word overlapping and the dictionary features along with the word and synset definitions. The authors propose three VCU systems: VCU-Run-1 with the Lesk measure [Lesk, 1986]; VCU-Run-2 which

uses the First-order vector measure; and VCU-Run-3 which uses the Second-order vector measure.

The Lesk measure [Lesk, 1986] counts the number of overlaps between two definitions of two terms. The final measure is calculated as:

$$Lesk = \sum_{o_i \in O} o_i^2, \tag{2.2}$$

where $O = [o_1, o_2, ..., o_N]$ is the list of overlaps, N is the number of overlaps between the definitions of two terms.

Duluth team [Pedersen, 2016] presents four systems built upon each other. Duluth approaches use scoring of the overlaps between WordNet and new word input (OtherDict) glosses. To provide the final score, they calculate the square of the number of words in the overlap, and then all the overlaps between a pair of glosses are summed. The authors do simple preprocessing — converting to lowercase and keeping only alphanumeric symbols.

Duluth2 represents each word sense as its preprocessed gloss with stop words and single-character words excluded. The same is done for the input word and its gloss to compute the overlap score.

Duluth1 is an extension of Duluth2, where each WordNet gloss is expanded by concatenating to it the glosses of the hypernyms, hyponyms, derived forms and meronyms of the sense. The size of such combined descriptions can be too long and dramatically increase the computation time. Therefore, the authors limit the number of words in the gloss to nine and remove words consisting of four or less characters. However, in comparison to Duluth2, they do not exclude stop words during preprocessing. The same procedure is applied to the query words with glosses from OpenDict.

Duluth4 can be viewed as a modification of the two previous approaches. First, it excludes stop words like Duluth2. Second, it expands the main gloss only with hypernym and hyponym glosses rather than glosses of other related synsets.

As a result, the final WordNet gloss (also limited to nine tokens) becomes more specific.

The last method Duluth3 splits both WordNet and OpenDict glosses into trigrams (spaces are excluded beforehand) in order to allow matches of subwords. In this setup, stop words are not taken into account and the authors restrict the number of trigrams to 250.

All these models outperform the random baseline. However, they all yield the strong baseline on using the first word as the correct answer.

UMNDuluth team [Rusert and Pedersen, 2016] describes an approach based on definition overlaps and does not rely on sophisticated machine learning or deep learning approaches.

The authors iterate through each word in the query word definition and retrieve information about this word from WordNet (definition, synset, hyponyms, hypernyms, etc.). Then they check the overlap between the words in the query word definition and the definitions where this word presents, as well as in hypernyms and hyponyms of this word. The final score of the sense is the number of overlaps divided by the total length of the query word definition ($glossLength$):

$$finalScore = \frac{synsetO + hypernymO + hyponymO + lemmaO + bonusO}{glossLength}, \tag{2.3}$$

where $synsetO$ stands for overlaps in the synset gloss, $hypernymO$ stands for overlaps in the hypernym(s) gloss(es), $hyponymO$ stands for overlaps in the hyponym(s) gloss(es), $lemmaO$ stands for sense gloss. Bonus is calculated as follows: $2 \times lemmaLength$, where $lemmaLength$ is the length of the lemma from the query word gloss making part of a candidate sense.

Next, the authors verify that the chosen sense is more suitable than the other senses of the same lemma. They proceed by checking whether the candidate

sense overlaps with the lemma gloss. In the case it does not, the most common sense is chosen.

The choice between merging or attaching the new word is made according to the frequency of the selected candidate sense (attach if $freq = 0$, merge otherwise).

2.1.1 WordNet Path Prediction

A completely different approach to make use of fastText embeddings is presented in the work of Cho et al. [2020]. The authors experiment with encoder-decoder models in order to solve the task of direct hypernym prediction. They use a standard LSTM-based sequence-to-sequence model [Sutskever et al., 2014] with Luong attention [Luong et al., 2015]. First, they average fastText embeddings for the input word or phrase and put it through the encoder. The decoder sequentially generates a chain of synsets from the encoder hidden state, conditioned on the previously generated ones. The authors consider two different setups:

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

To be able to apply this sequence-to-sequence architecture to our data, we build new datasets similar to the ones described in Cho et al. [2020]. We generate a path from the WordNet starting from the root node to the target synset or word. Analogously to the original work, we include multiple paths from the root to the parents of the query word. We filter the validation set to only include queries that do not occur anywhere in the full taxonomy paths of the training data. To sort candidates generated by the decoder, we enumerate the generated *hypo2path* sequence from the right to the left or the *hypo2path reverse* from the left to the right and get the first 10 synsets.

Additionally, we extend this approach by replacing the LSTM with attention architecture with the Transformer architecture [Vaswani et al., 2017]. During the

training, we provide an embedding of a synset as input to the Transformer and expect the model to generate a sequence of synsets starting from the hypernym of the input synset. During inference, we provide embeddings of query words as input and expect the model to output sequences of synsets starting with the direct hypernyms.

This scenario may be unrealistic for manual annotation because linguists or domain experts write definitions for new words and add them to the taxonomy at the same time. 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").

2.2 Taxonomy Induction

The Taxonomy Induction (TI) problem [Bordea et al., 2015, 2016, Velardi et al., 2013] is about the creation of taxonomy from scratch. This task aims to extract hypernym-hyponym relations $\{v_i, r, v_j\} \subseteq V \times R \times V$ (where R is the hypo-hypernym relation set) between a given list of domain-specific terms $V \subseteq \{v_1, ..., v_n\}$ and then construct a domain taxonomy T based on them.

The authors of the most famous papers on the topic Pocostales [2016], Aly et al. [2019] solve the SemEval-2016 Task 13 on Taxonomy Induction relying on word embeddings: they compute the vector offset as the average offset of all the pairs generated and exploit it to predict hypernyms for the new data.

Another task, which is closely related to taxonomy creation, is the Knowledge Base construction. There exist multiple approaches to solving the problem: Text2Onto [Cimiano and Völker, 2005] a pillar language-independent approach which applies user interaction or fully automated methods [Dessì et al., 2021, Osborne and Motta, 2015] which apply text-mining tools and external sources, which are mostly applied for the scholarly domain. However, the KB completion task assumes a generic graph while the Taxonomy Enrichment task deals with tree structures and some specific methods of tree processing are commonly used in this field. In this thesis, we focus

on this specific task and narrow down our scope to enrichment and the population of taxonomic structures. It should be noted however that the majority of the ontologies and knowledge bases possess some kind of taxonomic backbone and therefore the task of constructing and maintaining such a semantic structure is fundamental.

Bordea et al. [2015, 2016] evaluate taxonomy construction models based on the extracted hypernym relations. The evaluation is performed for several domains. Gold standard datasets are collected from WordNet and EUROVOC thesaurus[1]. The authors suggest several metrics tailored for taxonomy evaluation. Levy et al. [2015] suggest that the results achieved in classification settings of hypernym extraction are mainly explained by the so-called "lexical memorization phenomenon" — a situation when models learn that in a relation "x is-a y" a word y is a prototypical hypernym. For example, if a classifier obtains many positive examples with the word $y = animal$, it may learn that anything that appears with y=animal should generate a positive answer. Camacho-Collados [2017] argue that hypernym classification is not a realistic scenario. Instead, hypernym-oriented evaluation should be organized as a Hypernym Discovery (HD) task, i.e., given a word *dog*, the system should be able to discover its hypernyms *mammal* or *animal* among a large number of other possible candidates. The author suggests evaluating performance of the models in this task with information-retrieval evaluation measures such as mean reciprocal rank (MRR) or mean average precision (MAP).

2.3 Hypernym Discovery Problem

Compared to the above mentioned competitions, RUSSE'2020 is closely related to the SemEval-2016 Taxonomy Enrichment Task [Jurgens and Pilehvar, 2016] and SemEval-2018 Hypernym Discovery Task [Camacho-Collados et al., 2018]. As in the mentioned SemEval tasks, in the competition the participants are asked to attach new words to the existing synsets, to create a ranked list of hypernym candidates, and the performance is evaluated using MAP and MRR metrics.

Formally, the Hypernym Discovery (HD) task aims to identify all hypernym terms

[1]Eurovoc: http://eurovoc.europa.eu/drupal

$y_1, ..., y_k$, $y \in Y$ for a given hyponym $x \in X$, that make (x, y) a hypernymy relation.

The Hypernym Discovery (HD) problem [Camacho-Collados et al., 2018] formulates the task as follows: 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.

In the Hypernym Discovery task at SemEval 2018 [Camacho-Collados et al., 2018], the organizers attempt to improve the quality of evaluation and formulated the hypernym extraction task as a ranking task. They create a list of hypernym candidates — these are all unigrams, bigrams, and trigrams that occurred more than N times (for example, 5 times in the corpus). For each of the new words and phrases, the participants are asked to rank the hypernym candidates by their relevance. Moreover, the participants have to find as many hypernyms as possible. The gold standard list of answers contains hypernyms of all hierarchy levels excluding only the most abstract concepts such as "entity". for each input (word or phrase), hypernyms are extracted at all hierarchy levels of gold-standard resources with the exclusion of the most abstract concepts such as "entity".

2.4 Background

This section is devoted to the theoretical background of the tools we apply in our methods for Taxonomy Enrichment. We discuss the most important technical details here, which do not make any contribution to the thesis, however, are still important for immersing in the topic and understanding the approach. We omit some advanced derivations and refer the reader to the original papers.

This section is divided in two parts. The first subsection describes machine learning models and neural network architectures used in the paper: Feed-Forward Neural Network and BERT. The second subsection focuses on the history of embeddings used in the paper and some technical details on vector representations for texts and graphs. We mention the models based on their application in the paper and not on chronological order.

2.4.1 Neural Networks

Nowadays, deep neural networks have become a standard tool for natural language processing. In numerous NLP tasks, such as named-entity recognition (NER), semantic role labeling (SRL), and POS tagging, the deep learning architecture surpasses most techniques that involve shallow machine learning models and hand-crafted features [Young et al., 2018]. We provide a comprehensive overview of deep learning models and approaches used in the Taxonomy Enrichment task. We start with the most basic neural network architecture: Feed-Forward Neural Network (FFNN). Then we provide an overview of the most popular modern architectures that are used in many state-of-the-art works – Transformers.

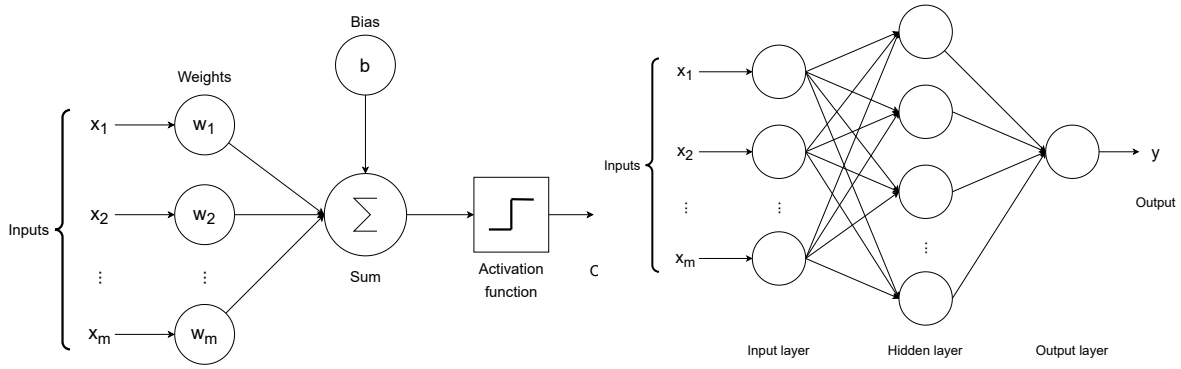
Feed-forward Neural Network

Neural networks first introduced to the field of language modeling were based on a feed-forward architecture [Bengio et al., 2003]. Schwenk [2012] has shown that a feed-forward neural network can be used successfully in phrase-based statistical machine translation, which achieved better performance than baseline statistical machine learning (SMT) system [Koehn et al., 2007].

A Feed-Forward Neural Network (FFNN) is an artificial neural network in which the connections between nodes do not form cycles or loops. The information in this network flows in forward direction: from the input nodes to the output nodes, passing through hidden nodes. Two types of FFNNs are usually distinguished: Single- and Multi-Layer Perceptron. They are both illustrated in Figure 2-1.

In a Single-layer Preceptron (SLP) inputs are multiplied by learnable weights as soon as they enter a network. Then, their weighted sum is compared to a threshold value to produce an output (0/1 or $-1/1$). SLPs are only capable of operating on linearly separable data and are often used in simple classification tasks.

Multi-layer Preceptron (MLP) consists of three types of layers: input, output and one or more hidden layers. Each node in a layer is directly connected to all other neurons in the next layer. The units of these networks have a linear activation function, which can be interpreted as an abstract representation of the (biological) neuron's rate of action potential firing. Hornik et al. [1989] have proven that MLP



(a) Single-layer perceptron. (b) Multi-layer perceptron.

Figure 2-1: Two types of feed-forward neural network.

with as little as one layer can arbitrarily closely approximate any continuous function. Moreover, MLP can work with not linearly separate data.

Transformers

Recurrent Neural Network (RNN) is sequential in nature, which inhibits parallelization. The authors have addressed this bottleneck in Vaswani et al. [2017] by introducing a new architecture called Transformer, which became groundbreaking in the field of NLP. The key idea of the model is to get rid of recurrence completely and rely solely on a new type of attention known as self-attention. Self-attention relates distinct positions within a sequence to compute a representation of it.

In comparison to RNNs, this mechanism allows for more parallelization than RNNs and therefore reduces training times. and can model longer contexts because each token attends to all the tokens in the input sequence. As a result, Transformers can be trained comparatively faster on larger amounts of data.

Transformers have classical encoder-decoder architecture. It is illustrated in Figure 2-2. The encoder consists of stacked identical two-part layers. The first part is a multi-head attention mechanism, the second – fully connected feed-forward network. Moreover, each block contains a residual connection. The decoder is composed of stacked identical three-part layers. The additional sub-layer is masked multi-head attention, which attends only to previously generated parts of the output sequence and masks subsequent positions. This is done to ensure that the new prediction is based only on known output that was predicted before it. In Vaswani et al. [2017]

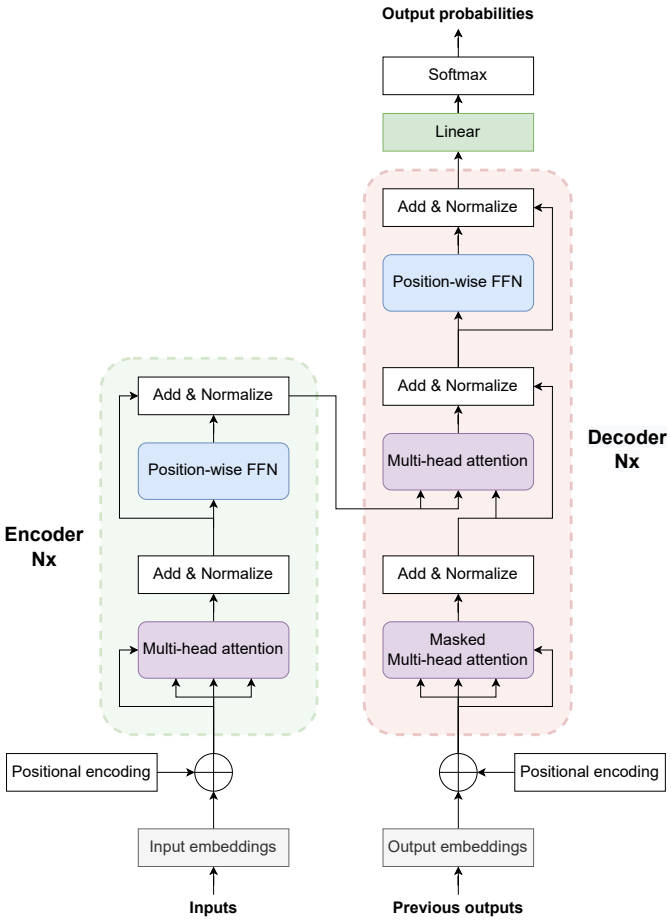
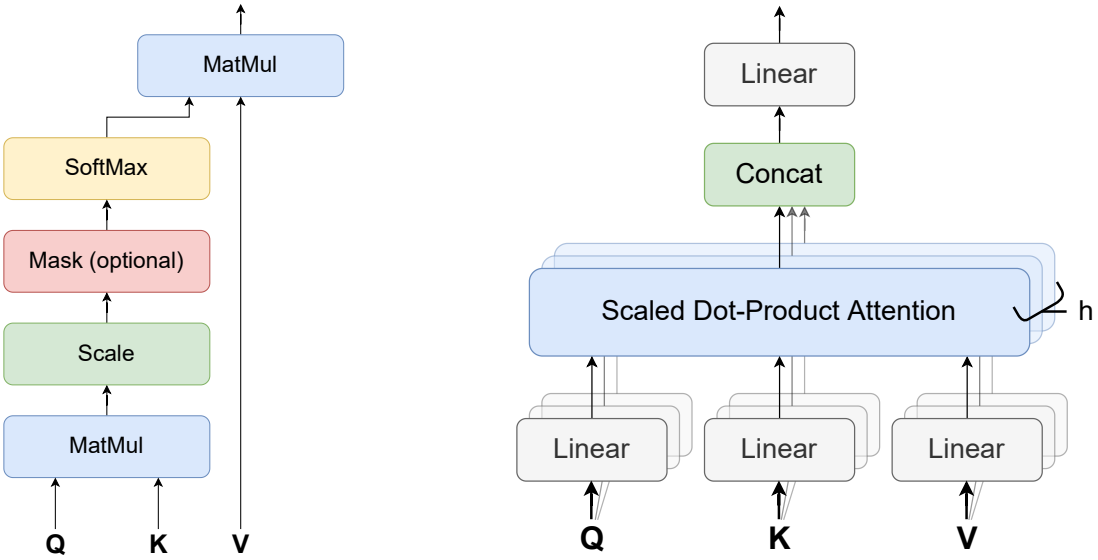


Figure 2-2: Encoder-decoder architecture of transformer model.

encoder and decoder contain 6 multi-part layers each. In following paragraphs we will look into detail of each encoder/decoder "building block".

Scaled dot-product attention is an attention sub-type invented by Vaswani et al. [2017]. It is illustrated in Figure 2-3a. The encoder takes embeddings matrix X as input and multiplies it by learnable weight matrices W^Q, W^K, W^V to get Q (query), K (key), and V (value) matrices. This triplet is an input to scaled dot-product attention. Queries and keys have dimension d_k, and values have dimension d_v. The square root of the former is used as a scaling factor in order to counteract vanishing gradient problem that might appear while applying softmax function on matrices with large d_k values. The output of the scaled dot-product attention is computed as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2.4}$$



(a) Scaled dot-product attention. (b) Multi-head attention.

Figure 2-3: The transformer building blocks.

Furthermore, the authors propose repeating the same projection and passing through scaled dot-product attention. This mechanism is called multi-head attention. It is illustrated in Figure 2-3b. The idea is to simultaneously calculate several attentions for different representation subspaces. Each attention head is represented by a set of three weight matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}, i \in h$. This means that the layer is able to attend to relevant tokens at different definitions of relevance. In Vaswani et al. [2017] following formula is given:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat\left(head_1, \ldots, \text{ head }_{\text{h}}\right) W^O, \\ \text{where head } &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right). \end{aligned} \tag{2.5}$$

Since transformers have discarded recurrence from the architecture, the model processes all tokens in a sentence simultaneously and has no way of knowing their order. Authors of Vaswani et al. [2017] have proposed a way of injecting back this knowledge into the model by summing positional encoding with the input embeddings. The authors propose the following formulae:

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d_{\text{model}}}\right), \end{aligned} \tag{2.6}$$

where *pos* is the position in an input sequence and i is the dimension of the output embedding space in range d_{model}. These formulas map each position to a vector of alternating sine and cosine functions. The proposed method has a number of advantages:

- the values of both sine and cosine functions are bounded, hence the encoding can be generalized to longer sequences;
- each time-step gets a unique encoding;
- since the distance between two positions is consistent, the positional encoding provides a way of measuring similarity between words.

Nowadays, the vast majority of NLP solutions involve transformers. A variety of architectures has been proposed since: XLNet [Yang et al., 2019], RoBERTa [Liu et al., 2019], ALBERT [Lan et al., 2020], BART [Lewis et al., 2020], GPT Brown et al. [2020], etc. In our research we use the classical Bidirectional Transformer architecture called BERT [Devlin et al., 2019]. The detailed overview of BERT can be found in 2.4.1.

Bidirectional Encoder Representations from Transformers (BERT)

Peters et al. [2018] gave rise to contextualized embeddings with their model called Embeddings from Language Model (ELMo). Although it uses bidirectional language models, the final concatenation of representations results in limited ability to leverage both right and left contexts simultaneously. This constraint was alleviated in Devlin et al. [2019]. The authors have proposed Bidirectional Encoder Representations from Transformers (BERT). The major advantage of BERT is that after pre-training phase it can be easily fine-tuned using labeled data for a variety of downstream tasks.

BERT input and output are represented using WordPiece embeddings Wu et al. [2016]. Intuitively, this approach is somehow similar to dividing words into n-grams. However, WordPiece starts with initializing the vocabulary as all the characters present in the data and gradually learns a given number of merge rules for symbols. After separating a word into wordpieces, each token is represented by its vocabulary id.

Moreover, each sequence starts with special classification token `[CLS]` and different sentences are separated by special `[SEP]` token.

BERT is based on transformer architecture. It is trained successively on two unsupervised tasks: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). The MLM task involves masking random tokens in the input, and the goal of the model is predicting them. Intuitively, this task allows training of deep bidirectional representations that are aware of the whole sentence. Specifically, the id of a masked token is calculated by feeding the final hidden representation of the token into a softmax over the vocabulary. The NSP task is used to improve BERT's performance on tasks that are based on understating of relationships between sentences (e.g., question answering). In this task, the objective is to predict a binary label (`IsNext`/`NotNext`) whether two sentences are successive. For this prediction only the hidden representation of `[CLS]` is used. The authors have performed separate experiments which proved that both MLM and NSP training is important and absence of one of them hurts performance significantly.

When BERT was published, it achieved state-of-the-art performance on eleven natural language understanding tasks. Pre-trained on massive amounts of unlabeled data configurations of BERT were made available for public. It can be used both as is to create text embeddings or finetuned with less resources on smaller datasets for a downstream task. BERT has become a common baseline in NLP experiments in just over a year, and it has prompted multiple research examining the model and proposing various enhancements Rogers et al. [2020].

2.4.2 Word Vector Representations for Taxonomies

Approaches using word vector representations are the most popular choice for all tasks related to taxonomies. When solving the Hypernym Discovery problem in SemEval-2018 Task 9 [Camacho-Collados et al., 2018], most of the participants use word embeddings. For instance, Bernier-Colborne and Barrière [2018] predict the likelihood of the relationship between an input word and a candidate using word2vec embeddings. Berend et al. [2018] use Word2vec vectors as features of a Logistic Regression classifier. Maldonado and Klubička [2018] simply consider the top-10

closest associates from the Skip-gram word2vec model as hypernym candidates. Pre-trained GloVe embeddings [Pennington et al., 2014] are also used to initialize embeddings for an LSTM-based Hypernymy Detection model [Shwartz et al., 2016].

Participants also solve the SemEval-2016 Task 13 on Taxonomy Induction with word embeddings [Pocostales, 2016]: 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 Aly et al. [2019] the authors apply word2vec embeddings similarity to improve the approaches of the SemEval-2016 Task 13 participants.

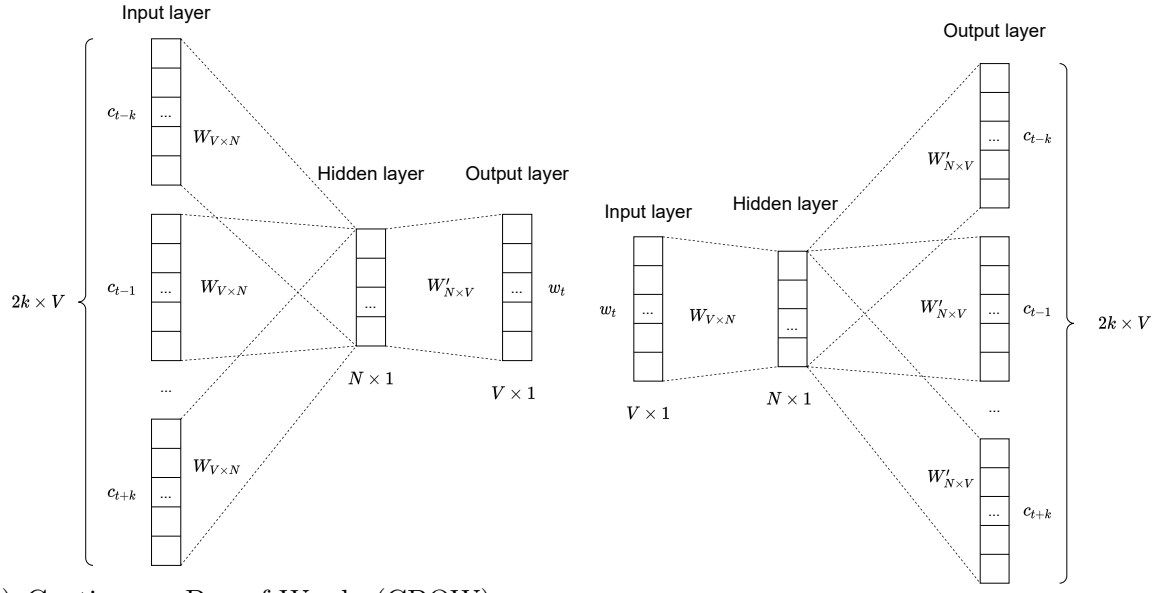
The vast majority of participants of SemEval-2016 task 14 [Jurgens and Pilehvar, 2016] and RUSSE'2020 [**Nikishina** et al., 2020a] also apply word embeddings to find the correct hypernyms in the existing taxonomy. For instance, the participants compute a definition vector for the input word by comparing it with the definition vectors of the candidates from the wordnet using cosine similarity [Tanev and Rotondi, 2016]. Another option is to train word2vec embeddings from scratch and cast the task to the classification problem [Kunilovskaya et al., 2020]. Some participants compare the approach based on the XLM-R model [Conneau et al., 2020] with the word2vec "hypernyms of co-hyponyms" method [Arefyev et al., 2020]. 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 taxonomy-related tasks and should be considered a strong baseline [Camacho-Collados et al., 2018, **Nikishina** et al., 2020a]. In the following paragraphs, we describe two word vector representation models that we further use in our experiments.

Word2vec

The main goal of Mikolov et al. [2013a] is the development of simple and fast architectures for learning word embeddings. The authors propose two models:

- Continuous Bag-of-Words (CBOW) model: predicts the target word given its context;
- Skip-gram model: predicts the context given the target word.



(a) Continuous Bag-of-Words (CBOW) model.

(b) Skip-gram model.

Figure 2-4: word2vec architectures.

Both models operate on word-context pairs (w, C), where w – target word, $C = \{c_{t-k}, \ldots, c_{t-1}, c_{t+1}, \ldots, c_{t+k}\}$ — set of all w's context words. Figure 2-4 depicts the difference between the architectures. The CBOW takes $2k$ (size of the context window) $1 \times V$ one-hot vectors (V — size of the vocabulary) and produces one $V \times 1$ vector. The skip-gram performs a reverse process. The weights between input and hidden layers is a $V \times N$ matrix W (N — dimension of embeddings), between hidden and output layers — $N \times V$ matrix W'. Then the posterior multinomial distribution of words can be obtained using softmax:

$$p\left(w_O \mid w_I\right) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{j=1}^{V} \exp\left(v'^{\top}_j v_{w_I}\right)}, \tag{2.7}$$

where w_I — input word, w_O — target word, v_w is a vector representation of word w from rows of W and v'_w is a vector representation of word from columns of W'. After the training process, the matrix W is repurposed as a source of word embeddings.

The denominator of the Equation 2.7 contains sum over all vocabulary, which is extremely inefficient. Initially, Mikolov et al. [2013a] proposed using hierarchical softmax for computation optimization. Subsequently, they came up with a simpler solution — negative sampling. The idea is to minimize similarity of words in different

contexts over a handful of words, not the whole dictionary. For a pair of two words (w, c) the probability that they came from training data is $p(D = 1|w, c)$. Consequently, the opposite probability would be $p(D = 0|w, c) = 1 - p(D = 1|w, c)$. Denote model's trainable weights as θ and data-points out of training data as D'. Then, the optimization function is:

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(D = 1 \mid w, c; \theta) \prod_{(w,c)\in D'} p(D = 0 \mid w, c; \theta). \tag{2.8}$$

Converting products to sums of logarithms and substituting the probability coming not from training data yields:

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log p(D = 1 \mid w, c; \theta) + \sum_{(w,c)\in D'} \log(1 - p(D = 1 \mid w, c; \theta)) \tag{2.9}$$

The probability $p(D = 1|w, c; \theta)$ can be computed using sigmoid function:

$$p(D = 1 \mid w, c; \theta) = \sigma\left(v_c v_w\right) = \frac{1}{1 + e^{-v_c \cdot v_w}}. \tag{2.10}$$

Furthermore, the Equation 2.9 takes form:

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log \sigma\left(v_c v_w\right) + \sum_{(w,c)\in D'} \log\left(\sigma\left(-v_c v_w\right)\right). \tag{2.11}$$

Rewriting the Equation 2.11 for one pair $(w, c) \in D$ and k pairs $(w, c_j) \in D'$:

$$\log \sigma\left({v'_{w_O}}^T v_{w_I}\right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma\left(-{v'_{w_i}}^T v_{w_I}\right)\right], \tag{2.12}$$

where $P_n(w)$ – noise distribution from which negative samples are drawn. Maximization of the Equation 2.12 leads to increasing similarity between words in the same context and decreasing similarity between words from different contexts. Since the computation is performed only over k negative samples, the training process is much faster as opposed to summing the whole vocabulary.

In practice, Skip-gram is more suitable for small monolingual datasets, while CBOW is faster and more convenient for larger datasets [Mikolov et al., 2013b].

According to Mikolov et al. [2013a], word2vec achieved much higher semantic and syntactic accuracy than other neural network language models such as Collobert and Weston [2008] and Turian et al. [2010]. Moreover, Mikolov et al. [2013a] provide evidence that learning high-quality word vectors from a 1.6 billion-word data set takes CBOW less than a day.

FastText

Word2vec has an important drawback – its vocabulary is fixed. Moreover, many languages contain rare word-forms that are often underrepresented in datasets. This problem is called word sparsity and affects word2vec representations negatively, too. These issues were addressed by Bojanowski et al. [2017]. Their algorithm, called fastText, is basically an extension of skip-gram with negative sampling described in Mikolov et al. [2013c]. The authors argue that the internal structure of words contains a lot of useful information, and in order to account for that they propose learning n-grams representations. In this setting, each word is represented by a set of character n-grams. For example, with $n = 3$ the word *apple* will become {*<ap, app, ppl, ple, le>*, *<apple>*}, where *<>* are special symbols that are used to distinguish prefixes and suffixes from whole words (in this example *app* $\neq$ *<app>*). Note, that the word itself is included in the set, too. In practice, for each word, all n-grams of lengths from 3 to 6 are extracted. Furthermore, the similarity between two word representations can no longer be expressed by a simple dot-product. Instead, the new measure is proposed:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} z_g^\top v_c, \tag{2.13}$$

where (w, c) – word-context pair, $\mathcal{G}_w$ – set of all w's n-grams, z_g – vector representation of an n-gram g, v_c – vector representation of a context word. Thus, both whole words' and n-grams' representations are learned during training. This method allows building semantic similarity between words through shared n-grams. Moreover, out-of-vocabulary (OOV) words can be represented as an average of n-grams.

The authors report significant improvements over baselines that do not take into account sub-word information. Overall, the method is simple, fast and does

not require any pre-processing. Moreover, fastText has gained a lot of popularity because the open-source library was released among pre-trained embeddings for 157 languages[2].

2.4.3 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 [Bansal et al., 2014], and in the medical domain [Chowdhury et al., 2019].

Coates and Bollegala [2018] show that simple vector combining approaches, such as concatenation or averaging, can significantly improve the overall performance for several tasks. For instance, Singular Value Decomposition (SVD) demonstrates good results with the ability to control the final dimension of vectors [Yin and Schütze, 2016]. Autoencoders [Bollegala and Bao, 2018] further promote the idea of creating meta-embeddings. The authors propose several algorithms for combining various word vectors into one vector by encoding initial vectors into some meta-embedding space and then decoding backwards.

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

For any AAEME approach, they experiment with different loss functions: Mean Squared Error loss, Kullback–Leibler divergence loss, cosine distance loss and also their combinations. In Neill and Bollegala [2018] the authors investigated the performance of the autoencoders depending on the loss function. They discover that there is no evident winner across tasks and that different loss functions are defined by different tasks.

Meta-embeddings are already used in such machine learning tasks as dependency

[2]https://fasttext.cc/

parsing [Bansal et al., 2014], classification in healthcare [Chowdhury et al., 2019], named-entity recognition [Winata et al., 2019, Neill and Bollegala, 2018], sentiment analysis [Neill and Bollegala, 2018], word similarity and analogy tasks [Coates and Bollegala, 2018, Bollegala and Bao, 2018, Yin and Schütze, 2016]. To the best of our knowledge, meta-embeddings have not been applied to the Taxonomy Enrichment task, especially for the fusion of texts and graph embeddings.

2.4.4 Graph-based Representations for Taxonomies

Taxonomies can be represented as graphs and there exist various approaches to learning graph-based representations. They are thoroughly compared in Makarov et al. [2021] on the link prediction task, which is closely related to Taxonomy Enrichment. The paper also demonstrates that the combination of text and graph embeddings gives a boost to the link prediction task. Most of those methods listed in Makarov et al. [2021] have also been tested on tasks related to Taxonomy Enrichment.

For instance, node2vec embeddings [Grover and Leskovec, 2016] are used for Taxonomy Induction among other network embeddings [Liu et al., 2018]. In Aly et al. [2019], the authors perform the same task. They use hyperbolic Poincaré embeddings to enhance automatically created taxonomies. The SemEval-2016 subtask of reattaching query words to the taxonomy is quite similar to Taxonomy Enrichment which we perform. However, the datasets of the SemEval-2016 Task 13 are restricted to specific domains, which leaves an open question of the efficiency of Poincaré embeddings for the general domain and larger datasets. Moreover, Aly et al. [2019] use Hearst Patterns to discover hyponym-hypernym relationships. This technique operates on words, and cannot be transferred to word-synset relations without extra manipulation.

As for the Knowledge Graph Construction task, which is a more general task in relation to the Taxonomy Induction, the vast majority of approaches also use word embeddings as node representations. Several approaches like Luan et al. [2018] and apply ELMO embeddings [Peters et al., 2018] to predict entities and their relations to the Knowledge Graph. Other approaches [Han et al., 2020] utilize a combination

of ELMO, Poincaré and node2vec embeddings to enhance the Knowledge Graph built upon Wikipedia.

Graph convolutional networks (GCNs) [Kipf and Welling, 2017] as well as graph autoencoders [Kipf and Welling, 2016] are mostly applied to the link prediction task on large knowledge bases. For example, in Rossi et al. [2020] 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 [Pujary et al., 2020]. As for the Taxonomy Enrichment task, we are only aware of a recent approach TaxoExpan [Shen et al., 2020] which applies position-enhanced graph neural networks (GCN [Kipf and Welling, 2017] and GAT [Velickovic et al., 2018]) that we also evaluate on our datasets[3].

Graph representation learning methods can be divided into two categories: non-inductive and inductive. Non-inductive (or transductive) approaches require the presence of all nodes in the graph during training. Inductive frameworks allow to generalize to work with the new input data. In the following paragraphs we briefly outline the main ideas of each graph-based representation model used in our research.

Non-inductive Node Representation Models

In this section we review six non-inductive node representation models used in the thesis: DeepWalk [Perozzi et al., 2014], TADW [Yang et al., 2015], node2vec [Grover and Leskovec, 2016], Poincaré embeddings [Nickel and Kiela, 2017], GNN [Scarselli et al., 2009], GCN [Kipf and Welling, 2017] and HOPE [Ou et al., 2016]. The first three methods explore the same idea of using random walks for learning node representation. Poincaré method, unlike all others, models tree data in hyperbolic space. GNN and its extension, GCN, are applying deep learning framework for graph embedding learning. There exist many other graph-based vector representation, e.g., representations of graphs [Ivanov and Burnaev, 2018, Narayanan et al., 2017] or representations of edges [Wang et al., 2020, Gao et al., 2019], however, they lay out of scope of the thesis.

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

Perozzi et al. [2014] are the first to apply Skip-Gram method towards learning graph node representations. Their method, DeepWalk, uses truncated random walks to acquire information about the local neighbourhood. This information is then utilized to learn latent representations by treating walks as the equivalent of sentences. Given an undirected graph, a random walk is a stochastic process that starts at a vertex u and then randomly chooses which next vertex from u's neighbourhood to visit. Given a graph $G = (V, E)$, where V – set of all vertices and E – set of all edges, let $N_S(u) \subset V$ be a neighborhood of node u generated with a neighborhood sampling strategy S. $N_S(u)$ can also be viewed as a multiset of nodes (that can occur several times) that were visited during random walks that started at u. Then the embedding optimization is performed with the goal of maximization of the likelihood of random walk co-occurrences. Thus, the loss for embedding $\mathbf{z_u}$ of node u and embedding $\mathbf{z_v}$ of node v in $N_S(u)$ is:

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_S(u)} - \log(\mathrm{Pr}(\mathrm{v}|\mathrm{z_u})), \tag{2.14}$$

where $Pr(v|z_u)$ is parameterized with softmax:

$$Pr(v|z_u) = \frac{\exp(\mathrm{z_u^\intercal z_v})}{\exp(\sum_{\mathrm{n} \in \mathrm{V}} \mathrm{z_u^\intercal z_n})}, \tag{2.15}$$

which results in updated Equation 2.14:

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_S(u)} - \log(\frac{\exp(\mathrm{z_u^\intercal z_v})}{\exp(\sum_{\mathrm{n} \in \mathrm{V}} \mathrm{z_u^\intercal z_n})}). \tag{2.16}$$

Optimization of Equation 2.16 is computationally expensive and gives $O(|V^2|)$ complexity. Same as in word2vec Mikolov et al. [2013a], DeepWalk uses negative sampling on order to speed up computation. The new formulation implies normalizing only against k random "negative samples" instead of all vertices in a graph.

[Yang et al., 2015] provides a matrix-factorization view of DeepWalk which inspires them to propose a Text-Associated Deep Walk (TADW) algorithm. Denote a sequence of nodes generated by random walks as $S = \{v_1, v_2, \ldots, v_{|S|}\}$. Then, a

context for a central node v_i with window-size t are nodes $v \in \{v_{i-t}, \ldots, v_{i+t}\} \setminus v_i$. Let r_{v_i} and c_{v_j} be representation vectors of the center node v_i and its context node v_j. Thus, each node gets two representations: r_v when the node is central and c_v when the node is inside a context. Suppose for a graph $G = (V, C)$ random walks produce a vertex-context set D, that consists of pairs (v, c). Denote V_C as the set of context vertices. DeepWalk generates an embedding vector $r_v \in \mathbb{R}^k$ for a node v. Similarly, for a context node $v \in V_C$ a context embedding is generated: $c_v \in \mathbb{R}^k$. In the majority of cases $V = V_C$. Authors of Yang et al. [2015] prove that DeepWalk performs following factorization:

$$M = W^T H, \tag{2.17}$$

where ith column of matrix $W \in \mathbb{R}^{k \times |V|}$ is vector r_{v_i}, and jth column of matrix $H \in \mathbb{R}^{k \times |V_C|}$ is vector c_{v_j}. Each entry in matrix $M \in \mathbb{R}^{|V_C| \times |V_C|}$ can be viewed as a logarithm of the average probability that random walk from node i arrives at a node j in t steps. Furthermore, the proposed update has a following form:

$$M = W^T H T, \tag{2.18}$$

where $T \in \mathbb{R}^{f_t \times |V|}$ – text feature matrix, and matrix H has changed dimensions to $\mathbb{R}^{k \times f_t}$. Thus, the new factorization form allows fusion of textual information into the learning of node representations. The authors provide empirical evidence that their method outperforms both classic DeepWalk as well as direct concatenation of DeepWalk generated network representations with text features.

The node2vec algorithm was suggested by Grover and Leskovec [2016]. It can be viewed as an update of DeepWalk with different and less constrained random walk sampling for neighbourhood generation. The notion of network neighbourhood in node2vec is flexible and is defined as a combination of two sampling strategies:

- Breadth-first Sampling (BFS): neighbourhood consists only of immediate neighbours of the source node;
- Depth-first Sampling (DFS): neighbourhood consists of a sequential chain of

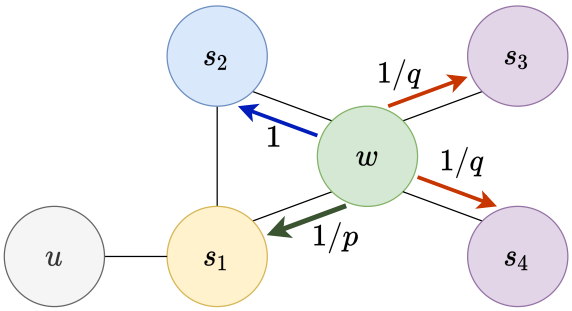


Figure 2-5: Illustration of random walk in node2vec.

neighbours that is spread as far as possible from the source node along the graph.

The neighbourhoods are generated using fixed-length biased random walks along the graph. They are parameterized by:

- Return parameter p, that controls likelihood of instantly returning back to the previous node;
- In-out parameter q, that can be seen as a "ratio" between BFS and DFS and controls moving inwards or outwards.

For example, refer to Figure 2-5. A random walk has transitioned from the node s_1 to the node w. From there it can go back to s_1 with unnormalized probability $1/p$, go to s_2 and stay at the same distance from s_1, or go to s_3 or s_4 with unnormalized probabilities $1/q$. The lower value of p makes the strategy more BFS-like, the lower value of q – more DFS-like. Thus, the overall steps performed by node2vec algorithm are:

1. Pre-compute all the transition probabilities;
2. For each node u simulate r length l random walks;
3. Optimize the objective with Stochastic Gradient Descent.

The ability to model complex data of embeddings in Euclidean space is bounded by their dimensionality. Thus, for large graph-structured data, representation capacity must be increased. This issue is addressed in Nickel and Kiela [2017]. They focus

on large graph-structured datasets with hierarchical structure. Authors propose computing embeddings in hyperbolic space, specifically using the Poincaré ball model. The hyperbolic space naturally fits for representation of a tree (which can even be though of as a "discreet hyperbolic space"). Leaves that are the furthest from the root (l levels below it) are placed on a surface of a sphere in hyperbolic space with radius proportional to the number of levels, while everything that is less than l levels under the root is placed within the sphere and the root itself – in the center. The distance from the center to nodes at different levels expands exponentially, and as a result, in hyperbolic space, we can fit an arbitrary number of levels with substantially fewer dimensions than in Euclidean space. The optimized loss function is

$$\mathcal{L}(\Theta) = \sum_{(u,v)\in\mathcal{D}} \log \frac{e^{-d(\boldsymbol{u},\boldsymbol{v})}}{\sum_{\boldsymbol{v}'\in\mathcal{N}(u)} e^{-d(\boldsymbol{u},\boldsymbol{v}')}}, \tag{2.19}$$

where Θ – set of embeddings, u and v – two nodes, $\mathcal{D}$ – set of observed hierarchical relations (edges), $\mathcal{N}(u) = \{v \mid (u,v) \notin \mathcal{D}\} \cup \{u\}$ – set of negative examples (random nodes that do not have relationships with u). Also, $d(\boldsymbol{u},\boldsymbol{v})$ is a Poincaré distance defined as:

$$d(\boldsymbol{u},\boldsymbol{v}) = \mid arcosh\left(1 + 2\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{(1-\|\boldsymbol{u}\|^2)\,(1-\|\boldsymbol{v}\|^2)}\right). \tag{2.20}$$

The Equation 2.19 is optimized via Riemann Gradient Descent. In Nickel and Kiela [2017] empirical evidence was provided that Poincaré embeddings outperform Euclidean embeddings at reconstruction and link prediction on network data. The method has shown good results for hyponymy-hypernymy prediction on WordNet.

From a deep learning perspective, all methods described above are considered "shallow". There are no hidden layers and the output vectors are optimized directly. Furthermore, different class of graph representation learning approaches has emerged – Graph Neural Network (GNN). In recent years, many variations of GNNs have been developed, one of them – Graph Convolutional Network (GCN) Kipf and Welling [2017]. The GCN takes two matrices as an input:

- X – $N \times D$ feature description matrix (N – number of nodes in a graph, D – size of input feature vectors);

- A – graph adjacency matrix.

As an output GCN produces a $N \times F$ feature description matrix Z (F – size of output feature vectors). Furthermore, every layer (total number – L) of neural network can be represented as one of following:

$$\begin{aligned} H^0 &= X, \\ H^{l+1} &= f(H^l, A), \\ H^L &= Z. \end{aligned} \tag{2.21}$$

Kipf and Welling [2017] propose the following layer-wise propagation rule:

$$f\left(H^l, A\right) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^l W^l\right), \tag{2.22}$$

where σ – non-linear activation function (e.g., ReLU), $\hat{A} = A + I$ (I – identity matrix), $\hat{D}$ – diagonal node degree matrix of $\hat{A}$, W^l – l-th neural network layer weight matrix. The addition of an identity matrix to the adjacency matrix is done to sum a feature vector of a node with feature vectors of its neighbourhood during multiplication, essentially enforcing self-loops in the graph. The $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ part represents a symmetric normalization of $\hat{A}$.

High-Order Proximity preserved Embeddings (HOPE) [Ou et al., 2016] is yet another approach that embeds a graph into a vector space preserving information about graph properties and structure. Unfortunately, most structures cannot preserve asymmetric transitivity, which is a critical property of directed graphs. To solve the problem, the authors employ matrix factorization to directly reconstruct asymmetric distance measures like Katz index, Adamic-Adar or common neighbors. This approach is scalable — it preserves high-order proximities of large-scale graphs and is capable of capturing asymmetric transitivity. HOPE outperforms state-of-the-art algorithms in tasks of reconstruction, link prediction and vertex recommendation. As for our Taxonomy Enrichment task, we also apply HOPE to generate graph embeddings to be used as input in the DWRank model. The main difference with the other embeddings is that HOPE does not incorporate textual information from the nodes.

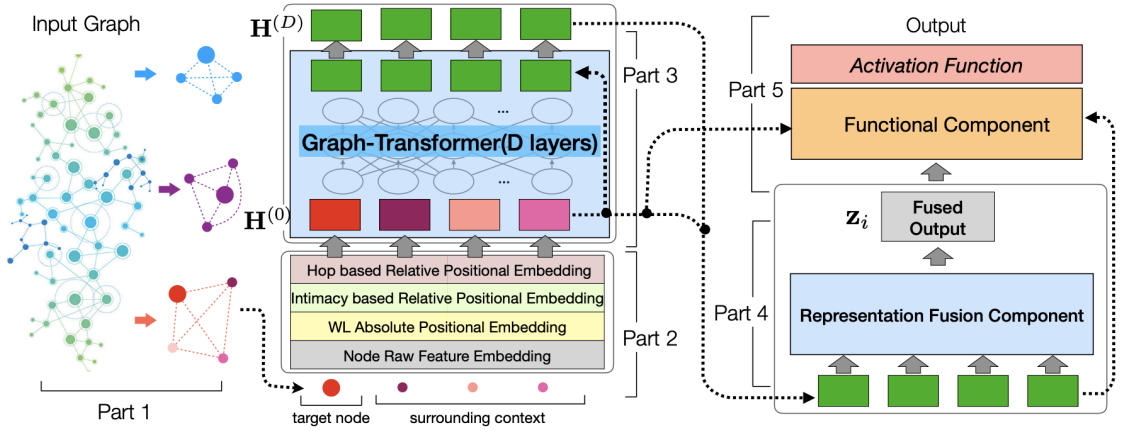


Figure 2-6: Graph-BERT model architecture (image source: Zhang et al. [2020]).

Inductive Node Representation Models

In this subsection we are going to review two inductive node representation models: Graph Attention Network (GAT) Velickovic et al. [2018] and Graph-based BERT (Graph-BERT) Zhang et al. [2020]. They all are based on deep learning approaches, and each consecutive method can be seen as an evolution of the previous one.

Inspired by Vaswani et al. [2017], a further advancement of GNN emerged – Graph Attention Network (GAT) Velickovic et al. [2018]. The advantages of this new approach are: computational efficiency due to parallelization across node-neighbour pairs, possibility of application to graph nodes with different degrees and generalization towards unseen nodes. An input a GAT layer is a set of N graph node features $\mathbf{h} = \left\{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\right\}, \vec{h}_i \in \mathbb{R}^F$, where F is a size of each node's feature vector. The layer outputs a set of new feature vectors $\mathbf{h}' = \left\{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\right\}, \vec{h}'_i \in \mathbb{R}^{F'}$, where F' indicates a potential change of cardinality. Similar to the attention mechanism, an alignment model e_{ij} is computed. However, GATs use masked self-attention, which means that the coefficients are computed between a pair of feature vectors from the same layer and the vectors have to represent first-order neighbouring nodes in a graph. With such mechanism, each edge is being assigned a different importance attention coefficient.

Furthermore, recent advancement of language representation models motivated other architectures that leverage same power. One of the most popular language representation models, Bidirectional Encoder Representations from Transformers

(BERT) Devlin et al. [2019]. Utilization of context and attention mechanism in transformers led Zhang et al. [2020] to introduction of a new GNN model – Graph-based BERT (Graph-BERT) that is based on the attention mechanism without any graph convolution or aggregation operators. The Figure 2-6 shows the architecture of Graph-BERT. The original graph is sampled into linkless subgraphs that represent nodes' local neighbourhoods. Each subgraph in a batch consists of a target node and its learning context. Using the extended graph-transformer layers Graph-BERT can effectively learn the representations of the target node. The subgraphs are sampled using top-k intimacy sampling approach, which is based on the graph intimacy matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. In Zhang et al. [2020] matrix $\mathbf{S}$ is defined based on pagerank algorithm Page et al. [1998] and is defined as such:

$$\mathbf{S} = \alpha(\mathbf{I} - (1-\alpha)\overline{\mathbf{A}})^{-1}, \tag{2.23}$$

where $\alpha \in [0, 1]$ – factor that is usually set at 0.15, $\overline{A} = AD^{-1}$ is a column-normalized adjacency matrix, A – input graph adjacency matrix, and D – its corresponding diagonal matrix (where $D(i, i) = \sum_j A(i, j)$). Furthermore, four types of feature vector embeddings are computed: (1) raw feature vector embeddings, (2) Weisfeiler-Lehman (WL) Niepert et al. [2016] absolute role embeddings, (3) intimacy based relative positional embeddings and (4) hop based relative distance embeddings. The first type embeds node's raw feature vector into a shared vector space with its raw features (images, texts, etc.). The second type embeddings can capture the global node role information in the representations. The third, extracts the local information in the subgraph. The last type of embedding is somewhat between the absolute role embedding (that capture global information) and intimacy based relative positional embedding (that capture local information). Afterward, all embeddings are aggregated and passed to the transformer-based encoder.

Graph-BERT can be pre-trained with two tasks: node attribute reconstruction, and graph structure recovery. In our work we explore both options and compare them via intrinsic evaluation (see 7.4.1). The node attribute reconstruction task aims at capturing the node attribute information in the learned representations. The

graph structure recovery task focuses more on the graph connection information.

2.5 Conclusion

In this section we have reviewed the evolution of Taxonomy Enrichment task from its first formulation [Jurgens and Pilehvar, 2016] to the recent research [Shen et al., 2020, Cho et al., 2020]. It is clear that the trend has shifted from sole application of distributional word embeddings to incorporation of graph representations. The intuition behind this shift is clear, as the WordNet taxonomy itself can be viewed as a graph which structure might become a source of valuable information. Moreover, it is noticeable that an increasing effort is being put towards leveraging deep language models' linguistic capabilities and knowledge of lexical semantic relations in a context of hypo-hypernym relationships.

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.

After the discussion of the current status of the field, we move on to the dataset creation, duscussed in Chapter 3.

Chapter 3

Diachronichal Dataset for Evaluation

This chapter presents the datasets generated from the existing versions of the English WordNet and RuWordNet for the Russian language. It should be mentioned that query words for the RUSSE'2020 test set were selected manually from a larger list of new words to be added to the current version of WordNet.

The important part of our study is the observation that one can learn from the history of the development of lexical resources through time. More specifically, we make use of the various historical snapshots (versions) of WordNet lexical graphs and set up a task for their automatic completion assuming the manual update of the ground truth. This diachronic analysis – similar to diachronic lexical analysis of word meanings – is used to build two datasets in our study: one for English and another one for Russian, based respectively on Princeton WordNet [Miller, 1995] and RuWordNet taxonomies. It is important to mention that by using the word "diachronic" we do not imply lexical diachrony, e.g., semantic shifts of words over time [Schlechtweg et al., 2020], 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 3.1.

3.1 English Dataset

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

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

Table 3.1: Statistics of two diachronic WordNet datasets used in this study.

include them in order to make the evaluation less restricted.

To compile the 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 more sparse than those for nouns and verbs. Therefore, it makes the task more difficult.

In order to find the most suitable pairs of releases, we compute WordNet statistics (see Table 3.2). New words demonstrate the difference between the current and the previous WordNet version. For example, it shows that the dataset generated by "subtraction" of WordNet 2.1 from WordNet 3.0 would be too small, they differ by 678 nouns and 33 verbs. Therefore, we create several datasets by skipping one or more WordNet versions. The statistics for the datasets we selected for our study are provided in Table 3.1. In order to understand the relationship between the datasets and the wordnets, you may refer to Figure 3-1.

Taxonomy	Synsets		Lemmas		New words	
	Nouns	Verbs	Nouns	Verbs	Nouns	Verbs
WordNet 1.6	66 025	12 127	94 474	10 319	-	-
WordNet 1.7	75 804	13 214	109 195	11 088	11 551	401
WordNet 2.0	79 689	13 508	114 648	11 306	4 036	182
WordNet 2.1	81 426	13 650	117 097	11 488	2 023	158
WordNet 3.0	82 115	13 767	117 798	11 529	678	33

Table 3.2: Statistics of the English WordNet taxonomies used in this study.

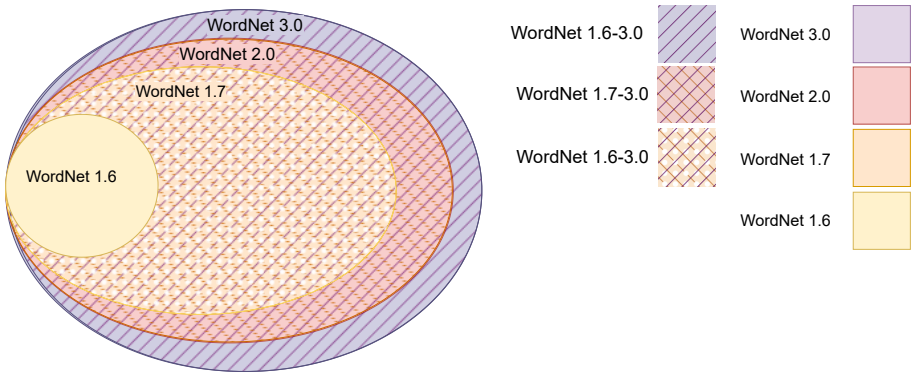


Figure 3-1: The display of relationships between different WordNet versions and the created datasets. WordNet-2.1 is not depicted, as it is not used for the dataset compilation.

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

Table 3.3: Examples of Russian nouns with translation mapped to English WordNet.

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 observations, the task of automatically identifying the exact hypernym might be too challenging and finding the "region" where a word belongs ("parents" and "grandparents") can already be considered a success.

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

All datasets[1] created for this research and the code[2] for their construction are publicly available.

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

3.2 Russian Datasets

In this section we show how the process of the language-independent dataset construction can be transferred to other wordnets or taxonomies with timestamped releases for Russian.

In order to create an analogous version of the English dataset for Russian, we use the RuWordNet taxonomy [Loukachevitch et al., 2016]. RuWordNet comprises *synsets* — sets of synonyms expressing a particular concept. A synset consists of one or more *senses* — words or multi-word constructions in the initial form. Therefore, we use the current version of RuWordNet-1.0 and the extended version of RuWordNet — RuWordNet-2.0 which has not been published yet to compile the dataset (see Table 3.1). We select words which appear only in a newer version and consider their hypernyms as gold standard hypernyms. Likewise in English dataset construction pipeline, we add "orphans" to the dataset, if only their hypernyms appear in both versions. As a result, the dataset comprises 14.660 nouns and 2.154 verbs, which is comparable to the WordNet1.6-3.0 dataset size. The relationship between the Russian datasets and the RuWordNets version are displayed in Figure 3-2.

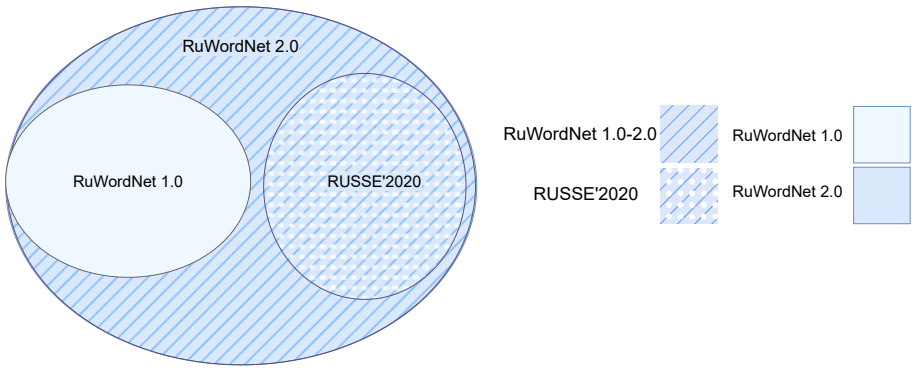


Figure 3-2: The display of relationships between different RuWordNet versions, RUSSE'2020 and the created dataset

RUSSE'2020 Dataset

The RUSSE'2020 dataset was created for the Dialogue Evaluation shared task [**Nikishina** et al., 2020a] and can be viewed as a restricted subset of the Russian dataset we present above. We split this data into two parts: public test set (763

nouns and 175 verbs) and private test set (1 525 nouns and 350 verbs).

The words included in the RUSSE'2020 dataset are collected in the following way. Given words (nouns and verbs) from the dataset RuWordNet1.0-2.0, we select only single tokens. Then we filter the obtained list, filtering the following words:

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

If one of the synsets selected for RUSSE'2020 from RuWordNet-2.0 belongs to hypernym which is also not yet presented in RuWordNet-1.0, we set its closest published "ancestor" as a gold hypernym.

We also create a training data set for the participants of the RUSSE'2020 competition shared task using RuWordNet1.0 data. The goal is to keep the published RuWordNet-1.0 taxonomy in the dataset format annotated analogously to the test data. To create the training set, we sample all leaves (synsets with no hyponyms) of depth equal to or more than 5. Overall, it comprises 12 393 nouns and 2 102 verbs.

3.2.1 WordNet ILI Mapping (ru-en)

In order to connect wordnets in different languages, the ILI is used [Bond et al., 2016]. This mapping is designed to make possible coordination between wordnet projects. For the Russian test sets, we also provide a mapping from RuWordNet to WordNet[3]. For each hypernym synset of each query word, we present the corresponding WordNet synset index. Table 3.3 demonstrates several examples of this kind of mapping. As

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

we can see, datasets for the Russian nouns and verbs are extended with an additional column called "WordNet synsets", where the corresponding WordNet3.0 synsets are listed in accordance with the Russian synset list.

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

Table 3.4: Coverage of the WordNet synsets for the hypernyms in the Russian test sets.

However, not all synsets have an equivalent in the mapping language, as there exist untranslatable concepts and lacunae. Therefore, we present in Table 3.4 the coverage of the WordNet synsets for the hypernyms of query words from the test set. This mapping can further be used for multilingual experiments.

3.3 Evaluation Metrics

This section presents the methodology for the usage of corpora presented in the previous chapter for the Taxonomy Enrichment experiments. We outline metrics that are normally used for the Taxonomy Enrichment task and explain our choice of the score for evaluation. Moreover, we discuss possible modifications of the existing metrics to increase their applicability to our task setting. More specifically, we

accept as correct answer hypernyms of the direct hypernyms in addition to the direct hypernym. In some cases, as we see later, it might be extremely challenging and unreal to point at a specific place in the taxonomy. Therefore, as part of our contribution, we suggest a new annotation metric for the facilitated Taxonomy Enrichment task setting.

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

$$MAP = \frac{1}{N}\sum_{i=1}^{N} AP_i,$$
$$AP_i = \frac{1}{M}\sum_{i}^{n} prec_i \times I[y_i = 1], \tag{3.1}$$

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 [Camacho-Collados et al., 2018]. The MAP score takes into account the whole range of possible hypernyms and their rank in the candidate list.

3.4 Evaluation Setup

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

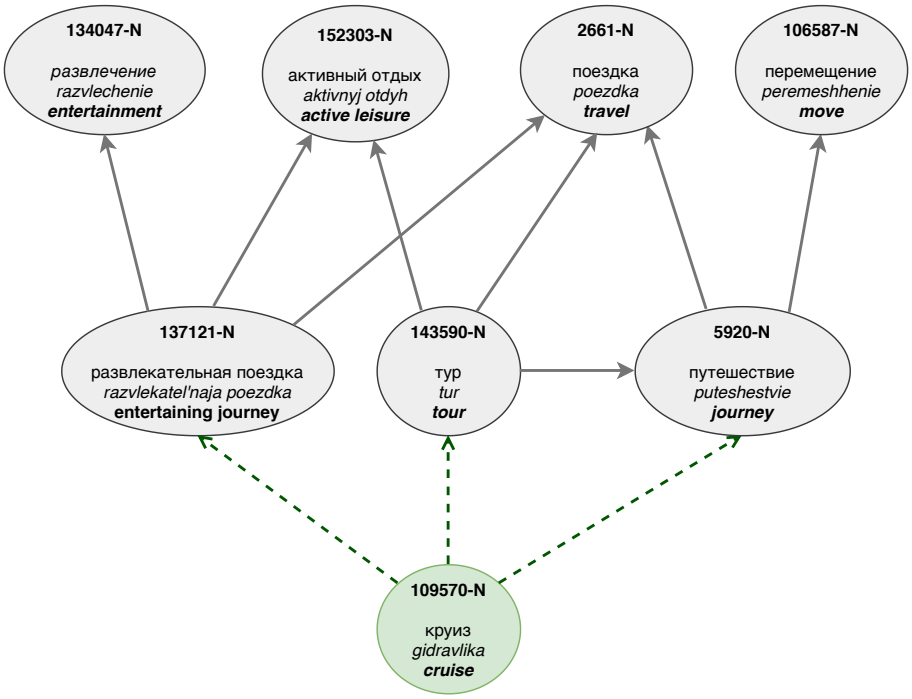


Figure 3-3: The example from RuWordNet-1.0 for case 1. Multiple related hypernyms are needed to reflect all shades of the meaning.

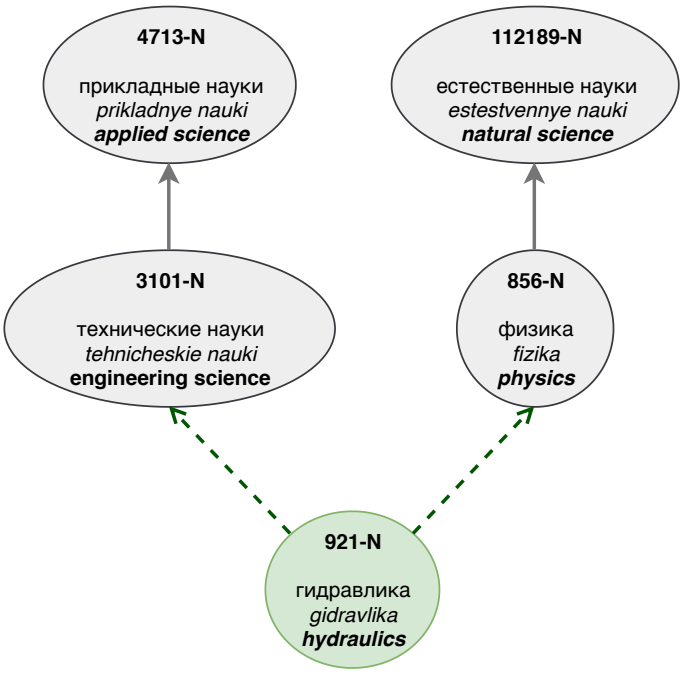


Figure 3-4: The example from RuWordNet-1.0 for case 2a. Word is a composition of senses

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 case 2a corresponds to a monosemous word and case 2b indicates polysemy, this difference does not affect the evaluation process. We propose that in both 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, 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 case 1 we will have a single connected component, as can be seen in Figure 3-3. To get the maximum MAP score, it is enough to guess *any* hypernym from this connection. In cases 2a and 2b we will have multiple components, and a model should guess *any* hypernym from *each* of the components, see Figures 3-4 and 1-1

correspondingly.

3.5 Conclusion

All in all, we created new diachronic datasets based on WordNet versioning 1.6-3.0. They feature:

- 11.551 noun synsets and 401 verb synsets from the WordNets 1.6-3.0,
- 4.036 noun synsets and 182 verb synsets from the WordNets 1.7-3.0,
- 2.023 noun synsets and 158 verb synsets from the WordNets 2.0-3.0.

We also compiled a new dataset based on RuWordNet 1.0-2.0 and mapped it to the English WordNet synsets using the ILI mapping. We described the evaluation metric and introduce its modification, regarding the complexity of the task. In the next chapter we discuss the existing methods for the Taxonomy Enrichment task that we test on the constructed datasets.

Chapter 4

Competing Systems

In this chapter, we first describe our baseline model which is a method of synset ranking based on distributional embeddings and hand-crafted features (the method was proposed as a baseline for the RUSSE-2020 shared task [**Nikishina** et al., 2020b]). Then, we propose extending it with new features extracted from Wiktionary and using alternative sources of information about words (e.g., graph representations) and their combinations.

4.1 RUSSE'2020 Participants

One of the contributions of the current thesis is also the organisation of the RUSSE-2020 competition on the Taxonomy Enrichment for the Russian language. The task formulation matches the one presented in Section 1.1: given words that are not yet included in the taxonomy, we need to associate each word with the appropriate hypernym synset(s) from the existing taxonomy RuWordNet. For example, given an input word "**утка**" (duck) the participants are asked to provide a ranked list of its most probable 10 candidate hypernym synsets, e.g., "**животное**" (animal), "**птица**" (bird), and so on. We assume that an *orphan* may be a "child" of one, two, or more "ancestors" (hypernym synsets) at the same time.

The task featured two tracks: detection of hypernym synsets for nouns and verbs. We provided participants with the following resources: (i) a training set based on the RuWordNet taxonomy, (ii) a collection of news texts from the year 2017 (2.2 billion

tokens), (iii) a parsed Wikipedia corpus[1], and (iv) a hypernym database from the Russian Distributional Thesaurus[2] [Panchenko et al., 2016, Sabirova and Lukanin, 2014], which contains a set of hypernyms and a set of distributionally related terms both extracted from a huge text corpus. The participants were allowed to use any additional data and were asked to indicate the additional resources in their model descriptions.

The competition was hosted on the Codalab platform[3]. To allow the participants to evaluate their models on real data, we split the gold standard data into public and private test sets (denoted as "PRACTICE" and "EVALUATION" phases in Codalab). Thus, the participants could test their models before the deadline on the public test set by submitting the results to the "PRACTICE" leaderboard. During the "EVALUATION" phase the leaderboard was hidden, so the participants were not able to overfit the test data.

The participated systems mainly relied on vector representations of words and the intuition that words used in similar contexts have close meanings. We describe their approaches in the following paragraphs.

Yuriy The participant generated candidate hypernyms and calculated features for them. Then candidates were ranked by a linear model with handcrafted weights. The list of features is provided below:

1. candidate is in the top-10 similar words from RuWordNet;
2. candidate is in hypernyms of top 10 similar words from RuWordNet;
3. candidate is in second-order hypernyms of top 10 similar words from RuWordNet;
4. candidate is in hypernyms on Wiktionary[4] page about the word;
5. candidate is in hypernyms of hypernyms on Wiktionary page about the word;

[1] https://doi.org/10.5281/zenodo.3827903
[2] https://doi.org/10.5281/zenodo.3827834
[3] https://competitions.codalab.org/competitions/22168
[4] https://ru.wiktionary.org

6. candidate is in "en-ru" translation of WordNet[Miller, 1998a] hypernyms of "ru-en" translation of the word (extracted with Yandex Machine Translation model[5]);

7. candidate is in the word definition on the Wiktionary page;

8. candidate is on the Yandex search result page;

9. candidate is on the Google search result page.

The candidates were collected using features 1-6. Features 1-3 are based on the fastText model[6]. Features 7-9 are used with all other features to rank candidates. This approach was applied for both "nouns" and "verbs" tracks.

xeno This participant merged candidates extracted by several methods. Those methods included: Russian Wiktionary semantic graph (taxonomic relations, synonymy, antonymy); rule-based plain text definition parsing; rule-based plain text parsing with Hearst patterns on Russian Wikipedia from Panchenko et al. [2016] and Russian language corpus; graph-based analysis of the nearest neighbor list obtained from word2vec. The definitions were taken from Russian Wiktionary, Russian Wikipedia, Big English-Russian polytechnic dictionary and Efremova dictionary [Efremova, 2000]. The above-mentioned methods were used for nouns. For verbs, the team used only the Russian Wiktionary semantic graph and rule-based plain text definition parsing.

KuKuPl [Kunilovskaya et al., 2020] This team trained a classifier on the official train data provided by the organizers. They considered synsets (occurring more than n times in the training data) as classes, representing words with the embeddings (standard CBOW from word2vec) pretrained on a concatenation of four corpora: Araneum Russicum Maximum, Russian Wikipedia, Russian National Corpus, and a corpus of Russian news (9.5 billion word tokens overall). The corpus was specifically tailored for this task: all multi-word entities which also occurred in the RuWordNet

[5] https://translate.yandex.ru
[6] https://fasttext.cc/docs/en/crawl-vectors.html

were merged into single tokens, thus making sure that the majority of the RuWordNet entries received their respective vector representations.

A neural network classifier with one hidden layer (size 386), dropout of 0.1, ReLu activation, and a softmax output layer was trained on all the training data until convergence, using hypernym synset ids as class labels. At test time, the trained model obtains the vector representation of a query word and predicts possible classes (hypernym synsets) for this vector. 10 synsets with the highest probability are considered predictions. This approach is applied to both "nouns" and "verbs" tracks.

RefalMachine, Parkat13 [Tikhomirov et al., 2020] This team implemented the algorithm consisting of three stages. Firstly, they created a list of similar words using a combination of vector representations of words obtained with Parkinson's Progression Markers Initiative (PPMI) weighting and SVD factorization (window = 1). Secondly, they selected candidates from those similar words (depending on pattern matching), their hypernyms, and second-order hypernyms. These candidates were ranked based on the following features:

- cosine similarity;
- patterns matching co-hyponyms;
- patterns matching hypernyms (Hearst patterns). The patterns were extracted from the news corpus provided by the organizers;
- the number of synset occurrences in the candidate list;
- probabilities based on ruBERT predictions [Kuratov and Arkhipov, 2019].

The final rank for each candidate was computed using the weighted feature combination; the weights are hand-picked during the experiments. This approach was applied to both "nouns" and "verbs" tracks.

MorphoBabushka (alvadia, maxfed, joystick) [Arefyev et al., 2020] This team used the following pipeline. First, they retrieved the nearest neighbors for the target word from word2vec "SkipGram with Negative Sampling" model trained on Librusec

book collection [Arefyev et al., 2015] and searched for their direct and indirect hypernyms in RuWordNet. Then they counted direct and indirect hypernyms of the nearest neighbors, combining their counts, converting words with the wrong POS to the correct form if possible (otherwise excluding). They took the 10 most frequent hypernyms of the nearest neighbors' synsets. Finally, they combined those hypernyms with the hypernyms extracted from Wiktionary by matching the definition N-grams with the synsets. This method was applied to both "nouns" and "verbs" tracks.

cointegrated [Dale, 2020] The participant used similarity scores between word embeddings to predict hypernym relations. For each RuWordNet synset, the team computed the embedding of its title, all senses, and the mean embedding of the title and all senses. Each type of the above-mentioned embeddings was computed as an L2-normalized weighted mean of its word embeddings from RusVectores [Kutuzov and Kuzmenko, 2017] (weight is 1.0 for nouns, 0.1 for prepositions, and 0.5 for all other POS). For OOV words, the embedding was computed as a mean embedding of all words in the vocabulary with the longest prefix matching the target word.

For each query word (orphan), the participant found its 100 nearest neighbors from RuWordNet and all the first and second-order hypernyms of the corresponding synsets, considering them as answer candidates. The resulting list of hypernyms comprises 10 candidates with the highest scores. The score for each candidate is a sum of "neighbor scores" overall nearest neighbors from RuWordNet; if the candidate is a second-order hypernym, its "neighbor score" is multiplied by 0.5. The "neighbor score" is calculated as $\exp(-3 \cdot d) \cdot s^5$, where d is the distance between the queries and neighbor embeddings; s is their cosine similarity. The described approach was applied to both "nouns" and "verbs" tracks.

On one hand, our methods in the presented thesis have pretty much things in common, such as using cosine similarity for comparing word embeddings and using additional resources, such as Wiktionary. On the other hand, we work out methods which depend on graph-based structures and combine them with the approaches applying word embeddings.

4.2 Web-based Synset Ranking (WBSR)

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

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

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

Similarly to the approaches described above, here we also make use of Wiktionary and fastText embeddings cosine similarity. However, we treat synsets and words/phrases that they consist of in a different way. In the previously described approaches, we computed embeddings for multiword phrases by averaging word embeddings of individual words in them. Here we treat them as sentences — we

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

compute their embeddings using the `get_sentence_vector` method from the fastText Python library. FastText vectors are divided by their norms and then averaged so that only vectors with the positive L_2-norm value are considered. Secondly, we do not combine the word/phrase vectors into a synset vector but operate with the word/phrase embeddings directly.

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

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

Then, we rank all candidates by a Logistic Regression model which uses the following features:

- the candidate synset contains a word/phrase from the list of query word's nearest neighbours;
- the candidate synset is a hypernym of one of the nearest neighbours;
- the candidate is a second-order hypernym of one of the nearest neighbours;
- the candidate synset contains a hypernym of the query word from Wiktionary;
- the candidate synset is a hypernym of the synset which contains a hypernym of the query word from Wiktionary;

- the candidate synset contains a word which is present in the definition of the query word from Wiktionary;
- the candidate synset is present in the list of English WordNet synset candidates (for Russian language only);
- the candidate synset is present in the list of hypernyms of the WordNet candidates (for Russian language only);
- the candidate synset contains words which occur on the Google results page;
- the candidate synset contains words which occur on the Yandex results page (for Russian language only).

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

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

4.3 Taxonomy Enrichment with Meta-Embeddings

Most existing approaches to Taxonomy Enrichment consider only a single type of distributional information, i.e. pre-trained word embeddings. Meanwhile, Taxonomy Enrichment models may benefit from combining different types of embeddings at the same time. As it can be seen from the approaches of the RUSSE'2020 participants, fasttext embeddings perform better on nouns and word2vec embeddings demonstrate larger scores on the verb subsets. Therefore, Tikhomirov and Loukachevitch [2021] study meta-embeddings approaches, which combine several source embeddings, to the hypernym prediction of novel words and show that meta-embedding approaches

obtain better results for this task if compared to other methods. Meta-embeddings are further used in our approaches for distributional and graph-based embedding combinations.

4.3.1 Base Meta-Embeddings

The easiest way of combining embeddings of different types is to concatenate them and use the concatenated vector as an input. It combines different subsets of distributional (fastText, word2vec, GloVe) embeddings. In addition to that, the authors perform Singular Value Decomposition (SVD) over this concatenation as proposed in Yin and Schütze [2016].

4.3.2 Autoencoded Meta-Embeddings

In Tikhomirov and Loukachevitch [2021] the authors use two variants of autoencoders for the generation of meta-embeddings: Concatenated Autoencoded Meta-Embeddings (CAEME) and Averaged Autoencoded Meta-Embeddings (AAEME) from Bollegala and Bao [2018]. The authors apply meta-embeddings for the first time for the Taxonomy Enrichment task.

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

$$E(s(w)) = h(w),$$

$$D(h(w)) = \hat{s}(w), \tag{4.1}$$

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

where E and D are the encoder and the decoder, and L is the loss used for training the autoencoder. The loss is implemented as the distance ($dist$) between the original and the reconstructed embeddings. The $dist$ can be any distance or similarity measure such as MSE, KL-divergence, or cosine distance. In their preliminary study,

the cosine distance showed the best results, so they use it further in their experiments.

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

$$m(w) = \frac{E_1(s_1(w)) \oplus E_2(s_2(w))}{||E_1(s_1(w)) \oplus E_2(s_2(w)||_2}, \tag{4.2}$$

where $\oplus$ is the concatenation operation.

The drawback of this model is the growing dimensionality of meta-embeddings for cases where multiple source embeddings are combined. To fight that, the concatenation operation is replaced with averaging, yielding AAEME. It 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 internal representations $E_1(s_1(w))$ and $E_2(s_2(w))$:

$$m(w) = \frac{E_1(s_1(w)) + E_2(s_2(w))}{||E_1(s_1(w)) + E_2(s_2(w))||_2}. \tag{4.3}$$

In CAEME, the dimensionality of the meta-embedding space is the sum of the dimensions of the source embeddings, whereas in AAEME it stays the same. Averaging in AAEME gives the possibility to avoid increasing the dimensionality of the meta-embedding.

4.3.3 Training of Autoencoders

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

$$\begin{aligned} L(w_a, w_p, w_n) = max(||m(w_a) - m(w_p))|| - \\ ||m(w_a) - m(w_n))|| + margin, 0), \end{aligned} \tag{4.4}$$

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

The algorithm for calculating triplet loss is as follows:

1. for each word presented in the taxonomy, they compile a list of semantically related words which includes synonyms, hyponyms and hypernyms;

2. at each epoch, they randomly select K positive words from this related words set and form a set of K negative words by selecting them randomly from the vocabulary;

3. if the word is not presented in the taxonomy, then they cannot form a list of related words for it. In this case, they generate positive vectors for it by adding random noise to its vector;

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

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

4.4 Conclusions

In the current section we described several existing approaches to the Taxonomy Enrichment task. In the next Chapter, we discuss DWRank — a model that comprises different types of embeddings and demonstrates promising results. Chapter 6 demonstrates the experimental results for all methods including the ones, described in the current chapter.

Chapter 5

Distributional Wiktionary-based Synset Ranking (DWRank)

We present a new method of Taxonomy Enrichment — Distributional Wiktionary-based synset Ranking (DWRank). It combines distributional representations with features from Wiktionary. First, we describe a simple baseline in Section 5.1 and then enhance it with additional features (Section 5.2). We also test various word and graph embeddings, as well as their combinations in DWRank-Graph and DWRank-Meta approached (Sections 5.4-5.5).

5.1 Baseline

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

The intuition behind the method is the following. We propose that if a synset of

a taxonomy is a *parent* of a word which is similar to our query word, it can also be a parent of this query word.

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

We rank the candidate set using the following features:

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

5.2 Distributional Wiktionary-based Synset Ranking (DWRank)

We extend the presented Logistic Regression model with new features that mainly account for the number of occurrences of a synset in the candidate lists of different synset associates (nearest neighbours) of the query word. We introduce the following new features:

- the number of occurrences (n) of the synset in the merged candidate list and the quantity $log_2(2 + n)$ which serves for smoothing,

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

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

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

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

5.3 Word Representations for DWRank

We test our baseline approach and DWRank with different types of embeddings: fastText [Bojanowski et al., 2017], word2vec [Mikolov et al., 2013c] embeddings for English and Russian datasets and also GloVe embeddings [Pennington et al., 2014] for the English dataset.

We use the fastText embeddings from the official website[1] for both English and Russian, trained on Common Crawl (CC) from 2019 and Wikipedia CC including lexicon from the previous periods as well. For word2vec we use models from Fares et al. [2017], Kutuzov and Kuzmenko [2017] for both English[2] and Russian.[3] We lemmatize words and synsets for both languages with the same UDPipe [Straka and Straková, 2017] model which was used while training the representations. For the out-of-vocabulary (OOV) words we find all words in the vocabulary with the longest prefix matching this word and average their embeddings like in Dale [2020]. As for the GloVe embeddings, we also use them from the official website[4] trained on Common Crawl with a vocabulary size of 840 billion tokens.

5.4 DWRank-Graph

The DWRank method extracts a set of candidate synsets based on the similarities of word vectors. So far, we used only distributional word vectors (fastText, GloVe, etc.) to represent words. However, graph-based representations can contain the taxonomic information which is absent in distributional embeddings [Makarov et al., 2021].

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

[1] https://fasttext.cc/docs/en/crawl-vectors.html
[2] http://vectors.nlpl.eu/repository/20/29.zip
[3] http://vectors.nlpl.eu/repository/20/185.zip
[4] https://nlp.stanford.edu/projects/glove/

5.4.1 Poincaré Embeddings

Poincaré embeddings is an approach for "learning hierarchical representations of symbolic data by embedding them into hyperbolic space — or more precisely into an "n-dimensional Poincaré ball" [Nickel and Kiela, 2017]. Poincaré models are trained on hierarchical structures and simultaneously capture hierarchy and similarity due to the underlying hyperbolic geometry. According to the authors, hyperbolic embeddings are more efficient on the hierarchically structured data and may outperform Euclidean embeddings in several tasks, e.g, in Taxonomy Induction [Aly et al., 2019].

Therefore, we use the Poincaré embeddings of our wordnets for the Taxonomy Enrichment task. We train the Poincaré ball model for our wordnets using the default parameters and the dimensionality equal to 10. This yields the best results on the link prediction task [Nickel and Kiela, 2017].

However, applying these embeddings to the task is not straightforward, because of the non-extensible nature of the Poincaré model's vocabulary. This means that the new words needed to be attached to the existing taxonomy will not have any Poincaré embeddings at all. Thus, we cannot make use of the embeddings similarity. To overcome this limitation, we compute top-5 fastText nearest synsets (analogously to the procedure described in Section 5.1) and then aggregate their embeddings in hyperbolic space using Einstein midpoint, following [Gülcehre et al., 2019]. The Einstein midpoint is straightforward to compute by adjusting the aggregation weights appropriately:

$$m_i(\{\alpha_{ij}\}_j, \{\mathbf{v}_{ij}\}_j) = \sum_j \left[\frac{\alpha_{ij}\gamma(\mathbf{v}_{ij})}{\sum_l \alpha_{il}\gamma(\mathbf{v}_{il})} \right] \mathbf{v}_{ij} \tag{5.1}$$

where the $\gamma(\mathbf{v}_{ij})$ are the Lorentz factors, that are given by $\gamma(\mathbf{v}_{ij}) = \frac{1}{\sqrt{1-||\mathbf{v}_{ij}||^2}}$. The norm in the denominator of the Lorentz factor is the Euclidean norm of the Klein coordinates of the point $\mathbf{v}_{ij}$.

The resulting vector is considered as an embedding of the input word in the Poincaré space. The pseudocode for this process is given in Algorithm 1.

Then, we use this vector space to generate candidates for the DWRank approach. As the model presented in Section 5 does not depend on the types of input embeddings,

Algorithm 1 Algorithm of translation of a new word fastText embedding to a non-inductive graph vector space.
Inputs: fastText embedding of the new word $new\_word\_ftt\_emb$, indexed set of fastText embeddings of the wordnet synsets ftt_embs, indexed set of graph embeddings of the wordnet synsets ni_graph_embs
Outputs: embedding $new\_word\_ni\_graph\_emb$ of the input word new_word in the graph vector space.

function TRANSLATE_TO_GRAPH_EMB($new\_word\_ftt\_emb$, ftt_embs, ni_graph_embs)
    $ftt\_neighbours \leftarrow ftt\_embs.get\_nearest\_neighbors(new\_word\_ftt\_emb, k = 5)$
    $graph\_embs \leftarrow []$
    for $neighbour \in ftt\_neighbours$ **do**
        $emb \leftarrow ni\_graph\_embs[neighbour]$
        $graph\_embs.append(emb)$
    end for
    $new\_word\_ni\_graph\_emb \leftarrow$ aggregate($graph\_embeddings, axis = 0$)
    return $new\_word\_ni\_graph\_emb$
end function

we are able to provide Poincaré embeddings as input.

5.4.2 Node2vec Embeddings

The hierarchical structure of the taxonomy is a graph structure, and we may also consider taxonomies as graphs and apply random walk approaches to compute embeddings for the synsets. For this purpose, we apply node2vec [Grover and Leskovec, 2016] which represents a "random walk of a fixed length l" with "two parameters p and q which guide the walk in breadth or in depth". Node2vec randomly samples sequences of nodes and then applies the skip-gram model of Mikolov et al. [2013a] 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 the Poincaré vector space, the node2vec model is not capable of representing out-of-vocabulary words. Thus, it is unable to map new words onto 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. THe process is the same as described in Algorithm 1.

5.4.3 Graph Neural Networks

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

Therefore, we test different Graph Neural Network (GNN) architectures — Graph Convolutional Network (GCN) [Kipf and Welling, 2017], Graph Attention Network (GAT) [Velickovic et al., 2018] and GraphSAGE (SAmple and aggreGatE) (GraphSAGE) [Hamilton et al., 2017] which make use of both fastText embeddings and the graph structure of the taxonomy.

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

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

5.5 DWRank-Meta

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

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

words and synsets. We performed experiments using the whole range of existing word embeddings: fastText, word2vec and GloVe and several graph vector representations: GCN, GraphSAGE, node2vec, and TADW. We do not cover the whole range of possible word and graph embeddings, but we demonstrate performance on various combinations of vector representations.

Chapter 6

DWRank Experiments

In this chapter, we present an empirical investigation of the proposed Taxonomy Enrichment method DWRank described in Chapter 5. We present the results on different datasets and perform an error analysis.

6.1 Experiments

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

6.1.1 Experimental Setup

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

The MAP metric should be interpreted as follows: the higher the score, the better the results. For instance, $MAP@k = 1.0$ means that all of the N correct hypernyms are present in the first top-N positions in the ranked list of candidates. Moreover, with $MAP@k = 1.0$ the first candidate is always correct. $MAP@k \geq 0.5$ means

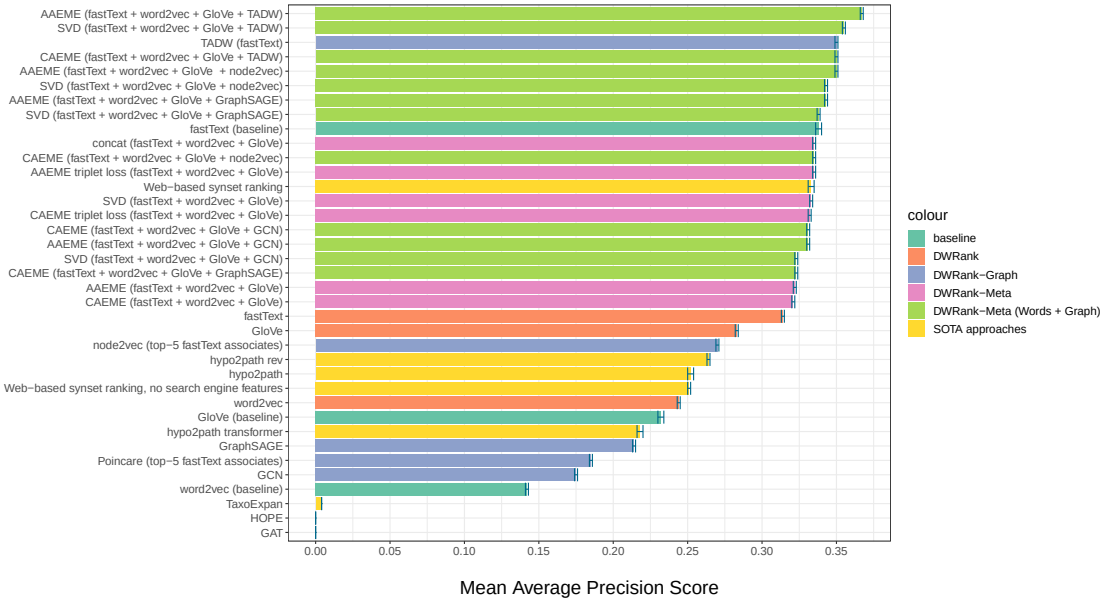


Figure 6-1: Comparison of the method performance on *nouns_1.6* dataset for English. Each colour denotes the method type and the embedding type used.

that at least one of the correct hypernyms is present in the two first positions (*top-2*) in the ranked lists of candidates. $MAP@k \geq 0.3$ means that at least one of the correct hypernyms is present in the first three positions (*top-3*) in the ranked lists of candidates.

We show the performance of different methods on the nouns attribution task for English (nouns 1.6) in Figure 6-1 and Russian (non-restricted nouns) in Figure 6-2. The X axis shows the MAP score for each method, the methods are listed along the Y axis. For DWRank-Meta models, we list the embeddings used in the model in brackets. The colors of the bars in the figures correspond to different types of input embeddings. The orange color stands for the vanilla DWRank – DWRank which uses only distributional embeddings. Purple denotes the DWRank-Graph variants. For the DWRank-Meta approaches exploiting only distributional embeddings, we use pink, and DWRank-Meta on word and graph embeddings is denoted with the bright green color. Previous SOTA approaches are shown in yellow.

The full results for all experiments on the English and Russian datasets can be seen in Appendix A in Tables A.6 and A.7, respectively. Here, when listing embeddings used in DWRank-Meta models, we use the shortcut "*words*" to denote the combination of fastText, word2vec, and GloVe embeddings.

6.1.2 Results

DWRank-Meta Figures 6-1 and 6-2 show that the leaderboard for both English and Russian nouns is dominated by DWRank-Meta models. While English benefits from the union of distributional and graph embeddings, distributional embeddings for Russian perform on par with their combinations with graph embeddings. Besides that, high-performing variants of DWRank-Meta for English feature TADW, node2vec, and GraphSAGE, whereas for Russian TADW is the only graph embedding model which does not decrease the scores of DWRank-Meta.

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

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

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

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

However, the choice of embedding models is crucial for the baselines (as well as for the vanilla DWRank which performs closely). We see that fastText outperforms word2vec and GloVe embeddings for almost all languages and datasets. The low scores of GloVe and word2vec embeddings on the baseline and DWRank methods

can be explained by data coverage issues. Fixed vocabularies of word2vec and Glove do not allow generating any representation for missing query words, whereas fastText can handle them.

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

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

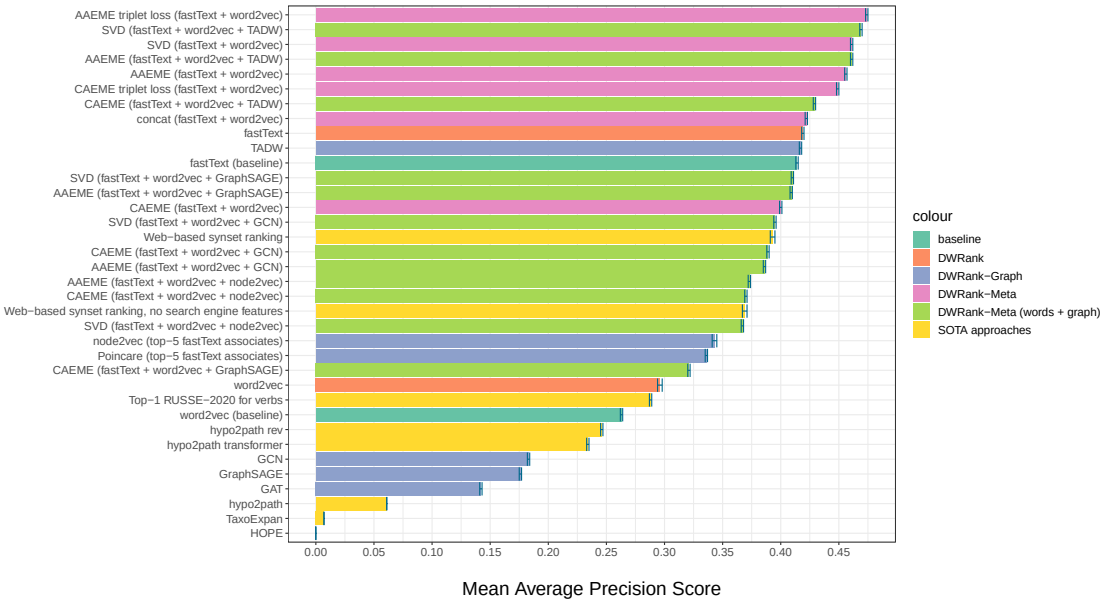


Figure 6-2: Performance of different models on the Russian non-restricted dataset. Each colour denotes the method type and the embedding type used.

Performance for different datasets Figures 6-1 and 6-2 as well as the results in the appendix show that there is no single best-performing model. While DWRank-Meta is almost always the best, for different datasets different variations of this model are the most successful. The results are usually consistent for the same language and POS (e.g., for different versions of English nouns datasets the best-performing model is the same), but there are exceptions to this regularity.

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

Table 6.1: Precision@k for the best-performing models for the English and Russian nouns datasets.

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

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

Table 6.1 shows the Precision@k scores for the best performing English and Russian models on the nouns datasets. Both of them are DWRank-Meta models with AAEME autoencoders. The English model uses three types of distributional embeddings and TADW graph embeddings, while the Russian model uses only fastText and word2vec but benefits from the triplet loss. We see that Precision@k is particularly high for Russian. There, over half of generated lists contain a correct synset in the first position. This shows that DWRank-Meta can successfully be used as a helper tool for taxonomy extension.

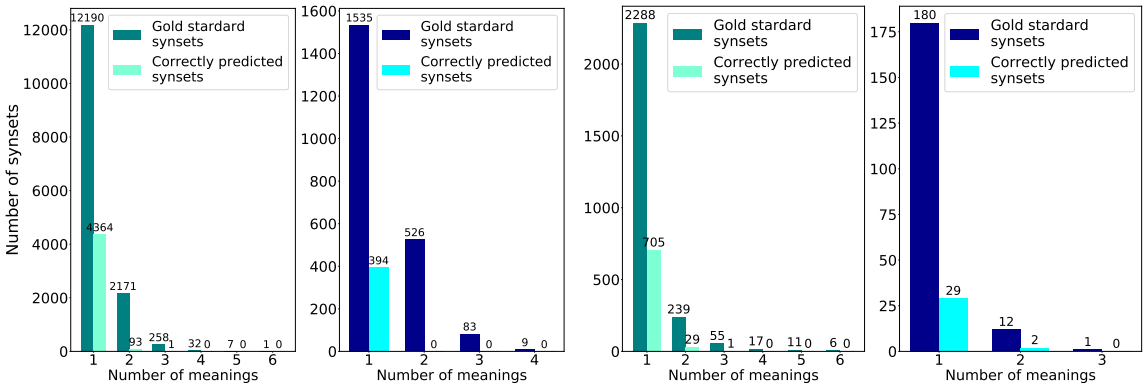
The results for English are lower. However, this should not be considered a sign of the lower performance of models for English. The Russian and English datasets consist of different words, so they cannot be directly compared.

6.2 Error Analysis

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

6.2.1 Comparison of Graph-based Approaches with Word Vector Baselines

First of all, we wanted to know to what extent the set of correct answers of graph-based models overlaps with one of the 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.



(a) Russian dataset (nouns and verbs)

(b) English dataset (nouns and verbs)

Figure 6-3: Distribution of words over the number of senses.

Therefore, for each new word we computed the average precision (AP) score and compared those scores across different approaches. We found that at least 90% of 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% of 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 performance significantly. This observation is corroborated by the scores of the combined models.

To contrast the performance of the text and graph embeddings and to demonstrate the input and the output formats of the models we present Tables A.1 and A.2 in Appendix A. They demonstrate the main features of the tested approaches. The examples do not pretend to be general case examples, however, they do provide an idea about the ranking of the results and the performance of text, graph and fusion embedding types.

6.2.2 Performance on Polysemous Words

The differences in word semantics make the dataset uneven. In addition to that, we would also like to understand whether the performance of models depends on the number of connected components (possible meanings) for each word. Thus, we examine how many words with more than one meaning can be predicted by the system.

Figure 6-3 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.

6.2.3 Error Types

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

Type 1. Extracted nearest neighbours can be semantically related words but are 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. The system predicts too broad / too narrow concepts:

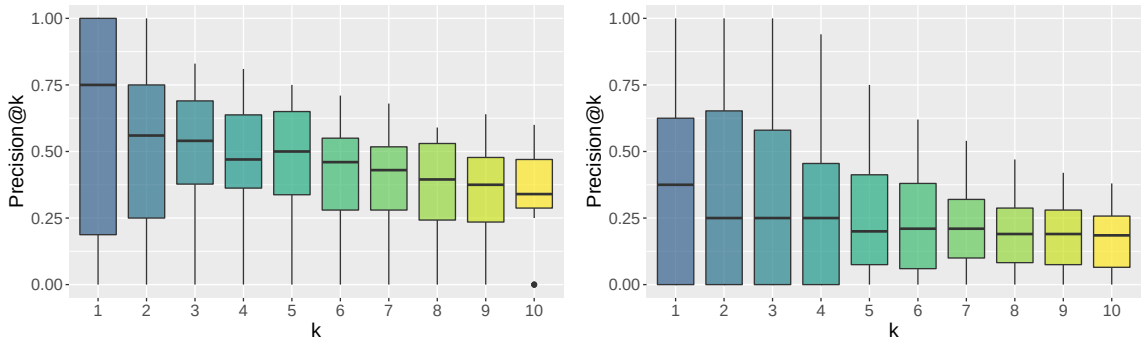
- midweek (WordNet); expected senses: day of the week, weekday; predicted senses: time period, week, day, season;
- **медянка** (smooth snake, RuWordNet); expected senses: **неядовитая змея, уж** (non-venomous snake, grass snake); predicted senses: **змея, рептилия, животное** (snake, reptile, animal).

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

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

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

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



(a) Russian dataset (b) English dataset

Figure 6-4: Manual datasets evaluation results: Precision@10.

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

Table 6.2: Words selected for the manual evaluation.

In order to check how useful the predicted synsets are for a human annotator (i.e., if a short list of possible hypernyms can speed up the manual extension of a taxonomy), we conduct the manual evaluation of 10 random nouns and 10 random verbs for both languages (the words are listed in Table 6.2). We focus on worse-quality cases and thus select words whose MAP score is below 1. Annotators with expertise in the field and 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 the Precision@k score (the share of correct answers in the generated lists from position 1 to k) for k from 1 to 10, as shown in Figure 6-4. We can see that even for words with MAP below 1 our model manages to extract useful hypernyms.

6.3 Conclusions

In the previous chapters, we paid a lot of attention to the models that attach predefined words to the existing taxonomy, achieving SOTA results. In the next chapter, we will present another view on the task providing necessary datasets, methods and experiments.

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. 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.

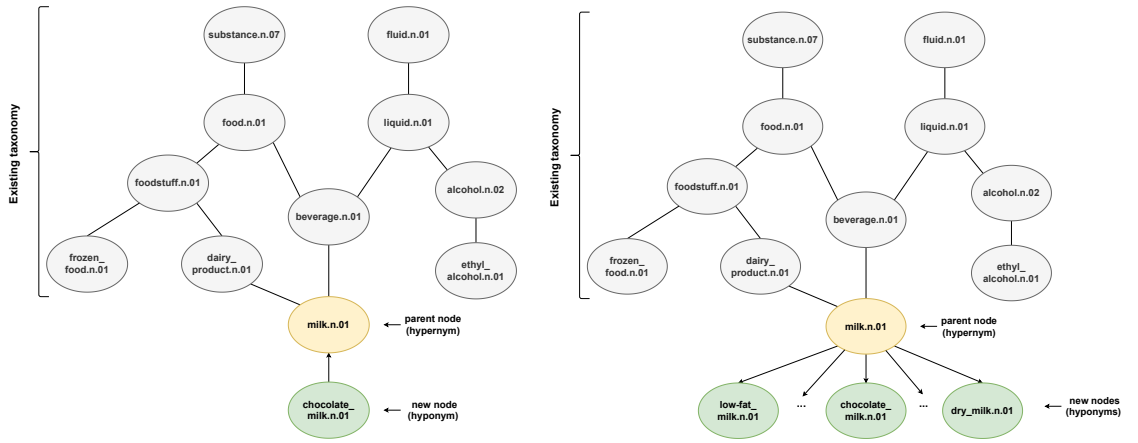
Chapter 7

Cross-modal Contextualized Hidden State Projection for Candidate-free Taxonomy Enrichment

7.1 Introduction

Normally, taxonomies are compiled manually by linguists, which is a time-consuming process. This also requires expertise and language proficiency. At the same time, many approaches have been proposed to update existing taxonomies. However, we argue about one crucial limitation of the existing setups for Taxonomy Enrichment questioning their usefulness in a real-world application. Namely, they all require some set of pre-collected orphans (new words). Figure 7-1a demonstrates the traditional Taxonomy Enrichment task setting where the system is provided with the candidate to add and the task is to find the correct place for it in the existing taxonomy. Compiling lists with the new words to add is an inherently challenging problem. It might be not clear to which of the multiple sources we would give our preference: neologisms, teenage slang from the Internet or professional jargon.

On the contrary, large pre-trained language models such as BERT [Devlin et al., 2019], ELMo [Peters et al., 2018], and GPT [Brown et al., 2020] already contain information about the majority of terms in a language as they were trained on huge web-scale corpora which can be further used in the downstream tasks. Many probing



(a) Standard taxonomy enrichment using candidate terms.

(b) Candidate-free generative taxonomy enrichment.

Figure 7-1: Two types of taxonomy enrichment task formulation. We explore option (b) in this chapter.

studies [Rogers et al., 2020, Jawahar et al., 2019, Ettinger, 2020] show that a vast amount of linguistic information is encoded inside large transformer networks, e.g., syntax or lexical semantics. Specifically, it has been shown that models are able to correctly retrieve hypernyms in cloze tasks.

In our study, we assume that a huge amount of knowledge from pre-trained models can be leveraged to predict new words missing in taxonomic resources. We suggest a novel candidate-free task formulation for Taxonomy Enrichment, arguing that compiling word lists may be redundant. Information about new words is already present in large pre-trained networks. Therefore, we provide subgraphs sampled from the existing taxonomy as input to predict hyponyms at a certain place (see Figure 7-1b as an example). We suggest using synsets 2 hops away from the target node, as further located synsets may not be semantically related. There would be not need in compiling lists of "parents" to predict hyponyms either, as language models should be able to predict words only if necessary.

The approach includes several stages: learning embeddings of WordNet taxonomy, projecting them into the hidden states space of BERT and decoding them back to text candidates. Intuitively, our approach can be seen as collecting information from graph modality and applying it to another modality. Therefore, the candidate prediction part of our method works as a regular language model in the sense that the

predictions are based on a pre-trained model, but the masked token representation either combines graph information with context from the running text or relies solely on the former.

Thus, the contribution of our work is three-fold:

- First, we formulate a new, yet realistic yet more challenging setting for the Taxonomy Enrichment task and present a new dataset based on WordNet 3.0 taxonomy [Miller, 1995]
- Second, we evaluate baselines for this task based on BERT and fastText [Bojanowski et al., 2017] models, demonstrating the difficulty of the task;
- Third, we propose a method for incorporating graph information into pre-trained language models, based on hidden contextualized state projection. It demonstrates superior performance in comparison with fasttext- and BERT-based baselines.

7.2 Related Work

There have been two important competitions that have introduced the task of Taxonomy Enrichment to the public: SemEval 2016 [Jurgens and Pilehvar, 2016] and our competition RUSSE-2020 [**Nikishina** et al., 2020a]. However, both their formulations required a predefined list of candidates. A detailed overview of taxonomy-related papers is presented in Jurgens and Pilehvar [2016], **Nikishina** et al. [2022].

At the same time, there exists a lot of research on how suitable is BERT for capturing and transferring information about the hypo-hypernym relationship Ravichander et al. [2020], Hanna and Mareček [2021], Schick and Schütze [2019]. For instance, Ravichander et al. [2020] examine hypernymy knowledge encoded in BERT representations. In their experiments, BERT demonstrated the ability to correctly retrieve hypernyms, however, they argue that it does not necessarily follow that BERT is capable of systematic generalisation.

Another paper about BERT's knowledge of hypernymy [Hanna and Mareček, 2021] applies several patterns to predict possible hypernym candidates: "[MASK], such

as x" and "My favorite [MASK] is x". Such prompts often elicit correct hypernyms from BERT. However, BERT still fails in 43% of cases, therefore, the authors cannot claim that BERT has a limited understanding of hypernymy. There exist many more Hearst patterns Hearst [1992] that aim to identify hypo-hypernym relationship in unlabeled texts Snow et al. [2006], Pantel and Pennacchiotti [2006].

	Embeddings	Pr@1	Pr@2	Pr@5	Pr@10	R@1	R@2	R@5	R@10
Inductive	Graph-BERT directed (node reconstruction)	0.127	0.099	0.064	0.041	0.127	0.113	0.150	0.182
	GraphBERT directed (graph recovery)	**0.190**	**0.163**	**0.115**	**0.073**	**0.190**	**0.182**	**0.260**	0.314
	Graph-BERT undirected (node reconstruction)	0.166	0.142	0.107	0.070	0.160	0.166	0.273	0.349
	Graph-BERT undirected (graph recovery)	0.164	0.140	0.100	0.062	0.164	0.153	0.227	0.268
	GAT	0.018	0.016	0.014	0.011	0.008	0.021	0.068	0.099
Non-inductive	Node2vec directed root2leaf	0.227	0.217	0.212	0.181	0.227	0.241	0.368	0.509
	Node2vec directed leaf2root	0.451	0.359	0.244	0.173	0.451	0.470	0.563	0.674
	Node2vec undirected	**0.988**	**0.807**	**0.515**	**0.321**	**0.988**	**0.987**	**0.988**	**0.990**
	Poincare directed	0.769	0.671	0.464	0.297	0.769	0.818	0.882	0.910
	Poincare undirected	0.716	0.618	0.434	0.283	0.716	0.727	0.804	0.862
	TADW	0.006	0.005	0.005	0.004	0.006	0.006	0.008	0.010
	GCN	0.021	0.024	0.028	0.030	0.021	0.033	0.073	0.137

Table 7.1: Graph embeddings comparison on the tree representation task.

Anwar et al. [2020] examine the application of context-aware word representation models for lexical units and frame role expansion task. This task is related to our setting in a sense of generation of meaningful substitutes with preservation of content. We adopt their context-aware methods for our task. In our case, the meaningful substitute will be generated for a masked hyponym with the preservation of meaning represented in projected embeddings (see Section 7.4).

7.3 Candidate-Free Taxonomy Enrichment Task

We formulate taxonomy enrichment in a new way avoiding the need of pre-supplied candidates (cf. Fig. 7-1a) making it more challenging yet realistic. Given a taxonomy $T = \{h, r, t\} \subseteq E \times R \times E$, the task is to predict new nodes $n \in N, N \not\subseteq E$, which are not yet included in the taxonomy T, starting from the current node $h_i \in E$.

7.3.1 Dataset

Our dataset relies on the latest version of the English wordnet — WordNet3.0, it contains 82,115 noun synsets and 117,798 lemmas. In this research, we perform experiments only on nouns.

From this taxonomy, we select 1000 nodes with children that are leaves. This means that they do not have hyponyms of their own. We also take into consideration the distance length from the root to the leaf which should be more than 5 hops. This allows us to exclude the case of predicting very abstract or broad concepts. For each "parental" hypernym, all its hyponyms (leaves) were replaced by a single "masked" node, e.g., *handwear.n.01* had hyponyms *glove.n.02* and *muff.n.01* that were replaced by a single *ORPHAN_100000243*. This place in the taxonomy was then considered for extension and the candidates predicted for the masked node could be compared against the true hyponyms. All in all, we masked 4,376 leaves with 1000 "[MASK]" tokens.

We limit our experiments to leaves only, replacing all children with one mask in order to be able to compare with a wide range of possible answers (like it typically is in real-life cases) and not searching for the one and only true option. Moreover, focusing on the leaves and not any nodes, in general, could help us to expand non-common and remote parts of the tree-like graph.

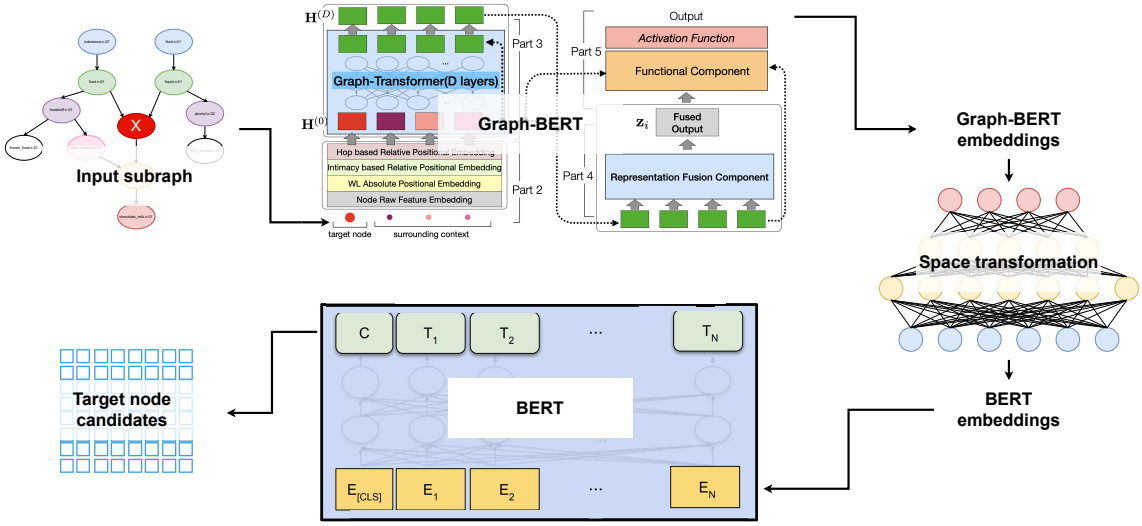


Figure 7-2: Cross-modal Contextualized Hidden State Projection Method (CHSP): graph-based BERT architecture that makes use of both node and text embeddings. Graph-BERT architecture illustration source: [Zhang et al., 2020], BERT architecture illustration source [Devlin et al., 2019]. Graph-BERT to BERT embeddings mapping will be trained on SemCor corpus [Langone et al., 2004].

7.4 Cross-modal Contextualized Hidden State Projection (CHSP) Method

The main idea of the approach is to predict new words using knowledge preserved in BERT and enhance the word generation process with graph information. Figure 7-2 demonstrates the overall architecture of the Cross-modal Contextualized Hidden State Projection (CHSP) approach that we use to solve the task. First, we train a graph representation model to compute graph embeddings. It is either node2vec [Grover and Leskovec, 2016] or Graph-BERT [Zhang et al., 2020], as depicted in Figure 7-2. Furthermore, we fit a projection layer to transform target graph embeddings to the BERT vector space. Then we apply the projected embeddings as input to the masked language modelling part of the BERT model. The prediction head generates new lemmas that are treated as candidate hyponyms for parent nodes. This process results in the gradual joining of the graph and textual modalities.

Further Subsections 7.4.1-7.4.5 discuss each part of the CHSP method and are ordered according to Figure 7-2. Subsection 7.4.1 describes the choice of graph embedding algorithm. Subsection 7.4.2 explains the projection of embeddings from graph space to BERT space. Subsection 7.4.3 explains how BERT is used to predict candidates from the projected embeddings. Subsection 7.4.4 gives a thorough explanation of the multi-token candidate generation algorithm. Finally Subsection 7.4.5 lists post-processing filters applied to the list of generated candidates.

7.4.1 Graph Embedding Computation

The first step concerns the choice of embedding we use to incorporate into the BERT language model. In Figure 7-2, it is the Graph-BERT model that is depicted, however, it could be any model for representing a graph structure. In order to identify which embeddings could be used for the task, we evaluated several inductive and non-inductive embeddings such as Graph-BERT, node2vec, GCN [Kipf and Welling, 2017], GAT [Velickovic et al., 2018], TADW [Yang et al., 2015] and Poincaré [Nickel and Kiela, 2017] embeddings. We also tested directed and undirected structures of Graph-BERT, node2vec and Poincaré. Finally, we performed both intrinsic and

extrinsic evaluations of the computed embeddings.

As for the intrinsic evaluation, which was conducted on the unmasked WordNet, we generated the top-10 nearest neighbours and computed Precision@k and Recall@k scores (k=1, 2, 5, 10) metrics that assess the number of hyponyms presented in the top-k list. We assume that the more "children" are presented in the list, the more suitable embeddings are for the tree-like structures and hyponym prediction. From Table 7.1 we can see that the best inductive embedding model is Graph-BERT on the directed graph and non-inductive node2vec on the undirected graph. We observe that node2vec and Poincaré show much higher scores than other methods. We speculate that this can be explained by the fact that these two algorithms are the only ones that do not incorporate textual features into the learned embeddings. Intuitively, similarity in textual features is not equal to the similarity in a graph. Additionally, degradation of node similarity in models that aggregate information from graph structure and node features is a known issue [Jin et al., 2021] and is linked to the over-smoothing problem. We believe that this could be one of the reasons why the approaches, which demonstrate promising results on a traditional Taxonomy Enrichment task [**Nikishina** et al., 2022], like GAT, GCN, and TADW do not perform well in predicting nearest neighbours. Moreover, it might be explained by the fact that such models better represent co-hyponymy or hypernymy, rather than hyponymy. Graph-BERT is known for avoiding an over-smoothing problem, thus, it performs much better than GAT, GCN and TADW. Furthermore, there is another explanation for the exceptional scores of node2vec trained on undirected taxonomy. The reason for that lies in the random walk sampling algorithm. Firstly, on average each node in taxonomy has one hypernym and 4.5 hyponyms. This means that of all random walks starting from a particular node generally there will be more walks that include the node's child rather than a parent. This presents more opportunities for learning about node's hyponyms. Secondly, undirected node2vec outperforms directed because sampled random walks can go in an arbitrary direction and thus provide a less biased neighbourhood.

For the extrinsic evaluation (evaluation of the downstream task), we have used two models: the best non-inductive and the best inductive embeddings. It is either a

Graph-BERT [Zhang et al., 2020] that accepts a sequence of node representations and their positional embeddings describing their local and global positioning in the graph or a node2Vec [Grover and Leskovec, 2016] that learns low-dimensional representations for nodes in a graph through the use of random walks. Graph-BERT was initialized with fastText raw textual features (each node – average of the according synset lemmas). It was trained for 200 epochs on the node attribute reconstruction task, and the process continued for 200 more epochs on the graph structure recovery task. The learning rate was set to 1e-3 and subgraph size to 5, and the resulting vectors were 300-dimensional. Node2vec was trained to generate embeddings of the same dimensionality, with 30 nodes in each random walk and 200 walks per node. In both cases, the remaining parameters are set to default values. However, as we further see, good coverage of hyponyms in the nearest neighbour list does not guarantee high performance on hyponym prediction.

7.4.2 Space Transformation

In order to project graph embeddings into the BERT embedding space, we use a simple Multi-layer Preceptron (MLP) neural network model. It consists of three hidden layers ($source\_embs \times 1024$, 1024×512, $512 \times target\_embs$) with the exponential linear unit (ELU) activation. During training, we used the AdamW [Loshchilov and Hutter, 2017] optimizer, which is a variant of the Adam [Kingma and Ba, 2015] with an improved implementation of weight decay. The loss function is a sum of two components: (1) cosine embedding loss between the GraphBERT model output and the corresponding target BERT embedding; (2) a negated cosine similarity between the GraphBERT embedding and a random negative example embedding. We consider any entity from the dataset that is not our current target. The components are presented below:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_+ - \mathcal{L}_- \\ \mathcal{L}_+ &= 1 - \cos\left(y, \hat{y}\right) \\ \mathcal{L}_- &= \max\left(0, \cos\left(y_{neg}, \hat{y}\right)\right), \end{aligned} \tag{7.1}$$

where y – target embedding, $\hat{y}$ – predicted embedding, y_{neg} – negative example embedding. The projection layer is trained for 500 epochs with batch size 64 and 1e-4 learning rate.

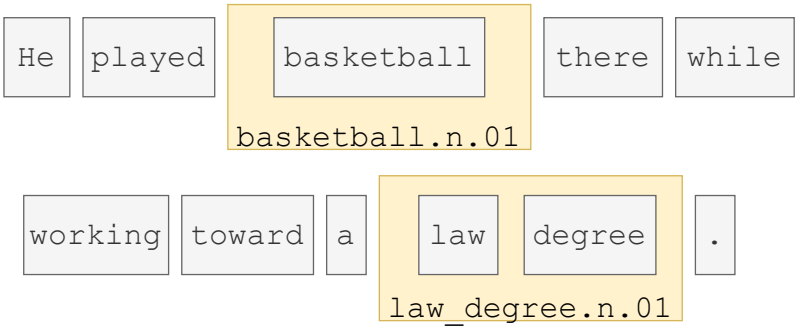


Figure 7-3: Excerpt from SemCor 3.0 used for learning a space transformation model.

BERT embeddings are contextualized. Therefore, to learn a projection from graph space into BERT, the target words should not be simply embedded as is because their representations differ in various contexts. In order to generate contextualized embeddings, we use the SemCor dataset [Langone et al., 2004]. It consists of 352 texts from Brown Corpus [Kucera and Francis, 1967], English text electronic collection. SemCor contains manually annotated sentences where words are matched with the according synsets. In our research, we use SemCor 3.0, which was automatically created from SemCor 1.6 by mapping senses from WordNet 1.6 to WordNet 3.0. We extract embeddings of annotated words and use them as contextualized target synset embeddings for learning projection. Figure 7-3 demonstrates the annotation scheme of SemCor. Therefore, we use the pre-trained GraphBERT embeddings of synsets "basketball.n.01" and "law_degree.n.01" as input to the space transformation model and consider BERT embeddings of "basketball", "law", and "degree" from this sentence as the target embeddings. If there is more than one target token for a single synset, then the tokens are averaged.

7.4.3 BERT Masked Language Modelling Prediction

We use *bert_base_uncased*[1] pre-trained configuration of BERT to embed a structure "[MASK] is a {parent}" where "{parent}" is a lemma of a hypernym whose hyponyms

[1] https://huggingface.co/bert-base-uncased

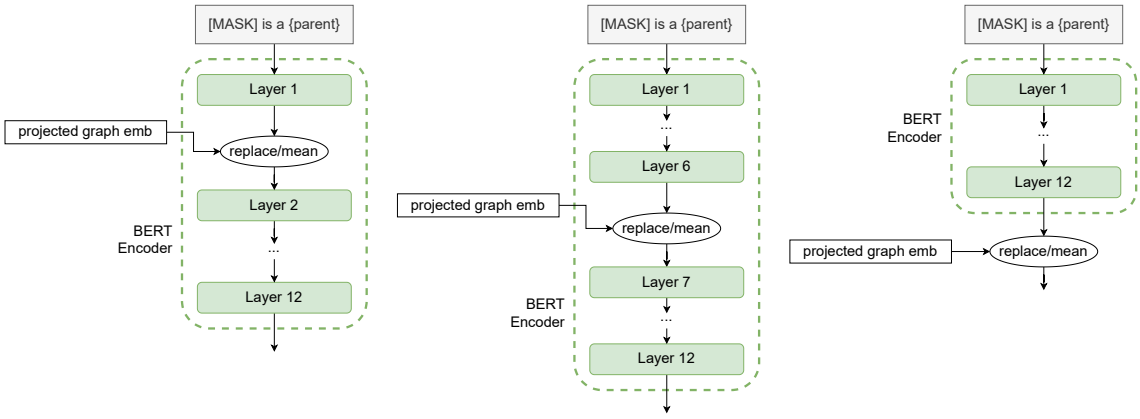


Figure 7-4: Illustration of replacement approaches. The projected graph embedding is inserted after (a) the 1st BERT encoder layer, (b) the 6th BERT encoder layer, (c) the 12th BERT encoder layer. The "replace/mean" denotes the replacement strategy: the projected embedding either replaces "[MASK]" token hidden representation or is averaged with it.

are to be predicted. In the following parts, we will refer to this structure as input context. The choice of the structure was not accidental. We have evaluated three different context constructions suggested in Hanna and Mareček [2021]:

1. "[MASK] is a/an {parent}";
2. "My favourite {parent} is a [MASK]";
3. "{parent} such as a [MASK]"

The scores for the predicted candidates with such patterns are presented in the first three lines of Table 7.2 and Table 7.3, respectively. These experiments are also repurposed as three baselines. *Precision*@10 indicates that the best results are produced by the first prompt, which proves to be the most stable. Therefore, we chose the first pattern for further experiments in all CHSP configurations.

Furthermore, we create three different approaches to incorporate graph embedding into the language model prediction:

- *pure-BERT* prediction: embedding of the "[MASK]" token is left as is;
- *replaced* prediction: embedding of the "[MASK]" token is replaced by projected graph embedding;

- *mixed* (or contextualized) prediction: embedding of the "[MASK]" token is averaged with projected graph embedding.

The replacement can happen at three different stages: after the first layer of BERT encoder, after the sixth (middle) or after the twelfth (last). In the first two cases, space transformation learns to project graph embeddings into intermediate hidden states and after replacement, the hidden states are passed through the remaining encoder layers. The replacement strategies are illustrated in Figure 7-4. Thus, by performing this process, we combine textual and graph modalities in order to improve candidate prediction at a certain place in the taxonomy.

7.4.4 Multi-token Prediction

Algorithm 2 Algorithm of multi-token generation with BERT.
Inputs: name of parent synset *parent*, graph embedding of according masked child node projected into BERT space $proj\_emb$, layer of replacement l_num, replacement strategy $repl\_strategy$
Outputs: sorted list $final\_res$ that consists of tuples (*candidate*, *score*).

```
function MULTI_TOK_GENERATE(parent, proj_emb, l_num, repl_strategy)
    tokens ← tokenize("[MASK] is a {parent}")
    hidden_states ← BERT.encode(tokens, proj_emb, repl_strategy, l_num)
    final_res ← predict_candidates(hidden_states, tokens, mask_pos = 0)
    return final_res
end function
```

```
function PREDICT_CANDIDATES(hidden_states, tokens, mask_pos)
    mask_hidden_state ← hidden_states[mask_pos]
    single_tokens, single_scores ← pred_single_mask(BERT, hidden_states, mask_pos)
    f_preds, f_scores ← extract_mask_preds(single_tokens, single_scores)
    multi_preds, multi_scores ← [], []
    for seq_len ∈ [2, 3] do
        new_tokens, new_scores ←
            ← beam_search(tokens, mask_pos, mask_hidden_state, seq_len)
        m_p, m_s ← extract_mask_preds(new_tokens, new_scores)
        multi_preds.append(m_p)
        multi_scores.append(m_s)
    end for
    final_res ← merge_sort_results(f_preds, f_scores, multi_preds, multi_scores)
    return final_res
end function
```

We experiment with single- and multi-token prediction. In the first case, one embedding produces one token. In the second, we adopt a condBERT [Dementieva

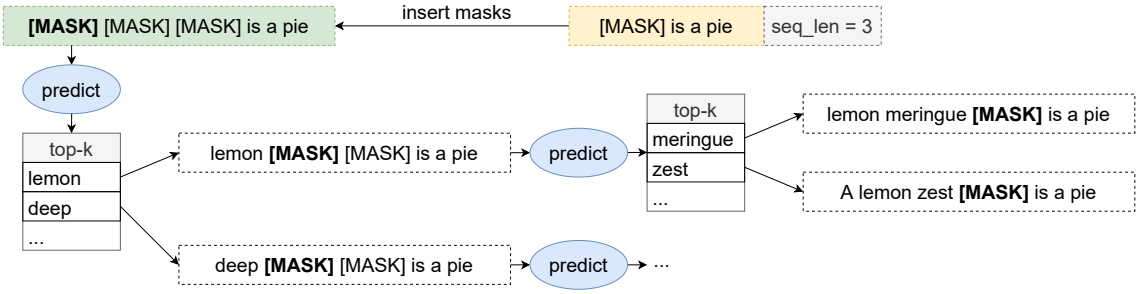


Figure 7-5: Beam search for a multi-token generation. In this figure 3-token case is illustrated. In our research, we also use a 2-token case which is generated similarly.

et al., 2021] multi-token generation mechanism. In addition to "[MASK] is a {parent}", "[MASK][MASK] is a {parent}" or "[MASK][MASK][MASK] is a {parent}" sentences are used. Furthermore, tokens are generated progressively by beam search while each multi-token sequence is scored by the harmonic mean of the probabilities of its tokens. The pseudocode for multi-token prediction is given in Algorithm 2. It is split into two functions: multi_tok_generate() and predict_candidates(). We are going to provide a detailed explanation for each of them.

The multi_tok_generate() function takes as input the name of a parent synset, projected graph embedding, layer of replacement for the incorporation of the embedding and the replacement strategy. Line 2 generates tokens for the context construction "[MASK] is a {parent}", and line 3 encodes them with the incorporation of projected embedding according to the scheme described in Subsection 7.4.3. Furthermore, the tokens and the hidden states are passed to the predict_candidates() function. It also takes the position of the "[MASK]" token, which in this context prompt is 0. Finally, predict_candidates() returns a sorted list of tuples $(candidate, score)$, where each *candidate* – predicted hyponym, and *score* harmonic mean of scores for each token in the multi-token sequence.

The predict_candidates() function starts with saving the embedding of the "[MASK]" token that incorporates graph information (line 2). Furthermore, in line 3 the single-token candidates are predicted. This is done because multi-token generation produces sequences of up to three tokens in length, which usually displaces correct single token predictions. Function extract_mask_preds() (line 3) separates the predictions of hyponyms from the generated sentences. For example, the sentence

"[MASK] is a claim" was predicted into "dibs is a claim". Then extract_mask_preds() extracts the predicted hyponym "dibs" and returns it as a candidate paired with its score. Next, multi-token candidates of lengths 2 and 3 are generated (line 6). It is done with a beam search (line 7), which is illustrated schematically in Figure 7-5. The beam_search() function takes as input the tokenized sentence, position of a mask, saved embedding of a mask and a maximum length of the multi-token sequence. The beam search starts with the insertion of one or two (according to the maximum length) additional mask tokens in the token sequence. Furthermore, the masks are predicted iteratively while maintaining the best sequences as in a classical beam search algorithm. For example, a sentence from Figure 7-5 "[MASK] is a pie" got extended into "[MASK][MASK][MASK] is a pie" for a 3-token case. Then, the new sentence is encoded with BERT in a usual way and the hidden state of the first mask is substituted with saved embedding that incorporates graph information. Candidates for the first mask are predicted, ranked, and a new set of hypotheses with the predictions of the first mask are generated (e.g., "lemon [MASK][MASK] is a pie" and "deep [MASK][MASK] is a pie"). The top ones are passed to the next iterations for prediction of the second mask, and the process is repeated. The beam search generation ends when the maximum sequence length of the multi-token prediction is reached. The top hypotheses sentences as well as their scores are returned. Next, in line 8 candidate hyponyms are extracted with extract_mask_preds() and together with scores are saved. Finally, multi- and single-token predictions are merged together and sorted by scores (line 10).

7.4.5 Post-processing

Predictions generated by the BERT language model contain a lot of noise like punctuation and common stop words such as "it", "this". Thus, we apply several filters to the generated set of new words.

The general procedure removes all predictions containing non-alphabetical symbols, usually, these are numbers or punctuation. Moreover, we remove all predictions that are in a list of English stop words obtained from Stopwords Corpus [Porter,

1980] in NLTK library[2]. The list contains 179 function words such as prepositions, pronouns, auxiliary verbs, etc.

The multi-token generation case requires further post-processing. A single multi-token candidate is a list of word pieces (or sub-words). If a word-piece is supposed to be used with a prefix it has a "##" mark in the beginning. For example, a word "strawberry" is tokenized as ["straw", "##berry"]. Furthermore, in order to be evaluated, such multi-tokens need to be merged together. If the first token in a multi-token candidate list starts with "##", it is skipped, as there is no prefix for it. If all tokens in a multi-token candidate list start with "##", the candidate is discarded. If a new (not first) token in a list does not contain "##", it will be added to the previously merged sequence with a preceding space. Continuing the "strawberry" example, the aforementioned merging rules can be illustrated as:

- ["straw", "berry"] $\longrightarrow$ "straw berry";
- ["straw", "##berry"] $\longrightarrow$ "strawberry";
- ["##straw", "##berry"] $\longrightarrow$ candidate is discarded;
- ["##berry"] $\longrightarrow$ candidate is discarded;
- ["##straw", "berry"] $\longrightarrow$ "berry".

Furthermore, we check merged candidates for containing permutations of the same sets of words and eliminate the repeating ones with lower scores. For example, if there are two multi-token candidates "apple pie" and "pie apple", the least probable is going to be discarded. Finally, the whole list of merged candidates is checked for duplicates and sorted by their scores.

7.5 Baselines

In our experiments we are using five baselines:

1. fastText (nearest neighbours);

[2]https://www.nltk.org/

2. BERT (parent embeddings on inference);

3. three patterns from Hanna and Mareček [2021], Schick and Schütze [2019]

7.5.1 FastText (Nearest Neighbours)

The first baseline uses 300-dimensional fastText [Bojanowski et al., 2017] English embeddings pre-trained on Common Crawl and Wikipedia. Hypernym embeddings are computed as an average of all lemmas embeddings. Furthermore, nearest neighbours of the resulting vectors are retrieved and scored as hyponym predictions. It is inspired by the baseline method in Chapter 5, which also combines fastText representations and nearest neighbours method. However, in their case, it is used for prediction of hypernyms of orphans. Our approach can be seen as a reverse of DWRank. In a single-token evaluation case, multi-token hyponyms are dropped from the list of gold hyponyms (see Section 6.1).

7.5.2 BERT (Parent Embeddings on Inference)

The second baseline uses BERT to encode each hypernym lemma and decode it back in a single- or multi-token setting. Predictions for each parent lemma are aggregated and evaluated. This method is loosely motivated by Anwar et al. [2020] and the idea of lexical substitution, which goal is to find meaning-preserving alternatives to a particular target word in its context. However, with this baseline, we wanted to evaluate BERT's ability to predict hyponyms in a contextless setting.

7.5.3 Pattern Comparison

The three last baselines are based on the approach described in these two publications: Hanna and Mareček [2021], Schick and Schütze [2019]. They propose a variety of constructions for prompting BERT in order to identify its linguistic capabilities and test its ability to capture semantic properties of words. Both works use a similar set of constructions, however, only Hanna and Mareček [2021] compare them against each other in order to identify the most efficient ones. According to their evaluations we have selected three best patterns: "[MASK] is a/an {parent}", "My favourite

{parent} is a [MASK]", "{parent} such as a [MASK]". The constructions were encoded with BERT and then decoded in single- and multi-token settings with "[MASK]" predictions treated as new candidate hyponyms.

7.6 Evaluation

The generated candidates are compared against the true candidates from the existing taxonomy. We use standard metrics for Information Retrieval like Precision@k and Mean Reciprocal Rank (MRR).

Both metrics are commonly employed in the Hypernym Discovery and Taxonomy Enrichment shared tasks, which require systems to produce ranked lists of potential hypernyms [Camacho-Collados et al., 2018, Dale, 2020]. Furthermore, numbers for both metrics are multiplied by 100 for clearer presentation.

Since we are experimenting with single- and multi-token generations, we apply according restrictions on the evaluation metrics. Namely, for single-token generation, we are computing upper-bound scores. A model incapable of multi-token generation will never predict such results, hence, a reduction of its score would be not fair. Thus, in case of the single-token generation, we are removing multi-token lemmas from the list of gold hyponyms and the generated candidates are evaluated against the reduced target set.

Method	Context	Replaced	MRR@5	MRR@10	MRR@20	Pr@1	Pr@5	Pr@10
Pattern comparison [Hanna and Mareček, 2021]								
"[MASK] is a {parent}"	Yes	No	2.461	2.704	3.091	1.546	1.289	1.057
"My favourite {parent} is a [MASK]"	Yes	No	0.554	0.863	1.001	0.000	0.464	0.490
"A {parent} such as a [MASK]"	Yes	No	0.168	0.193	0.235	0.000	0.155	0.103
BERT (parent embedding on inference)	No	No	1.003	1.083	1.203	0.940	0.251	0.188
fastText (nearest neighbours)	No	No	2.400	3.500	4.000	0.130	1.839	2.100
CHSP (Graph-BERT)	Yes	Mix	**7.229**	**8.037**	**8.624**	**3.608**	**3.247**	**2.474**

Table 7.2: Prediction scores for single-token hyponyms generation for different source graph embeddings and replacement strategies (x100).

7.7 Experiments

Our experiments can be categorised by the following features: source graph embeddings, usage of context structure, replacement layer and replacement strategy. This

Method	Context	Replaced	MRR@5	MRR@10	MRR@20	Pr@1	Pr@5	Pr@10
Pattern comparison [Hanna and Mareček, 2021]								
"[MASK] is a {parent}"	Yes	No	0.930	1.027	1.177	0.600	0.460	0.370
"My favourite {parent} is a [MASK]"	Yes	No	0.425	0.693	0.844	0.000	0.361	0.438
"A {parent} such as a [MASK]"	Yes	No	0.051	0.137	0.137	0.000	0.052	0.077
BERT (parent embedding on inference)	No	No	0.320	0.345	0.390	0.300	0.080	0.060
fastText (nearest neighbours)	No	-	1.860	**2.673**	**3.069**	0.100	**1.420**	**1.620**
CHSP (Graph-BERT)	Yes	Yes	**2.150**	2.281	2.378	**1.600**	0.740	0.530

Table 7.3: Prediction scores for multi-token hyponyms generation for different source graph embeddings and replacement strategies (x100).

Graph embeddings	Context	Replaced	Layer	MRR@5	MRR@10	MRR@20	P@1	P@5	P@10
Node2vec	Yes	Yes	1st	0.975	1.831	2.252	0.000	0.670	1.186
		Mix	1st	2.328	2.685	2.903	1.546	1.186	1.005
		Yes	6th	3.316	3.799	4.070	1.031	1.804	1.340
		Mix	6th	2.414	3.079	3.391	1.289	1.289	1.469
		Yes	12th	2.436	3.185	3.486	1.289	1.082	1.160
		Mix	12th	3.329	4.073	4.597	1.031	1.649	1.675
Graph-BERT	Yes	Yes	1st	4.502	4.995	5.371	3.093	1.598	1.340
		Mix	1st	1.448	1.813	2.033	0.773	0.876	0.979
		Yes	6th	5.503	6.216	6.453	3.093	2.371	2.010
		Mix	6th	2.981	3.500	3.836	1.546	1.649	1.495
		Yes	12th	5.215	5.674	6.027	3.093	2.113	1.598
		Mix	12th	**7.229**	**8.037**	**8.624**	**3.608**	**3.247**	**2.474**

Table 7.4: CHSP prediction scores for single-token hyponyms generation for different source graph embeddings, replacement strategies and substitution layer (x100).

Graph embeddings	Context	Replaced	Layer	MRR@5	MRR@10	MRR@20	P@1	P@5	P@10
Node2vec	Yes	Yes	1st	0.945	1.231	1.395	0.515	0.515	0.515
		Mix	1st	0.287	0.374	0.492	0.000	0.206	0.180
		Yes	6th	0.587	0.674	0.732	0.200	0.300	0.210
		Mix	6th	1.924	2.073	2.193	1.200	0.740	0.550
		Yes	12th	0.520	0.534	0.586	0.500	0.120	0.070
		Mix	12th	0.453	0.534	0.610	0.400	0.120	0.110
Graph-BERT	Yes	Yes	1st	1.908	2.054	2.149	1.400	0.680	0.500
		Mix	1st	1.350	1.522	1.625	0.800	0.600	0.500
		Yes	6th	**2.150**	**2.281**	**2.378**	**1.600**	**0.740**	0.530
		Mix	6th	1.468	1.694	1.806	0.700	0.700	**0.560**
		Yes	12th	1.278	1.312	1.368	1.200	0.340	0.190
		Mix	12th	1.767	1.899	2.071	1.400	0.540	0.390

Table 7.5: CHSP prediction scores for multi-token hyponyms generation for different source graph embeddings, replacement strategies and substitution layer (x100).

section is divided into two parts. The first subsection compares various combinations of CHSP configurations. The second subsection analyzes the performance of the best CHSP configurations against the baselines.

7.7.1 Replacement Strategy Comparison

Table 7.4 and Table 7.5 compare single-token and multi-token hyponym predictions for methods with different source embeddings, replacement strategies and replacement

layers. We observe that in a single-token case for both node2vec and Graph-BERT the best replacement point is after the last (12th) BERT encoder layer with the first and the sixth being close seconds. We hypothesise that the reason is that, when injecting the projected graph embedding at earlier stages, the remaining encoder layers dilute information incorporated in the embedding, thus deflecting from the right answers. In the case of single-token generation, Graph-BERT with the replacement point after the last layer is a clear winning strategy among all the combinations. On the contrary, for multi-token generation significantly better scores were obtained by replacement after the 6th layer. We suggest that this replacement strategy helped to diversify generated subwords and produce more meaningful results.

In general, the "mixing" replacement strategy produces better results for the last-layer replacement strategy, because it allows for the incorporation of context information encoded in a final hidden state of the "[MASK]" token. However, there are some cases where the context actually diverts the method from the real answer (see Section 7.8). The complete replacement showed better scores in 1st and 6th layer replacement, because this strategy already incorporates a lot of context in the "[MASK]" embedding while passing it through the remaining layers of the encoder, and "mixing" replacement reduces the influence of projected embedding too much. To sum up, both replacement strategies are important and none of them can be deemed winning as there is a clear pattern of where to apply each of them.

We can observe that node2vec did not perform as well as was expected judging from the graph embedding comparison. In many cases of single-token generation, words synonymous with the hypernym were predicted, instead of hyponyms. The reason for the low scores on node2vec embeddings might be explained by the fact that the Graph-BERT embeddings are easier to transform to the BERT vector space. Another hypothesis is that the performance on hyponym prediction does not guarantee high scores on predicting hyponyms for Taxonomy Enrichment.

In the next subsection, we are going to compare our CHSP against the baselines. For our method, we are using winning combinations for single- and multi-token strategies, namely, Graph-BERT with mixing after the 12th layer and Graph-BERT with complete replacement after the 6th layer.

7.7.2 Overall Comparison

Table 7.2 and Table 7.3 contain the overall scores for different hyponym prediction methods. We can see that our approach significantly outperforms other methods on single token setup, however, it fails on predicting multi-token candidates. We observe that the patterns from Hanna and Mareček [2021], Schick and Schütze [2019] show results are mostly far from the top ones. This happened because the context encapsulated in the patterns, in general, contains little information. We also see that our method outperforms the BERT baseline (which is a simple prediction of encoded parent synset) and a simple approach on fastText nearest neighbours candidates. Even though the results for multi-token predictions are better for the fastText baseline, we still consider our method to be the most effective, as fastText is also not capable to predict multi-token candidates and yields to our method in the single token setup.

For all setups, the multi-token generation did not result in an improvement in the scores. This can be explained by the flawed nature of our multi-token sampler and suggests a major stream of future work.

7.8 Error Analysis

We can categorise common errors into several groups: failing to differentiate the real meaning of the hypernym, prediction of synonymical/same domain words instead of hyponyms, and weakness of multi-token generator.

Many times, the method fails to recognise a rare meaning of a synset and mistakes it for a more common one. For example, for hypernym "depression.n.10" (pushing down) the correct prediction would be "click". However, almost all results contain mostly mental illness or medical-related predictions, e.g., headache, coma, schizophrenia, etc.

An example of the second type of errors might be predictions of multi-token pipeline with Graph-BERT embeddings for the "jazz_musician.n.01" hypernym. While the correct answer is "syncopator", the top produced predictions are "singer", and "dj", which obviously come from the same music-related domain.

citrus.n.01			
Gold hyponyms: citrange, citron, grapefruit, kumquat, lemon, lime, mandarin, orange, pomelo			
	pure BERT	replaced	mixed
1	fruit	date	date
2	one	year	tree
3	rose	horse	year
4	another	turkey	snow
5	**citrus**	dates	horse
6	cherry	tree	turkey
7	**orange**	snow	dates
8	tomato	calendar	winner
9	mine	winner	grass
10	wood	loser	trees

Table 7.6: Example of node2vec embeddings for the node "citrus.n.01" (single-token generation)

beverage.n.01			
Gold hyponyms: alcoholic drink, oenomel, fruit crush, cooler, alcoholic beverage, hot chocolate, fizz, ade, milk, inebriant, cocoa, drinking chocolate, drinking water, tea, java, mixer, refresher, tea-like drink, alcohol, coffee, fruit drink, ginger beer, wish-wash, potion, soft drink, near beer, smoothie, chocolate, cyder, intoxicant, fruit juice, cider, mate, hydromel			
	pure BERT	replaced	mixed
1	beer	**milk**	**coffee**
2	**coffee**	drink	**milk**
3	**alcohol**	**coffee**	drink
4	water	butter	**tea**
5	cola	pot	chocolate
6	**tea**	whisky	butter
7	wine	**tea**	beer
8	**milk**	turkey	whisky
9	chocolate	chocolate	brandy
10	rum	brandy	water

Table 7.7: Example of Graph-BERT embeddings for the node "beverage.n.01" (single-token generation).

For multi-token node2vec we observed a lot of cases where one strong word was produced and further multi-token hypothesis would retain this first word and simply permute other different words. Example output for test hypernym "suburb.n.01": suburb, suburb suburbs, suburbs, suburb suburban, suburb suburbs suburban, etc.

Because of the weak multi-token decoding mechanism, many predictions failed. For example, none of the setups managed to produce adequate hyponyms for "berry.n.01", because all correct answers are multi-token in BERT vocabulary (e.g.,

"ras"+"##p"+"##berry").

	meal.n.01		
	Gold hyponyms: nosh-up, tea, snack, breakfast, supper, brunch, tiffin, lunch, refection, mess, ploughman's lunch, dejeuner, feast, spread, afternoon tea, picnic, dinner, square meal, luncheon, teatime, banquet, bite, buffet, potluck, collation		
	pure BERT	replaced	mixed
1	life	**breakfast**	**breakfast**
2	food	breakfast lunch	breakfast lunch
3	**dinner**	**lunch**	**lunch**
4	**lunch**	breakfast dinner	breakfast dinner
5	**breakfast**	breakfast lunch dinner	breakfast lunch dinner
6	everything	lunch dinner	lunch dinner
7	love	breakfast dining	**dinner**
8	tomorrow	breakfast meals	breakfast meal
9	today	breakfast meal	breakfast lunch meal
10	nothing	breakfast lunch dining	breakfast meals

Table 7.8: Example of Graph-BERT embeddings for the node "meal.n.01" (multi-token generation)

All in all, the results are diverse and controversial. For instance, Table 7.6 demonstrates that graph information from node2vec is confusing for the model. According to Tables 7.7 and 7.8, Graph-BERT improves the ranking of the results. However, none of the models handles multi-token prediction: the only case where the model manages to predict the correct answer is presented in Table 7.9.

We reckon that the low scores could be explained by the task complexity and automatic evaluation on a closed list of candidates excluded from taxonomy. For instance, the model can generate correct candidates but they are not yet included in the taxonomy. In this case, the evaluation system will still mark them as incorrect. Another reason for the low results is the way the test set has been generated. In most previous work [Cho et al., 2020] the data is selected from the well-known and widespread domains like "pets", "food", "sport", etc. Our test set is generated to respect the taxonomy structure and normally comprises narrow and uncommon terms, which BERT does not manage to process. At the same time, we can see that simple examples like "beverage" or "meal" gain better scores. As future work, we want to tackle the problem of rare terms.

stock.n.01			
Gold hyponyms: capital stock, treasury stock, quarter stock, preference shares, growth stock, preferred stock, no-par-value stock, voting stock, common shares, authorized shares, hot stock, ordinary shares, authorized stock, float, reacquired stock, common stock, no-par stock, common stock equivalent, treasury shares, preferred shares, hot issue, control stock, watered stock			
	pure BERT	replaced	mixed
1	stock	capital	capital
2	one	capital cash	capital cash
3	c	capital investment	capital investment
4	b	capital financing	capital money
5	today	capital funds	capital financial
6	x	capital financial	capital equity
7	gold	capital income	**capital stock**
8	everything	capital funding	capital financing
9	life	capital revenue	capital funds
10	r	capital crop	capital leverage

Table 7.9: Example of node2vec embeddings for the node "stock.n.01" (multi-token generation).

7.9 Conclusion

In this chapter, we presented a novel candidate-free task formulation for Taxonomy Enrichment. We performed a computational study of various methods using knowledge from BERT. We compared different graph-based embeddings on the task of hypernym prediction and projected the best inductive and non-inductive representations to the BERT vector space. Then we identified the best position for the projected graph embedding to incorporate into the BERT model and compared the results with several baselines. The results demonstrate that the incorporation of graph embedding to the [MASK] embedding (at both "replace" and "mix" strategies) is beneficial in the task of hyponym prediction using contextualized models like BERT. Overall, this research has proven the difficulty of the novel task. The BERT architecture does not allow us to easily operate with multi-token words and the pipeline accumulates errors in each component. This leaves room for improvement for generative models like GPT or T5 and their prompt-tuning.

Despite the low results of the application of graph-based methods, we propose

further exploration of the combination of textual and graph modalities to predict new nodes to the existing taxonomy. In the next chapter we present a visualization of the CHSP method in a demonstration system which is able to generate new candidates.

Chapter 8

System Description

In this chapter, we demonstrate the application of our approach for candidate-free Taxonomy Enrichment, introducing **TaxFree** — a visualization tool for reviewing and enhancing taxonomies. This tool shows how the existing taxonomy can be enriched automatically without predefined candidates on the example of WordNet-3.0.

Plenty of tools are available for generic visualization of networks like Gephi [Bastian et al., 2009], GraphX [Gonzalez et al., 2014], D3[1], and GraphViz [Ellson et al., 2001]. At the same time, there exists some software specific for the visualization of wordnets which are not available from the original interface of WordNet. For example, Collins [2006] is one of the earliest papers that present a design paradigm. Visualization from Kamps and Marx [2002] demonstrates not only the relations between synsets but also denotes lemmas as graph nodes. WordNet Atlas [Abrate and Bacciu, 2012] is designed for "users like computer scientists that are not familiar with computational linguistics and/or WordNet".

TaxFree (see the example of the visualization page in Figure 8-1) is an open-source, web-based visualization and enrichment tool for taxonomies. We demonstrate the capacities of **TaxFree** on WordNet-3.0 [Miller, 1995] with support for the visual representation of WordNet synsets using ImageNet [Deng et al., 2009]. The tool applies BERT [Devlin et al., 2019] as a pretrained language model to predict new nodes (synsets). It allows to perform the search in WordNet by words (lemmas) or nodes (synsets), to visualize the context of the query word (with $hop = 2$), and to

[1] https://github.com/d3/d3

generate new leaf nodes or nodes between the two existing ones. In order to create nodes, we apply the Cross-modal Contextualized Hidden State Projection Method. This approach includes several stages: (i) learning embeddings of the WordNet taxonomy and new synsets at the required places that we want to predict, regarding such synsets as masked, (ii) projecting all graph embeddings into the hidden states space of BERT, and (iii) decoding them back to text candidates.

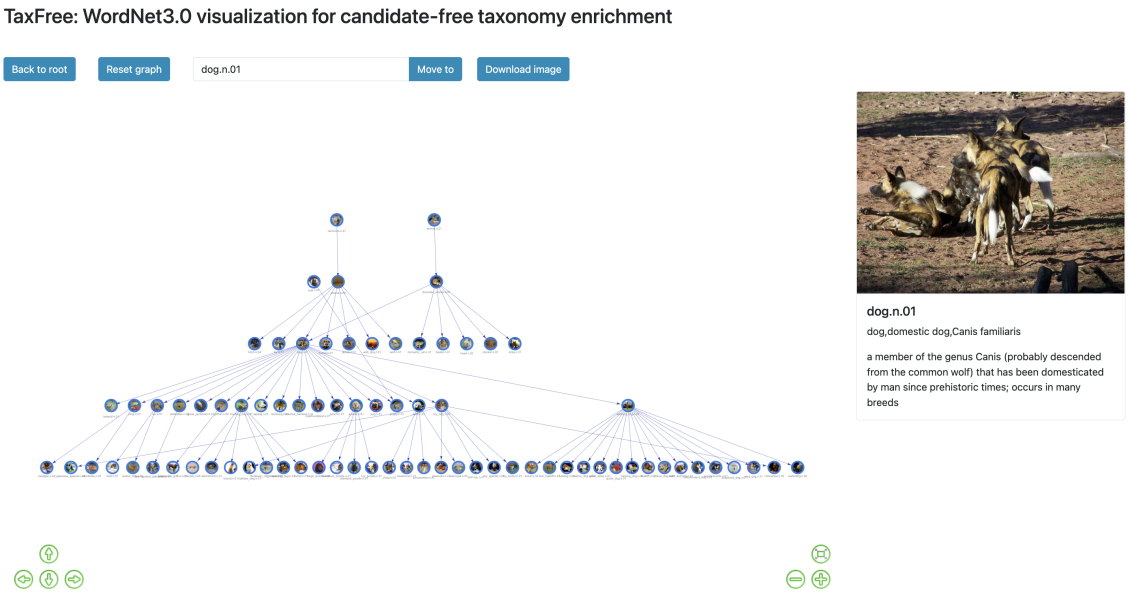


Figure 8-1: Visualization example for the node "dog.n.01"

Thus, the contribution of this demo is three-fold:

1. it performs a search on taxonomy and visualization of query node within its context (on the example of the English WordNet-3.0);

2. it allows the automatic extension of the existing taxonomy using the Cross-modal Contextualized Hidden State Projection Method;

3. it integrates ImageNet representations to the WordNet synset description card.

We also provide the links to the demo[2], code[3] and the screencast video demonstrating the system[4] in the footnotes.

[2] http://taxgen.ltdemos.informatik.uni-hamburg.de
[3] https://github.com/skoltech-nlp/taxgen-demo
[4] https://youtu.be/GF2AVlnWGag

8.1 System Design

TaxFree is designed to help lexicographers to update taxonomies and to inspect them for modifications. In the current section, we discuss each part of the tool and its usage in detail.

8.1.1 Software Architecture

The system is a web-based user interface through which users can explore the WordNet-3.0 taxonomy. The frontend is implemented with `JavaScript` library `vis.js`[5] used to display networks consisting of nodes and edges. It supports the hierarchical layout and allows us to interact with the network. The backend is written in `Python` using the `Flask`[6] framework. It has an Application Programming Interface (API) with several "`GET`" and "`POST`" queries that maintain the functioning of the system: (i) searching for synsets, (ii) retrieving image by node id, (iii) getting the current node graph context, and (iv) generating new nodes.

8.1.2 Main Page

The start page in Figure 8-1 shows the highest level of the taxonomy. It is a tree with the root node "entity.n.01" which is highlighted with green color. In most cases, the target node is displayed within its two-hop neighbourhood (if any). To the right of the graph visualization, there is a card with the description of the current node: its image from ImagNet (if any), definition and the list of lemmas. Above the graph visualization box there are two buttons: **"Reset graph"**, **"Back to root"** and a search box with a **"Move to"** button. **"Reset graph"** deletes all generated nodes from memory and displays only the initial WordNet-3.0 graph. **"Back to root"** returns the user to the display of the root of the taxonomy, leaving all generated nodes untouched. The search bar allows users to easily navigate through the taxonomy and display subgraphs for the queried node. More details for each box are provided in the corresponding subsections.

[5] https://visjs.github.io/vis-network/docs/network/
[6] https://flask.palletsprojects.com/en/2.2.x/

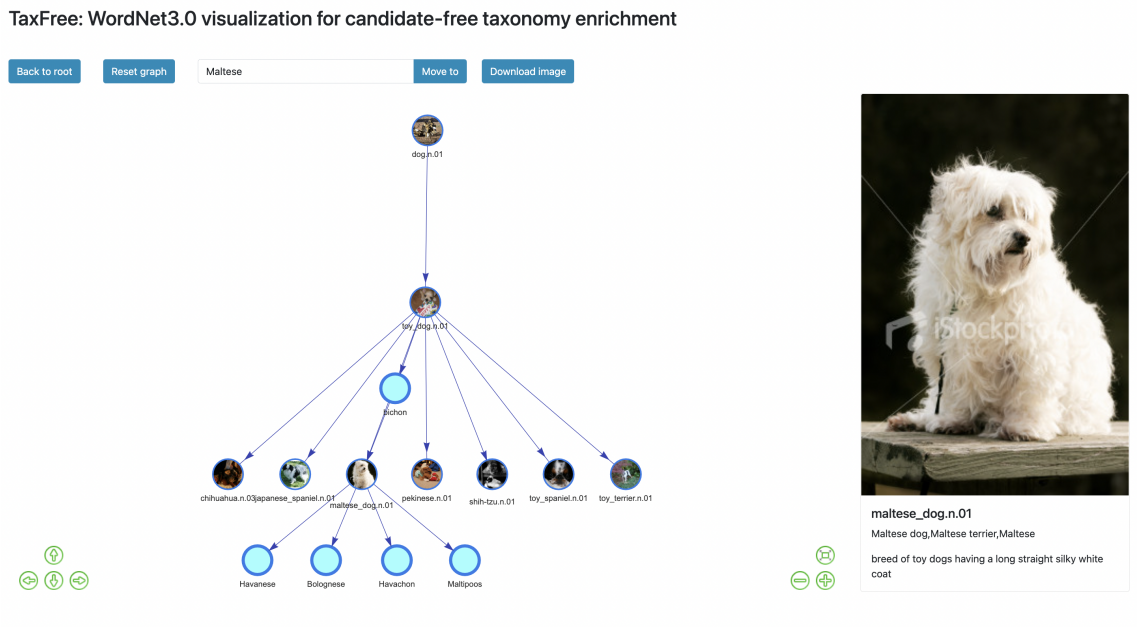


Figure 8-2: Generation of a new node for the leaf node "maltese_dog.n.01"

8.1.3 Synset Search

The search bar accepts both synset names and lemmas and helps to disambiguate unclear queries to WordNet-3.0. The user can enter a word or a phrase separated by spaces or underscores. Moreover, noun synsets such as "cat.n.01" or "standard_poodle.n.01" are also accepted. If the synset name is not recognized there is no error displayed, but the search bar becomes empty again. In he case of entering a word (lemma), the following pipeline is applied:

1. If there is only one synset corresponding to the query lemma, then this synset is displayed.

2. If there is more than one synset, the user is forwarded to the subgraph of the most common synset, displaying other disambiguation options under the synset description card (see Figure 8-3). Each disambiguated synset is presented with its synset name and definition.

After the query synset has been identified, the tool opens the required page with the query synset as the target node in the f context. The subgraph display and the synset description card are described in the following subsections.

dog.n.01

dog,domestic dog,Canis familiaris

a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds

Maybe you meant:

frump.n.01	a dull unattractive unpleasant girl or woman
dog.n.03	informal term for a man
cad.n.01	someone who is morally reprehensible
frank.n.02	a smooth-textured sausage of minced beef or pork usually smoked; often served on a bread roll
pawl.n.01	a hinged catch that fits into a notch of a ratchet to move a wheel forward or prevent it from moving backward
andiron.n.01	metal supports for logs in a fireplace

Figure 8-3: Disambiguation block for the "dog" lemma

8.1.4 Subgraph Display

Central (query) synset is displayed with the closest "relatives" at most two hops away from the query in the central box of the page. It might be less if there are no neighbours at a certain step away from the target node. It has green borders highlighting that the current image is the target one. However, if the image from the ImageNet is not presented, then the whole node is colored green. Other nodes are not highlighted and colored in blue (in case there is no image to display).

Figure 8-2 shows the result page for the synset "maltese_dog.n.01" as an example

to demonstrate synsets with images. Here, all nodes have their representations from WordNet-3.0. The node "maltese_dog.n.01" is a leaf node, therefore, it is placed at the bottom of the graph and has only co-hyponyms at the same level, one hypernym "toy_dog.n.01" and one hypo-hypernym "dog.n.01". The arrows always have the same direction: from abstract words to more concrete ones. Clicking on a node twice will open a subgraph for this node, as it would be considered the next query word. Therefore, it is possible to navigate through the graph even without queries. The graph can be downloaded using the **"Download graph" button**.

At the bottom of the visualization box, there are centring and in/out zoom buttons. The graph can be moved using the mouse or keys "`left`", "`right`", "`up`" and "down" `buttons` on the screen.

8.1.5 Synset Description Card

To the right of the subgraph display, there is a card with the summary of the query node. It consists of a definition, image from the ImageNet (if any), synset name, and list of lemmas. If any information about the node is missing, the row is skipped. The image for the node is selected randomly from the pool of the corresponding images, usually the first one from the ImageNet dataset. According to the statistics, only 19.167 synsets have their images.

8.1.6 New Synsets Prediction

Figure 8-2 demonstrates the process of adding new nodes to the taxonomy using the algorithm described in Section 5. First, we may generate a new node starting from a leaf. By clicking twice on it, we can generate children for the "maltese_dog" synset, which does not possess hyponyms. Otherwise, they would be displayed as "maltese_dog" is the central node. Furthermore, by clicking twice on the target (query) node we can predict a candidate child for it. Figure 8-2 shows that there were generated new nodes — the names of dog breeds. Another option for new synset generation is predicting a new node between two existing nodes (meaning that one of them is hypernym to the other). To generate this node, the user has to click

twice on the edge that connects them. This option has been added in case there are unaccounted words that should be placed in the middle of the graph. Figure 8-2 depicts the "toy_dog.n.01" and "maltese_dog.n.01" nodes. By clicking twice on the edge between them, a new node is generated. It is supposed to be more general than its hyponym "maltese_dog.n.01" and narrower than the word "toy_dog.n.01". However, we have not yet evaluated the performance of this specific type of node insertion. Consequently, we leave the application of our method for this subtask for further research.

8.2 Conclusion

The growing popularity of taxonomies in different research and industry tasks has created the need for a tree-like taxonomic subgraphs visualization platform. **TaxFree** provides such a platform for the visualization and analysis of hypo-hypernymy subgraphs. The tool allows users to explore wordnet synsets in context and predict new synsets for the leaf nodes. Our work aims to bring taxonomies to a broader audience, by making the WordNet interface more user-friendly than the standard WordNet[7] visualization. On this note, we conclude our thesis in the next chapter.

[7] http://wordnet.princeton.edu

Chapter 9

Conclusion

In this last chapter, we discuss the results, the limitations of our work, and provide an outlook on future research.

9.1 Conclusions

In this thesis, we performed a large-scale computational study of various methods for Taxonomy Enrichment. We also presented datasets for studying the diachronic evolution of wordnets for English and Russian. In contrast to the existing resources, our datasets mimic the real-world taxonomy extension scenario when no information about the new words is available.

We presented a new Taxonomy Enrichment method called DWRank. This method combines distributional information and information extracted from Wiktionary outperforming the baseline method (candidates retrieved from fastText nearest neighbour list and ranked with features extracted from Wiktionary) on the English datasets. We also presented its extensions: DWRank-Graph and DWRank-Meta, which use graph- and meta-embeddings via a common interface. We tested word2vec, fastText, GloVe, Poincaré, node2vec, HOPE and TADW embeddings along with GCN and GAT graph neural networks to predict hypernym synsets for the input word.

Additionally, we explored the benefits of meta-embeddings (combinations of embeddings) and graph embeddings for the task of Taxonomy Enrichment. On the

Russian datasets, DWRank-Meta performed best using fastText and word2vec word embeddings. For the English dataset, the combination of word (fastText, word2vec and GloVe) and graph (TADW) embeddings demonstrated the best performance. 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.

Error analysis on Taxonomy Enrichment with predefined candidates reveals that the correct synsets identified by graph-based models are usually retrieved by the fastText-based model alone. This makes graph representations mostly irrelevant and redundant. Nonetheless, there exist cases where graph representations were able to correctly identify some hypernyms that are not captured by fastText.

In this research, we also presented a novel candidate-free task formulation for taxonomy enrichment. We performed a computational study of various methods of identifying candidates for a taxonomy using knowledge from BERT. We compared different graph-based embeddings on the task of hypernym prediction and projected them to the BERT vector space. Then we identified the best position for the projected graph embedding to be injected to the BERT model. The results demonstrate that incorporation of graph embedding to the [MASK] embedding (at both "replace" and "mix" strategies) is beneficial in the task of hyponym prediction using contextualized models like BERT. The proposed task proven to be challenging for future research.

To sum up the section and this thesis in general, we provide answers to the research questions we stated in Section 1.3.

R1: Is it possible to predict hypernyms for new words from an existing taxonomy? It is definitely possible to predict the correct hypernyms for the new words from an existing taxonomy even without definition, but with the help of additional sources, e.g. word distributional information. According to our experiments,

word vector representations are a simple, powerful, and extremely effective tool for Taxonomy Enrichment, as the contexts (in a broad sense) extracted from pre-trained word embeddings (fastText, word2vec, GloVe) and their combination are sufficient to find hypernyms of new words in a taxonomy.

R2: Are graph-based embeddings beneficial for attaching new words to the taxonomy? Graph embeddings are not so much useful and efficient for the Taxonomy Enrichment task as expected. However, in combination with the textual representations (contextualized or non-contextualized) demonstrate SOTA results for both standard and candidate-free task formulations. The majority of graph embeddings implemented during this work do not perform well on the task yielding word vector representations. However, word embeddings still benefit from a combination with any type of graph embeddings. The best improvement on Taxonomy Enrichment is achieved when combined with TADW vector representations.

R3: Is it possible to predict new nodes in a taxonomy using a pre-trained language model like BERT? Taxonomy Enrichment seems to be quite a challenging task when the predefined list of new words is not given. Even though BERT managed to correctly identify some words absent in the taxonomy, the overall results are still poor. The pre-trained model still struggles to suggest the appropriate words for a given place in a taxonomy graph. Another problem is the multi-word prediction, which can be solved using other sophisticated models like GPT or T5 [Raffel et al., 2020], which could be the topic of further research.

9.2 Publicly Available Code, Datasets, and Models

Here are the links to the publicly available code:

- for the RUSSE-2020 competition: https://github.com/dialogue-evaluation/taxonomy-enrichment,
- for the Taxonomy Enrichment methods (with query words): https://github.com/skoltech-nlp/diachronic-wordnets,

- for candidate-free Taxonomy Enrichment: https://github.com/skoltech-nlp/taxgen,
- for the dataset: https://doi.org/10.5281/zenodo.4279821.

9.3 Future Directions

Despite the inconsistent results of graph-based methods, we propose a further exploration of the graph-based features as this resource contains principally different and complementary information to the distributional signal contained in text corpora. One potential way to improve the performance of the graph features is to use more sophisticated non-linear projection transformations from word to graph embeddings. Another promising direction is to explore other types of meta-embeddings to combine word and graph signals, e.g. GraphGlove [Ryabinin et al., 2020]. Moreover, we find it promising to experiment with temporal embeddings such as those of Goel et al. [2020] for the Taxonomy Enrichment task.

Another possible direction for further research is to develop a technology that would allow us to represent knowledge as a combination of textual, image and graph data. We aim at developing a fusion model based on the Transformer architecture. In principle, it would be able to predict missing representations of taxonomic objects as well as predict new objects that are not yet included in the taxonomy.

Overall, we set the following goals for future work:

- develop technology for multimodal taxonomy representation as well as for automatic taxonomy extension without predefined candidates;
- analyze the efficiency of the fusion of text, image and graph vector representations for the Taxonomy Enrichment task;
- apply the developed methods to both English and Russian taxonomies (WordNet and RuWordNet);
- improve on the quality of candidate-free TE by implementing more sophisticated pre-trained language models like Brown et al. [2020] and experimenting with different existing graph embeddings.

Our multimodal task formulation is fully novel, as there has been no previous work considering the Taxonomy Enrichment task from the multimodal perspective.

Bibliography

Matteo Abrate and Clara Bacciu. Visualizing word senses in WordNet atlas. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2648–2652, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/831_Paper.pdf.

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/W13-3520.

Rami Aly, Shantanu Acharya, Alexander Ossa, Arne Köhn, Chris Biemann, and Alexander 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*, pages 4811–4817, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1474. URL https://aclanthology.org/P19-1474.

Saba Anwar, Artem Shelmanov, Alexander Panchenko, and Chris Biemann. Generating lexical representations of frames using lexical substitution. In *Proceedings of the Probability and Meaning Conference (PaM 2020)*, pages 95–103, Gothenburg, June 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.pam-1.13.

Nikolai Arefyev, Alexander Panchenko, Artem Lukanin, Oleg Lesota, and Pavel Romanov. Evaluating three corpus-based semantic similarity systems for russian. *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue-2015"*, 28, 2015.

Nikolay Arefyev, Maksim Fedoseev, Andrey Kabanov, and Vadim 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. URL https://www.dialog-21.ru/media/5192/arefyevnvplusetal-152.pdf.

Mohit Bansal, Kevin Gimpel, and Karen 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)*, pages 809–815, 2014.

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. In *Proceedings of the international AAAI conference on web and social media*, volume 3, pages 361–362, 2009.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003. URL http://jmlr.org/papers/v3/bengio03a.html.

Gábor Berend, Márton Makrai, and Péter Földiák. 300-sparsans at SemEval-2018 task 9: Hypernymy as interaction of sparse attributes. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 928–934, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/S18-1152. URL https://aclanthology.org/S18-1152.

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

Gabriel Bernier-Colborne and Caroline Barrière. CRIM at SemEval-2018 task 9: A hybrid approach to hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 725–731, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/S18-1116. URL https://aclanthology.org/S18-1116.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi:10.1162/tacl_a_00051. URL https://aclanthology.org/Q17-1010.

Danushka Bollegala and Cong Bao. Learning word meta-embeddings by autoencoding. In *Proceedings of the 27th international conference on computational linguistics*, pages 1650–1661, 2018.

Francis Bond, Piek Vossen, John McCrae, and Christiane Fellbaum. CILI: the collaborative interlingual index. In *Proceedings of the 8th Global WordNet Conference (GWC)*, pages 50–57, Bucharest, Romania, 27–30 January 2016. Global Wordnet Association. URL https://aclanthology.org/2016.gwc-1.9.

Georgeta Bordea, Paul Buitelaar, Stefano Faralli, and Roberto Navigli. SemEval-2015 task 17: Taxonomy extraction evaluation (TExEval). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 902–910, Denver, Colorado, June 2015. Association for Computational Linguistics. doi:10.18653/v1/S15-2151. URL https://aclanthology.org/S15-2151.

Georgeta Bordea, Els Lefever, and Paul Buitelaar. SemEval-2016 task 13: Taxonomy extraction evaluation (TExEval-2). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1081–1091, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/S16-1168. URL https://aclanthology.org/S16-1168.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Jose Camacho-Collados. Why we have switched from building full-fledged taxonomies to simply detecting hypernymy relations. *arXiv preprint arXiv:1703.04178*, 2017.

Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa-Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. SemEval-2018 task 9: Hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 712–724, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/S18-1115. URL https://aclanthology.org/S18-1115.

William B Carper and William E Snizek. The nature and types of organizational taxonomies: An overview. *Academy of management review*, 5(1):65–75, 1980.

Yejin Cho, Juan Diego Rodriguez, Yifan Gao, and Katrin Erk. Leveraging WordNet paths for neural hypernym prediction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3007–3018, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi:10.18653/v1/2020.coling-main.268. URL https://aclanthology.org/2020.coling-main.268.

Shaika Chowdhury, Chenwei Zhang, Philip S Yu, and Yuan Luo. Mixed pooling multi-view attention autoencoder for representation learning in healthcare. *arXiv preprint arXiv:1910.06456*, 2019.

Philipp Cimiano and Johanna Völker. Text2onto: A framework for ontology learning and data-driven change discovery. In *Proceedings of the 10th International Conference on Natural Language Processing and Information Systems*, NLDB'05, page 227–238, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3540260315. doi:10.1007/11428817_21. URL https://doi.org/10.1007/11428817_21.

Joshua Coates and Danushka Bollegala. Frustratingly easy meta-embedding–computing meta-embeddings by averaging source word embeddings. *arXiv preprint arXiv:1804.05262*, 2018.

Christopher Collins. Wordnet explorer: applying visualization principles to lexical semantics. *Computational Linguistics Group, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada*, 2006.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 160–167. ACM, 2008. doi:10.1145/1390156.1390177. URL https://doi.org/10.1145/1390156.1390177.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747.

David 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.

Giuseppe De Giacomo and Maurizio Lenzerini. Tbox and abox reasoning in expressive description logics. *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR'96)*, 1996:37–48, 10 1996.

Daryna Dementieva, Daniil Moskovskiy, Varvara Logacheva, David Dale, Olga Kozlova, Nikita Semenov, and Alexander Panchenko. Methods for detoxification of texts for the russian language. *CoRR*, abs/2105.09052, 2021. URL https://arxiv.org/abs/2105.09052.

Daryna Dementieva, **Irina Nikishina**, Varvara Logacheva, Alena Fenogenova, David Dale, Irina Krotova, Nikita Semenov, Tatiana Shavrina, and Alexander Panchenko. Russe-2022: Findings of the first russian detoxification task based on parallel corpora. In *Computational Linguistics and Intellectual Technologies*, 2022. URL https://www.dialog-21.ru/media/5755/dementievadplusetal105.pdf.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi:10.1109/CVPR.2009.5206848.

Danilo Dessì, Francesco Osborne, Diego Reforgiato Recupero, Davide Buscaldi, and Enrico Motta. Generating knowledge graphs by employing natural language processing and machine learning techniques within the scholarly domain. *Future Generation Computer Systems*, 116:253–264, 2021. ISSN 0167-739X. doi:https://doi.org/10.1016/j.future.2020.10.026. URL https://www.sciencedirect.com/science/article/pii/S0167739X2033003X.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Tatyana Efremova. New dictionary of the russian language. *Explanatory-derivational. Moscow: Russky yazyk*, 2000.

John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. Graphviz—open source graph drawing tools. In *International Symposium on Graph Drawing*, pages 483–484. Springer, 2001.

Luis Espinosa-Anke, Francesco Ronzano, and Horacio 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)*, pages 1332–1336, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/S16-1208. URL https://aclanthology.org/S16-1208.

Allyson Ettinger. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48, 2020. doi:10.1162/tacl_a_00298. URL https://aclanthology.org/2020.tacl-1.3.

Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 271–276, Gothenburg, Sweden, May 2017. Association for Computational Linguistics. URL https://aclanthology.org/W17-0237.

Robert M French. The turing test: the first 50 years. *Trends in cognitive sciences*, 4 (3):115–122, 2000.

Zheng Gao, Gang Fu, Chunping Ouyang, Satoshi Tsutsui, Xiaozhong Liu, Jeremy Yang, Christopher Gessner, Brian Foote, David Wild, Ying Ding, and Qi Yu. edge2vec: Representation learning using edge semantics for biomedical knowledge discovery. *BMC Bioinformatics*, 20, 06 2019. doi:10.1186/s12859-019-2914-2.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. Diachronic embedding for temporal knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3988–3995, Apr. 2020. doi:10.1609/aaai.v34i04.5815. URL https://ojs.aaai.org/index.php/AAAI/article/view/5815.

Asunción Gómez-Pérez and Oscar Corcho. Ontology languages for the semantic web. *IEEE Intelligent systems*, 17(1):54–60, 2002.

Joseph E Gonzalez, Reynold S Xin, Ankur Dave, Daniel Crankshaw, Michael J Franklin, and Ion Stoica. {GraphX}: Graph processing in a distributed dataflow framework. In *11th USENIX symposium on operating systems design and implementation (OSDI 14)*, pages 599–613, 2014.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016. URL http://arxiv.org/abs/1607.00653.

Caglar Gülcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, R. Pascanu, K. Hermann, P. Battaglia, V. Bapst, D. Raposo, Adam Santoro, and N. D. Freitas. Hyperbolic attention networks. *arXiv preprint arXiv:1805.09786*, 2019.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

Kanyao Han, Pingjing Yang, Shubhanshu Mishra, and Jana Diesner. Wikicssh: extracting computer science subject headings from wikipedia. In *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, pages 207–218. Springer, 2020.

Michael Hanna and David Mareček. Analyzing BERT's knowledge of hypernymy via prompting. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 275–282, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi:10.18653/v1/2021.blackboxnlp-1.20. URL https://aclanthology.org/2021.blackboxnlp-1.20.

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, 1992. URL https://aclanthology.org/C92-2082.

Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi:10.1016/0893-6080(89)90020-8. URL https://doi.org/10.1016/0893-6080(89)90020-8.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 95–105, Beijing, China, July 2015. Association for Computational Linguistics. doi:10.3115/v1/P15-1010. URL https://aclanthology.org/P15-1010.

Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2186–2195. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/ivanov18a.html.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1356. URL https://aclanthology.org/P19-1356.

Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. Node similarity preserving graph convolutional networks. In Liane Lewin-Eytan, David Carmel, Elad Yom-Tov, Eugene Agichtein, and Evgeniy Gabrilovich, editors, *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 148–156. ACM, 2021. doi:10.1145/3437963.3441735. URL https://doi.org/10.1145/3437963.3441735.

David Jurgens and Mohammad Taher Pilehvar. SemEval-2016 task 14: Semantic taxonomy enrichment. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1092–1102, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/S16-1169. URL https://aclanthology.org/S16-1169.

Jaap Kamps and Maarten Marx. Visualizing wordnet structure. In *Proc. of the 1st International Conference on Global WordNet*, pages 182–186, 2002.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/P07-2045.

Evgeny Kotelnikov, Natalia Loukachevitch, **Irina Nikishina**, and Alexander Panchenko. RuArg-2022: Argument Mining Evaluation. In *Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue"*, 2022. URL https://www.dialog-21.ru/media/5773/kotelnikoveplusetal111.pdf.

H. Kucera and W. N. Francis. *Computational analysis of present-day American English*. Brown University Press, Providence, RI, 1967.

Maria Kunilovskaya, Andrey Kutuzov, and Alister Plum. Taxonomy Enrichment: Linear Hyponym-Hypernym Projection vs Synset ID Classification. In *Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue"*, 2020.

Yuri Kuratov and Mikhail Arkhipov. Adaptation of deep bidirectional multilingual transformers for russian language. *arXiv preprint arXiv:1905.07213*, 2019.

Andrey Kutuzov and Elizaveta Kuzmenko. *WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models*, pages 155–161. Springer International Publishing, Cham, 2017. ISBN 978-3-319-52920-2. doi:10.1007/978-3-319-52920-2_15. URL http://dx.doi.org/10.1007/978-3-319-52920-2_15.

Andrey Kutuzov and **Irina Nikishina**. Double-blind peer-reviewing and inclusiveness in russian nlp conferences. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 3–8. Springer, 2019. URL https://link.springer.com/chapter/10.1007/978-3-030-37334-4_1.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Helen Langone, Benjamin R. Haskell, and George A. Miller. Annotating WordNet. In *Proceedings of the Workshop Frontiers in Corpus Annotation at HLT-NAACL 2004*, pages 63–69, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-2710.

Michael E. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *SIGDOC '86*, 1986.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, 2015.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics, 2020. doi:10.18653/v1/2020.acl-main.703. URL https://doi.org/10.18653/v1/2020.acl-main.703.

Ninghao Liu, Xiao Huang, Jundong Li, and Xia Hu. On interpretation of network embedding via taxonomy induction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1812–1820, 2018.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Varvara Logacheva, Daryna Dementieva, Irina Krotova, Alena Fenogenova, **Irina Nikishina**, Tatiana Shavrina, and Alexander Panchenko. A study on manual and automatic evaluation for text style transfer: The case of detoxification. In *Proceedings of the 2nd Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 90–101, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.humeval-1.8. URL https://aclanthology.org/2022.humeval-1.8.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. URL http://arxiv.org/abs/1711.05101.

Natalia V Loukachevitch, German Lashevich, Anastasia A Gerasimova, Vladimir V Ivanov, and Boris V Dobrov. Creating russian wordnet by conversion. In *Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue*, pages 405–415, 2016.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1360. URL https://aclanthology.org/D18-1360.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi:10.18653/v1/D15-1166. URL https://aclanthology.org/D15-1166.

Ilya Makarov, Mikhail Makarov, and Dmitrii Kiselev. Fusion of text and graph information for machine learning problems on networks. *PeerJ Computer Science*, 7, 2021.

Alfredo Maldonado and Filip 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*, pages 924–927, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/S18-1151. URL https://aclanthology.org/S18-1151.

Deborah L McGuinness and Alexander Borgida. Explaining subsumption in description logics. *IJCAI (1)*, 3, 1995.

Bridget McInnes. VCU at Semeval-2016 task 14: Evaluating definitional-based similarity measure for semantic taxonomy enrichment. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1351–1355, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/S16-1212. URL https://aclanthology.org/S16-1212.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013a. URL http://arxiv.org/abs/1301.3781.

Tomás Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013b. URL http://arxiv.org/abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013c.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

George A Miller. Nouns in wordnet. *WordNet: An electronic lexical database*, pages 23–46, 1998a.

George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998b.

Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *CoRR*, abs/1707.05005, 2017. URL http://arxiv.org/abs/1707.05005.

Roberto Navigli and Simone Paolo Ponzetto. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 216–225, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://aclanthology.org/P10-1023.

James O' Neill and Danushka Bollegala. Meta-embedding as auxiliary task regularization. *arXiv preprint arXiv:1809.05886*, 2018.

Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6341–6350. Curran Associates, Inc., 2017.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2014–2023, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/niepert16.html.

Francesco Osborne and Enrico Motta. Klink-2: Integrating multiple web sources to generate semantic topic networks. In *Lecture Notes in Computer Science*, volume 9366, 10 2015. ISBN 978-3-319-25006-9. doi:10.1007/978-3-319-25007-6_24.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998. URL citeseer.nj.nec.com/page98pagerank.html.

Alexander Panchenko, Dmitry Ustalov, Nikolay Arefyev, Denis Paperno, Natalia Konstantinova, Natalia Loukachevitch, and Chris Biemann. Human and machine judgements for russian semantic relatedness. In *International conference on analysis of images, social networks and texts*, pages 221–235. Springer, 2016.

Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 113–120, Sydney, Australia, July 2006. Association for Computational Linguistics. doi:10.3115/1220175.1220190. URL https://aclanthology.org/P06-1015.

Ted Pedersen. Duluth at SemEval 2016 task 14: Extending gloss overlaps to enrich semantic taxonomies. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1328–1331, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/S16-1207. URL https://aclanthology.org/S16-1207.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke 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)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Joel 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)*, pages 1298–1302, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/S16-1202. URL https://aclanthology.org/S16-1202.

Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. doi:10.1108/eb046814. URL https://doi.org/10.1108/eb046814.

Dhruba Pujary, Camilo Thorne, and Wilker Aziz. Disease normalization with graph embeddings. In *Proceedings of SAI Intelligent Systems Conference*, pages 209–217. Springer, 2020.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Abhilasha Ravichander, Eduard Hovy, Kaheer Suleman, Adam Trischler, and Jackie Chi Kit Cheung. On the systematicity of probing contextualized word representations: The case of hypernymy in BERT. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 88–102, Barcelona, Spain (Online), December 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.starsem-1.10.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020. doi:10.1162/tacl_a_00349. URL https://aclanthology.org/2020.tacl-1.54.

Andrea Rossi, Donatella Firmani, Antonio Matinata, Paolo Merialdo, and Denilson Barbosa. Knowledge graph embedding for link prediction: A comparative analysis. *arXiv preprint arXiv:2002.00819*, 2020.

Jon Rusert and Ted Pedersen. UMNDuluth at SemEval-2016 task 14: WordNet's missing lemmas. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1346–1350, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/S16-1211. URL https://aclanthology.org/S16-1211.

Max Ryabinin, Sergei Popov, Liudmila Prokhorenkova, and Elena Voita. Embedding words in non-vector space with unsupervised graph learning. *arXiv preprint arXiv:2010.02598*, 2020.

Kristina Sabirova and Artem Lukanin. Automatic extraction of hypernyms and hyponyms from russian texts. In Dmitry I. Ignatov, Mikhail Yu. Khachay, Alexander Panchenko, Natalia Konstantinova, Rostislav E. Yavorsky, and Dmitry Ustalov, editors, *Supplementary Proceedings of the 3rd International Conference on Analysis of Images, Social Networks and Texts (AIST'2014), Yekaterinburg, Russia, April 10-12, 2014*, volume 1197 of *CEUR Workshop Proceedings*, pages 35–40. CEUR-WS.org, 2014. URL http://ceur-ws.org/Vol-1197/paper6.pdf.

Anna Safaryan, Petr Filchenkov, Weijia Yan, Andrey Kutuzov, and **Irina Nikishina**. Semantic recommendation system for bilingual corpus of academic papers. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 22–36. Springer, 2020. URL https://link.springer.com/chapter/10.1007/978-3-030-71214-3_3.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi:10.1109/TNN.2008.2005605.

Timo Schick and Hinrich Schütze. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. *CoRR*, abs/1904.06707, 2019. URL http://arxiv.org/abs/1904.06707.

Dominik Schlechtweg, Barbara McGillivray, Simon Hengchen, Haim Dubossarsky, and Nina Tahmasebi. SemEval-2020 task 1: Unsupervised lexical semantic change detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1–23, Barcelona (online), December 2020. International Committee for Computational Linguistics. doi:10.18653/v1/2020.semeval-1.1. URL https://aclanthology.org/2020.semeval-1.1.

Michael Schlichtkrull and Héctor 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)*, pages 1337–1341, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/S16-1209. URL https://aclanthology.org/S16-1209.

Holger Schwenk. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of COLING 2012: Posters*, pages 1071–1080, Mumbai, India, December 2012. The COLING 2012 Organizing Committee. URL https://aclanthology.org/C12-2104.

Jiaming Shen, Zhihong Shen, Chenyan Xiong, Chi Wang, Kuansan Wang, and Jiawei Han. Taxoexpan: Self-supervised taxonomy expansion with position-enhanced graph neural network. In *Proceedings of The Web Conference 2020*, pages 486–497, 2020.

Vered Shwartz, Yoav Goldberg, and Ido 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)*, pages 2389–2398, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1226. URL https://aclanthology.org/P16-1226.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 801–808, Sydney, Australia, July 2006. Association for Computational Linguistics. doi:10.3115/1220175.1220276. URL https://aclanthology.org/P06-1101.

Milan Straka and Jana 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*, pages 88–99, Vancouver, Canada, August 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/K/K17/K17-3009.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27:3104–3112, 2014.

Hristo Tanev and Agata Rotondi. Deftor at SemEval-2016 task 14: Taxonomy enrichment using definition vectors. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1342–1345, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/S16-1210. URL https://aclanthology.org/S16-1210.

Irina Nikishina, Varvara Logacheva, Alexander Panchenko, and Natalia 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"*, 2020a. URL https://www.dialog-21.ru/media/5111/nikishinaiplusetal-160.pdf.

Irina Nikishina, Varvara Logacheva, Alexander Panchenko, and Natalia Loukachevitch. Studying taxonomy enrichment on diachronic WordNet versions. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3095–3106, Barcelona, Spain (Online), December 2020b. International Committee on Computational Linguistics. doi:10.18653/v1/2020.coling-main.276. URL https://aclanthology.org/2020.coling-main.276.

Irina Nikishina, Natalia Loukachevitch, Varvara Logacheva, and Alexander Panchenko. Evaluation of taxonomy enrichment on diachronic WordNet versions. In *Proceedings of the 11th Global Wordnet Conference*, pages 126–136, University of South Africa (UNISA), January 2021. Global Wordnet Association. URL https://aclanthology.org/2021.gwc-1.15.

Irina Nikishina, Mikhail Tikhomirov, Varvara Logacheva, Yuriy Nazarov, Alexander Panchenko, and Natalia Loukachevitch. Taxonomy enrichment with text and graph vector representations. *Semantic Web*, 13(6):441–475, 02 2022. doi:10.3233/SW-212955. URL https://content.iospress.com/download/semantic-web/sw212955?id=semantic-web%2Fsw212955.

Mikhail Tikhomirov and Natalia Loukachevitch. Meta-embeddings in taxonomy enrichment task. In *Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue"*, 2021.

Mikhail Tikhomirov, Natalia Loukachevitch, and Ekaterina Parkhomenko. Combined Approach to Hypernym Detection for Thesaurus Enrichment. In *Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue"*, 2020.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://aclanthology.org/P10-1040.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Paola Velardi, Stefano Faralli, and Roberto Navigli. OntoLearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707, September 2013. doi:10.1162/COLI_a_00146. URL https://aclanthology.org/J13-3007.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

Changping Wang, Chaokun Wang, Zheng Wang, Xiaojun Ye, and Philip S. Yu. Edge2vec: Edge-based social network embedding. *ACM Trans. Knowl. Discov. Data*, 14(4), may 2020. ISSN 1556-4681. doi:10.1145/3391298. URL https://doi.org/10.1145/3391298.

Genta Indra Winata, Zhaojiang Lin, and Pascale Fung. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 181–186, 2019.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan

Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. Network representation learning with rich text information. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html.

Wenpeng Yin and Hinrich Schütze. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1351–1360, 2016.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Comput. Intell. Mag.*, 13(3):55–75, 2018. doi:10.1109/MCI.2018.2840738. URL https://doi.org/10.1109/MCI.2018.2840738.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-BERT: Only attention is needed for learning graph representations. *CoRR*, abs/2001.05140, 2020. URL https://arxiv.org/abs/2001.05140.

Appendix A

Ranked Examples and All Results

In this Appendix, we provide Tables A.1, A.2 and A.3 with the examples that demonstrate the input and the output formats of the models as well as Tables A.6 and A.7 that show the performance of all models for both English and Russian datasets.

From both tables A.1 and A.2 (underlined bold text denotes predictions of the model from the ground truth), we can see that at least one of the correct candidates usually appears in the list of candidates from word embeddings (first part of the table), whereas among candidates from graph embeddings we do not see any decent synsets. Poincaré embeddings retrieved by aggregating words from fastText provide too broad concepts which are clearly too far from the correct answers ("activity.n.01", "exposure.n.03", "action.n.02"). Node2vec embeddings are both semantically far and abstract. GraphSAGE is sticking to the word "play" and is too far from the correct answers in general. TADW manages to predict the correct synset "therapy.n.01" in the list of candidates, however, its position is much lower than the positions of the same synsets among the candidates provided by word embeddings-based systems.

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

DWRank-Meta on both word and graph embeddings may provide the results that improve the ranking, e.g. "therapy.n.01" is the correct candidate at the first place in the list for both CAEME triple loss (fastText, word2vec and GloVe) approach

and for the AAEME (fastText, word2vec, Glove, TASDW) approach. For the word "eyewitness" the DWRank-Meta models on words and graphs the true candidates are placed in the worse positions, however, they still win before the DWRank-GRaph approach.

We also provide several examples of the new words correctly attached to the taxonomy graph visualized by means of our demo visualization tool TaxFree in Figures A-1, A-2, A-3, A-4.

play therapy psychotherapy.n.02, therapy.n.01			
fastText (baseline)	fastText (DWRank)	AAEME triplet loss (fastText + word2vec + GloVe)	AAEME (fastText + word2vec + GloVe + TADW)
play.n.03	play.n.03	**therapy.n.01**	**therapy.n.01**
play.n.01	activity.n.01	activity.n.01	medical_care.n.01
baseball_play.n.01	plan_of_action.n.01	medical_care.n.01	play.n.03
therapy.n.01	**therapy.n.01**	diversion.n.01	play.n.01
activity.n.01	play.n.01	play.n.01	activity.n.01
diversion.n.01	dramatic_composition.n.01	act.n.02	dramatic_composition.n.01
plan_of_action.n.01	outdoor_game.n.01	**psychotherapy.n.02**	plan_of_action.n.01
action.n.01	medical_care.n.01	play.n.03	show.n.04
action.n.02	golf.n.01	dramatic_composition.n.01	treatment.n.01
dramatic_composition.n.01	diversion.n.01	behaviour_therapy.n.01	act.n.02
GraphSAGE	TADW	Poincaré	node2vec
baseball_play.n.01	play.n.03	activity.n.01	presentation.n.03
play.n.03	play.n.01	exposure.n.03	presentation.n.01
play.n.01	plan_of_action.n.01	rejection.n.01	operation.n.01
squeeze_play.n.02	dramatic_composition.n.01	agreement.n.06	performance.n.02
activity.n.01	activity.n.01	light_unit.n.01	contact.n.01
diversion.n.01	baseball_play.n.01	blessing.n.01	exposure.n.03
dramatic_composition.n.01	show.n.04	vulnerability.n.02	activity.n.01
attempt.n.01	diversion.n.01	action.n.02	exposure.n.08
plan_of_action.n.01	use.n.01	influence.n.02	union.n.04
play.n.17	action.n.02	assent.n.01	exposure.n.06

Table A.1: Examples of predictions for nouns from the English v 1.6-3.0 dataset with various models.

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

Table A.2: Examples of predictions for nouns from the English v 1.6-3.0 dataset with various models.

eyewitness witness.v.01, watch.v.01			
fastText (baseline)	fastText (DWRank)	AAEME triplet loss (fastText + word2vec + GloVe)	AAEME (fastText + word2vec + GloVe + TADW)
be.v.01	inform.v.01	inform.v.01	inform.v.01
be.v.03	testify.v.02	testify.v.02	testify.v.02
be.v.08	communicate.v.02	communicate.v.02	confirm.v.02
watch.v.01	announce.v.01	see.v.10	communicate.v.02
testify.v.01	confirm.v.02	confirm.v.02	**watch.v.01**
testify.v.02	**watch.v.01**	**watch.v.01**	be.v.01
man.v.02	testify.v.01	witness.v.02	affirm.v.03
talk.v.01	affirm.v.03	affirm.v.03	testify.v.01
guard.v.01	report.v.03	testify.v.01	reject.v.01
confirm.v.01	record.v.01	verify.v.01	experience.v.01
graphSAGE	TADW	Poincaré	node2vec
see.v.05	inform.v.01	confirm.v.02	affirm.v.03
testify.v.02	testify.v.02	examine.v.02	confirm.v.02
err.v.01	communicate.v.02	affirm.v.03	understand.v.02
confirm.v.01	declare.v.01	testify.v.02	uphold.v.03
pronounce.v.02	announce.v.01	inform.v.01	determine.v.08
idealize.v.01	testify.v.01	justify.v.02	stay_in_place.v.01
judge.v.02	record.v.01	declare.v.01	justify.v.02
negate.v.03	**watch.v.01**	validate.v.03	fall_asleep.v.01
reason.v.01	report.v.03	uphold.v.03	resettle.v.01
disbelieve.v.01	report.v.01	testify.v.01	settle.v.04

Table A.3: Examples of predictions for verbs from the English v 1.6-3.0 dataset with various models.

theologise cover.v.05, broach.v.01, chew_over.v.01, think.v.03			
fastText (baseline)	fastText (DWRank)	AAEME triplet loss (fastText + word2vec + GloVe)	AAEME (fastText + word2vec + GloVe + TADW)
change.v.01	change.v.01	change.v.01	change.v.01
match.v.01	preface.v.01	preface.v.01	preface.v.01
date.v.03	make.v.03	convert.v.05	equal.v.03
reconstruct.v.01	date.v.03	match.v.01	date.v.03
determine.v.03	equal.v.03	chronologize.v.01	reconstruct.v.01
make.v.03	reason.v.01	reason.v.01	reason.v.01
speculate.v.01	chronologize.v.01	change_state.v.01	convert.v.05
convert.v.05	match.v.01	represent.v.09	formulate.v.03
reason.v.01	film.v.02	make.v.03	reflect.v.04
automatize.v.01	formulate.v.03	change.v.02	film.v.02
graphSAGE	TADW	Poincaré	node2vec
process.v.02	change.v.01	state.v.01	settle.v.04
affect.v.01	preface.v.01	speculate.v.01	stay_in_place.v.01
change.v.01	date.v.03	reason.v.01	understand.v.02
dive.v.01	film.v.02	preface.v.01	study.v.03
tame.v.01	reconstruct.v.01	match.v.01	resettle.v.01
sensitize.v.02	equal.v.03	generalize.v.01	fall_asleep.v.01
convert.v.01	convert.v.05	announce.v.02	speculate.v.01
estimate.v.01	formulate.v.03	express.v.02	discover.v.07
subject.v.01	reason.v.01	add.v.02	explicate.v.02
compound.v.05	commemorate.v.03	equal.v.01	behave.v.02

Table A.4: Examples of predictions for verbs from the English v 1.6-3.0 dataset with various models.

immunise protect.v.01, defend.v.02, inject.v.01, administer.v.04			
fastText (baseline)	fastText (DWRank)	AAEME triplet loss (fastText + word2vec + GloVe)	AAEME (fastText + word2vec + GloVe + TADW)
indoctrinate.v.01	indoctrinate.v.01	teach.v.01	**inject.v.01**
inject.v.01	**protect.v.01**	treat.v.01	remove.v.01
defend.v.02	**defend.v.02**	insert.v.02	inform.v.01
remove.v.01	teach.v.01	prevent.v.01	change.v.01
treat.v.01	prevent.v.02	isolate.v.01	better.v.02
destroy.v.01	**inject.v.01**	inoculate.v.01	insert.v.02
prevent.v.02	insert.v.02	indoctrinate.v.01	defend.v.02
insert.v.02	defend.v.01	kill.v.01	kill.v.01
teach.v.01	kill.v.01	change.v.01	indoctrinate.v.01
administer.v.04	discriminate.v.02	enable.v.01	protect.v.01
graphSAGE	TADW	Poincaré	node2vec
defend.v.02	**protect.v.01**	teach.v.01	receive.v.01
protect.v.01	indoctrinate.v.01	insert.v.02	insert.v.02
act.v.01	teach.v.01	inform.v.01	stay_in_place.v.01
negociate.v.01	**defend.v.02**	treat.v.01	get.v.01
attack.v.03	prevent.v.02	train.v.01	**inject.v.01**
prevent.v.02	**inject.v.01**	deceive.v.02	fall_asleep.v.01
insert.v.02	prevent.v.01	indoctrinate.v.01	accept.v.02
demilitarize.v.01	insert.v.02	misinform.v.01	settle.v.04
disarm.v.02	isolate.v.01	gull.v.02	resettle.v.01
foreswear.v.02	discriminate.v.02	interact.v.01	discover.v.07

Table A.5: Examples of predictions for verbs from the English v 1.6-3.0 dataset with various models.

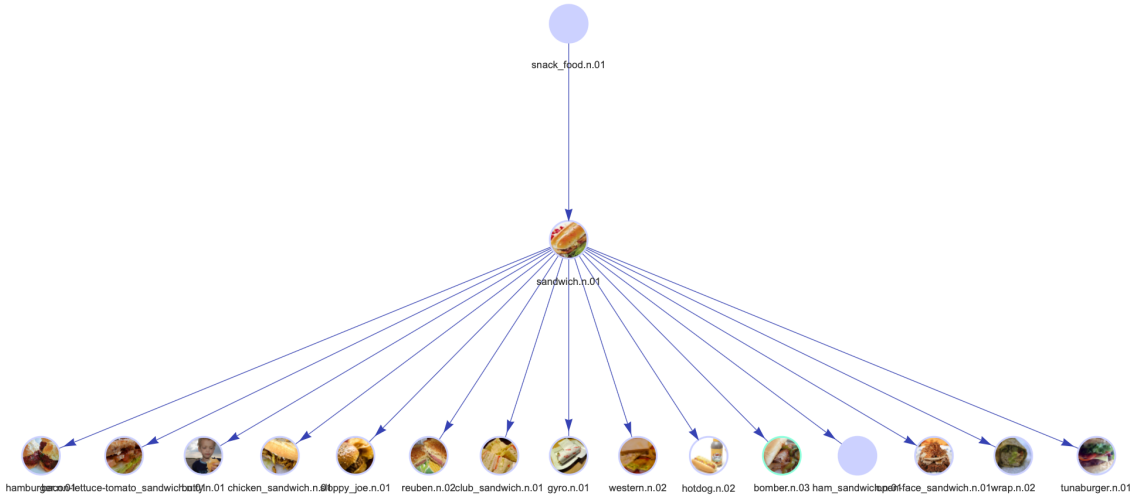


Figure A-1: Example of the correct attachment of the word "Cuban sandwich" (a large sandwich made of a long crusty roll split lengthwise and filled with meats and cheese (and tomato and onion and lettuce and condiments)) to the correct hypernym "sandwich.n.01"

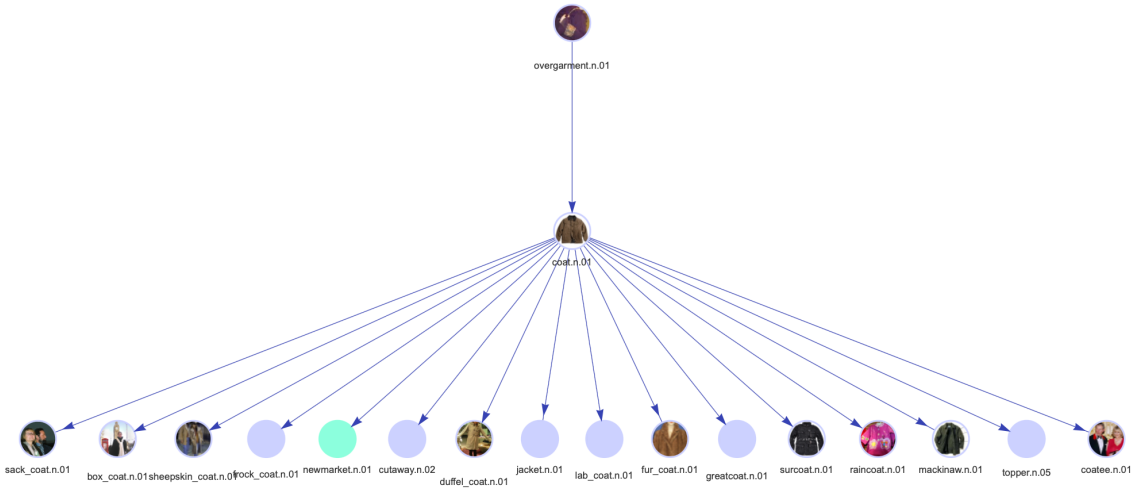


Figure A-2: Example of the correct attachment of the word "newmarket" (a long close-fitting coat worn for riding in the 19th century) to the correct hypernym "coat.n.01"

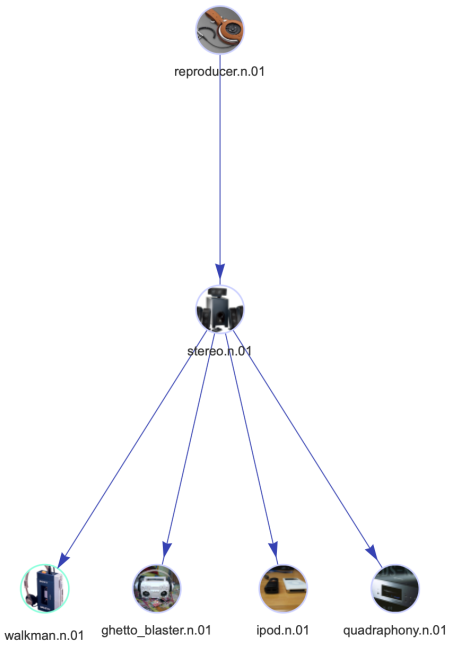


Figure A-3: Example of the correct attachment of the word “walkman” (a pocket-sized stereo system with light weight earphones) to the correct hypernym “stereo.n.01”

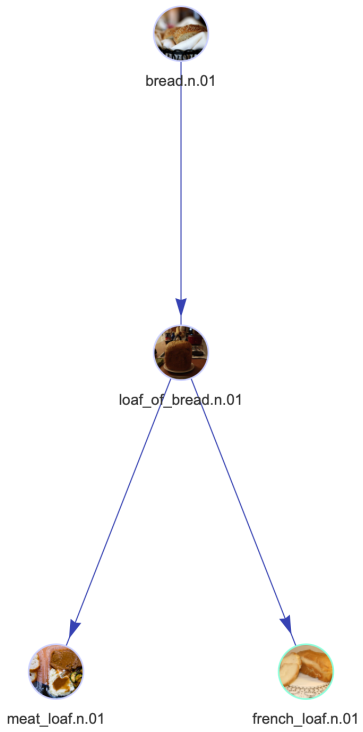


Figure A-4: Example of the correct attachment of the word “French loaf” (a loaf of French bread) to the correct hypernym “loaf_of_bread.n.01”

method	Nouns			Verbs		
	1.6-3.0	1.7-3.0	2.0-3.0	1.6-3.0	1.7-3.0	2.0-3.0
Baseline [Nikishina et al., 2020b]						
fastText [Bojanowski et al., 2017]	**0.338±0.002**	**0.371±0.002**	**0.400±0.004**	**0.270±0.007**	**0.203±0.010**	**0.236±0.011**
word2vec [Mikolov et al., 2013c]	0.142±0.001	0.178±0.002	0.164±0.004	0.229±0.006	0.155±0.008	0.212±0.009
GloVe [Pennington et al., 2014]	0.232±0.002	0.188±0.001	0.233±0.004	0.146±0.005	0.149±0.008	0.191±0.010
DWRank-Word						
fastText [Bojanowski et al., 2017]	**0.314±0.001**	**0.373±0.003**	**0.418±0.004**	**0.286±0.007**	**0.218±0.008**	**0.254±0.012**
word2vec [Mikolov et al., 2013c]	0.244±0.001	0.271±0.003	0.298±0.004	0.099±0.005	0.118±0.008	0.141±0.010
GloVe [Pennington et al., 2014]	0.283±0.001	0.329±0.003	0.377±0.004	0.182±0.007	0.159±0.008	0.203±0.011
DWRank-Meta (Meta-embeddings based on Word Embeddings)						
concat (*words*)	**0.335±0.001**	0.386±0.003	0.386±0.003	0.270±0.007	0.194±0.009	0.226±0.011
SVD (*words*)	0.333±0.001	**0.399±0.003**	**0.456±0.004**	0.277±0.007	0.209±0.010	0.264±0.012
CAEME*words*)	0.321±0.001	0.386±0.003	0.448±0.005	0.278±0.007	0.205±0.008	0.266±0.015
AAEME (*words*)	0.322±0.001	0.384±0.003	0.453±0.004	0.271±0.007	**0.218±0.008**	**0.273±0.012**
CAEME triplet loss (*words*)	0.332±0.001	0.394±0.003	0.451±0.004	0.273±0.007	0.205±0.007	0.256±0.013
AAEME triplet loss (*words*)	**0.335±0.001**	0.391±0.003	0.453±0.004	**0.280±0.008**	0.212±0.007	0.262±0.014
DWRank-Graph						
GCN [Kipf and Welling, 2017]	0.175±0.001	0.249±0.002	0.267±0.002	0.162±0.006	0.113±0.005	0.149±0.010
GAT [Velickovic et al., 2018]	0.000±0.000	0.252±0.002	0.000±0.000	0.081±0.003	0.064±0.004	0.000±0.000
GraphSAGE [Hamilton et al., 2017]	0.214±0.001	0.282±0.002	0.224±0.003	0.127±0.004	0.114±0.004	0.090±0.008
TADW [Yang et al., 2015] (on fastText)	**0.350±0.001**	**0.392±0.002**	**0.435±0.004**	**0.268±0.007**	**0.201±0.007**	**0.217±0.010**
Poincaré [Nickel and Kiela, 2017] (top-5 fastText associates)	0.185±0.001	0.211±0.002	0.229±0.002	0.208±0.006	0.147±0.006	0.172±0.012
node2vec [Grover and Leskovec, 2016] (top-5 fastText associates)	0.270±0.001	0.312±0.002	0.341±0.004	0.175±0.006	0.128±0.007	0.118±0.012
HOPE [Ou et al., 2016]	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
DWRank-Meta (Meta-embeddings based on Word and Graph Embeddings)						
SVD (*words* + node2vec)	0.343±0.001	0.383±0.003	0.434±0.005	0.272±0.006	0.194±0.009	0.239±0.011
CAEME (*words* + node2vec)	0.335±0.001	0.379±0.003	0.426±0.004	0.242±0.005	0.184±0.009	0.221±0.012
AAEME (*words* + node2vec)	0.350±0.001	0.394±0.003	0.446±0.004	0.252±0.007	0.184±0.008	0.208±0.012
SVD (*words* + TADW)	0.355±0.001	0.414±0.003	0.472±0.004	**0.288±0.007**	0.222±0.009	**0.280±0.013**
CAEME (*words* + TADW)	0.350±0.001	0.404±0.003	0.458±0.004	0.267±0.007	0.212±0.007	0.247±0.011
AAEME (*words* + TADW)	**0.367±0.001**	**0.418±0.002**	**0.480±0.004**	0.283±0.007	**0.227±0.007**	0.260±0.012
SVD (*words* + GCN)	0.323±0.001	0.385±0.003	0.443±0.004	0.260±0.005	0.209±0.009	0.249±0.011
CAEME (*words* + GCN)	0.331±0.001	0.395±0.003	0.457±0.004	0.251±0.006	0.207±0.009	0.235±0.012
AAEME (*words* + GCN)	0.331±0.001	0.392±0.003	0.456±0.004	0.243±0.006	0.200±0.008	0.228±0.012
SVD (*words* + GraphSAGE)	0.338±0.001	0.401±0.003	0.464±0.004	0.239±0.006	0.194±0.009	0.221±0.011
CAEME (*words* + GraphSAGE)	0.323±0.001	0.382±0.003	0.435±0.004	0.200±0.006	0.170±0.007	0.202±0.01
AAEME (*words* + GraphSAGE)	0.343±0.001	0.406±0.003	0.468±0.004	0.238±0.007	0.178±0.008	0.209±0.011
State-of-the-art Approaches						
WBSR (Top-1 RUSSE'2020 for nouns)	**0.333±0.002**	**0.393±0.003**	**0.436±0.003**	**0.252±0.006**	**0.206±0.011**	**0.252±0.013**
WBSR, no search engine features	0.251±0.001	0.309±0.003	0.344±0.004	0.231±0.006	0.180±0.008	0.222±0.009
hypo2path rev [Cho et al., 2020]	0.264±0.001	0.283±0.003	0.238±0.007	0.173±0.005	0.104±0.008	0.118±0.009
hypo2path [Cho et al., 2020]	0.252±0.002	0.261±0.002	0.208±0.006	0.162±0.005	0.093±0.006	0.067±0.008
hypo2path transformer	0.218±0.002	0.229±0.002	0.057±0.002	0.140±0.003	0.120±0.006	0.100±0.008
TaxoExpan [Shen et al., 2020]	*0.004±0.000*	*0.003±0.000*	*0.054±0.002*	*0.001±0.000*	*0.000±0.000*	*0.000±0.000*

Table A.6: MAP scores for the taxonomy enrichment methods for the English datasets. Numbers **in bold** show the best model within the category, **underlined** numbers denote the best score across all the models. The combination of word embeddings (fastText, word2vec, GloVe) is denoted as *words*.

method	nouns		verbs	
	non-restricted	restricted	non-restricted	restricted
Baseline				
fastText [Bojanowski et al., 2017]	**0.414±0.001**	**0.549±0.006**	0.296±0.004	0.389±0.011
word2vec [Mikolov et al., 2013c]	0.263±0.001	0.427±0.006	**0.343±0.004**	**0.445±0.013**
DWRank (Word Embeddings)				
fastText [Bojanowski et al., 2017]	**0.419±0.001**	**0.572±0.005**	**0.337±0.003**	**0.428±0.007**
word2vec [Mikolov et al., 2013c]	0.296±0.002	0.569±0.005	0.250±0.003	0.284±0.011
DWRank (Meta-embeddings based on Word Embeddings)				
concat (*words*)	0.422±0.001	0.589±0.005	0.351±0.004	0.426±0.009
SVD (*words*)	0.461±0.001	**0.600±0.005**	**0.426±0.005**	**0.475±0.010**
CAEME (*words*)	0.400±0.001	0.561±0.005	0.342±0.003	0.416±0.008
AAEME (*words*)	0.456±0.001	0.582±0.005	0.368±0.004	0.442±0.009
CAEME triplet loss (*words*)	0.449±0.001	0.581±0.005	0.374±0.003	0.427±0.010
AAEME triplet loss (*words*)	**0.474±0.001**	0.593±0.006	0.399±0.004	0.449±0.010
DWRank (Graph embeddings)				
GCN [Kipf and Welling, 2017]	0.183±0.001	0.306±0.005	0.220±0.003	0.287±0.009
GAT [Velickovic et al., 2018]	0.142±0.001	0.318±0.004	0.000±0.000	0.000±0.000
GraphSAGE [Hamilton et al., 2017]	0.176±0.001	0.348±0.005	0.181±0.003	0.226±0.008
TADW [Yang et al., 2015]	**0.417±0.001**	**0.562±0.005**	**0.328±0.003**	**0.423±0.008**
Poincaré [Nickel and Kiela, 2017] (top-5 fastText associates)	0.336±0.001	0.476±0.005	0.244±0.004	0.339±0.009
node2vec [Grover and Leskovec, 2016] (top-5 fastText associates)	0.343±0.002	0.477±0.005	0.226±0.003	0.322±0.010
HOPE [Ou et al., 2016]	0.000±0.000	0.000±0.000	0.003±0.001	0.003±0.001
DWRank (Meta-embeddings based on Word and Graph Embeddings)				
SVD (*words* + node2vec)	0.367±0.001	0.521±0.005	0.252±0.003	0.351±0.010
CAEME (*words* + node2vec)	0.370±0.001	0.533±0.005	0.267±0.003	0.362±0.010
AAEME (*words* + node2vec)	0.373±0.001	0.529±0.005	0.272±0.003	0.358±0.010
SVD (*words* + TADW)	**0.469±0.001**	**0.604±0.006**	**0.394±0.005**	**0.455±0.010**
CAEME (*words* + TADW)	0.429±0.001	0.571±0.005	0.349±0.003	0.437±0.009
AAEME (*words* + TADW)	0.461±0.001	0.584±0.005	0.362±0.004	0.439±0.009
SVD (*words* + GCN)	0.395±0.001	0.554±0.005	0.291±0.004	0.356±0.009
CAEME (*words* + GCN)	0.389±0.001	0.544±0.005	0.302±0.003	0.381±0.008
AAEME (*words* + GCN)	0.386±0.001	0.545±0.006	0.295±0.004	0.365±0.008
SVD (*words* + GraphSAGE)	0.410±0.001	0.603±0.005	0.336±0.004	0.426±0.009
CAEME (*words* + GraphSAGE)	0.321±0.001	0.541±0.005	0.266±0.004	0.345±0.007
AAEME (*words* + GraphSAGE)	0.409±0.001	0.577±0.006	0.323±0.004	0.419±0.009
State-of-the-art Approaches				
WBSR (Top-1 RUSSE'2020 for nouns)	**0.393±0.002**	**0.552±0.005**	0.293±0.004	0.428±0.010
WBSR, no search engine features	0.369±0.002	0.497±0.005	0.267±0.004	0.387±0.009
Top-1 RUSSE'2020 for verbs: [Dale, 2020]	0.288±0.001	0.418±0.006	**0.341±0.004**	**0.452±0.012**
hypo2path [Cho et al., 2020]	0.061±0.000	0.097±0.002	0.137±0.003	0.174±0.009
hypo2path rev [Cho et al., 2020]	0.246±0.001	0.342±0.006	0.151±0.003	0.194±0.008
hypo2path rev transformer [Cho et al., 2020]	0.234±0.001	0.331±0.004	0.152±0.003	0.201±0.008
TaxoExpan [Shen et al., 2020]	*0.007±0.000*	*0.006±0.001*	*0.009±0.001*	*0.008±0.002*

Table A.7: MAP scores for the taxonomy enrichment methods for the Russian datasets. Numbers **in bold** show the best model within the category, **underlined** numbers denote the best score across all the models.