

Skolkovo Institute of Science and Technology

*As a manuscript*

Alexander Ivanovich Panchenko

**Methods and Algorithms for Extraction, Linking, Vectorisation, and  
Disambiguation of Lexical-Semantic Graphs**

Dissertation Summary

for the purpose of obtaining academic degree  
Doctor of Science in Computer Science

Moscow – 2024

This doctoral dissertation was prepared at Skolkovo Institute of Science and Technology (Skoltech) and partially at Artificial Intelligence Research Institute (AIRI) based on publications prepared at University of Hamburg (UHH), Technical University of Darmstadt (TUDA), Skoltech, and AIRI. The earliest publication related to this dissertation dates back to 2016 while the latest one has been published in 2023. The current document provides a summary of scientific contributions.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Graph Clustering for Sense and Frame Induction</b>	<b>23</b>
2.1	Introduction . . . . .	23
2.2	Method . . . . .	24
2.3	Results . . . . .	30
<b>3</b>	<b>Word Sense Embeddings</b>	<b>32</b>
3.1	Introduction . . . . .	32
3.2	Method . . . . .	33
3.3	Results . . . . .	40
<b>4</b>	<b>Unsupervised Interpretable Word Sense Disambiguation</b>	<b>41</b>
4.1	Introduction . . . . .	41
4.2	Method . . . . .	43
4.3	Results . . . . .	45
<b>5</b>	<b>Linking Word Sense Representations</b>	<b>47</b>
5.1	Introduction . . . . .	47
5.2	Method . . . . .	48
5.3	Results . . . . .	52
<b>6</b>	<b>Prediction of Hypernym Embeddings</b>	<b>53</b>
6.1	Introduction . . . . .	53

6.2	Method . . . . .	54
6.3	Results . . . . .	55
<b>7</b>	<b>Extracting of Hypernyms via Sense Graph Clustering</b>	<b>56</b>
7.1	Introduction . . . . .	56
7.2	Method . . . . .	57
7.3	Results . . . . .	60
<b>8</b>	<b>Taxonomy Enrichment using Hyperbolic Embeddings</b>	<b>61</b>
8.1	Introduction . . . . .	61
8.2	Method . . . . .	62
8.3	Results . . . . .	65
<b>9</b>	<b>Node Embeddings of Lexical-Semantic Graphs</b>	<b>66</b>
9.1	Introduction . . . . .	66
9.2	Method . . . . .	67
9.3	Results . . . . .	69
<b>10</b>	<b>Lexical Substitution and Analysis of its Semantic Relation Types</b>	<b>70</b>
10.1	Introduction . . . . .	70
10.2	Method . . . . .	71
10.3	Results . . . . .	73
<b>11</b>	<b>Conclusion</b>	<b>75</b>
	<b>Bibliography</b>	<b>78</b>

# Chapter 1

## Introduction

The traditional lexical resources, such as Wordnets<sup>1</sup>, taxonomies, thesauri or dictionaries contain *precise* manually-encoded information about lexical items (such as words and phrases) and relations between them (such as synonyms and hypernyms) yet coverage i.e. *recall* and actuality of these resources is often inherently limited. This is due to the expensive, long and usually purely manual process of resource creation and keeping it up-to-date involving labor of lexicographers, linguists and domain experts. Besides, some domain-specific terms may be simply out of scope even for the largest lexicographical collaboratively created linguistic resources, such as Wiktionary.<sup>2</sup>

On the other hand, it is possible to apply data-driven approaches, such as distributional semantic models and information extraction, to mine for word senses and relations between them from large textual corpora, such as Wikipedia<sup>3</sup> or CommonCrawl<sup>4</sup>. This alternative automatic way of building lexical-semantic resources, in contrast to the manual approach, usually yield high *recall* due to the huge lexical coverage of unlabeled text corpora, but its results may be noisy i.e. of low *precision*. One of the overarching goals of this work was to bridge the gap between these two views on lexical semantics getting the best of both worlds while

---

<sup>1</sup><https://wordnet.princeton.edu>

<sup>2</sup><https://www.wiktionary.org>

<sup>3</sup><https://www.wikipedia.org>

<sup>4</sup><https://commoncrawl.org>

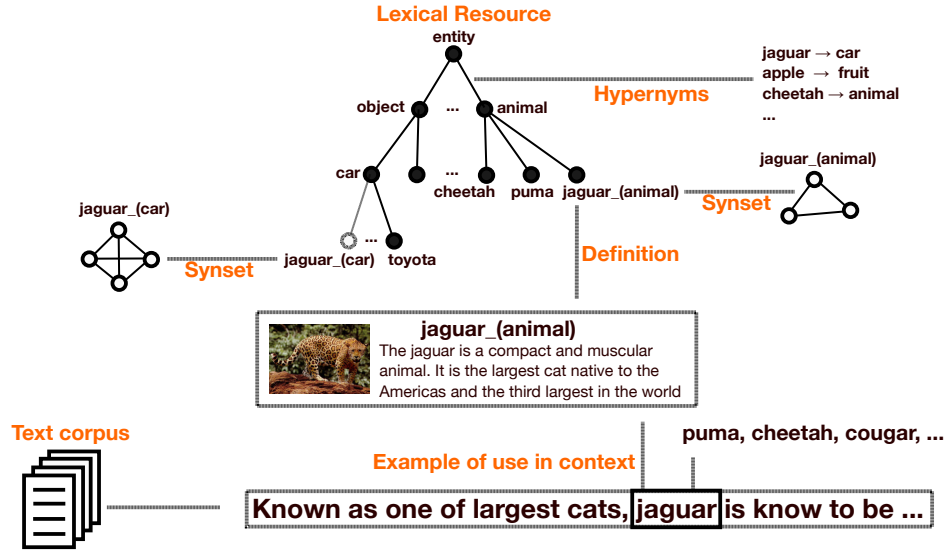


Figure 1-1: Overview of various concepts and tasks in computational lexical semantics modelled and used in this dissertation and their interrelations.

combining high precision of manual resources with the high recall of automatic techniques.

Figure 1-1 illustrates main linguistic concepts and interactions between them which are modelled and processed using computational methods discussed in this dissertation. Namely, a **lexical resource** is commonly represented as a graph  $G = (V, E)$  with the set of nodes  $V$  representing **concepts** and the set of edges  $E$  representing **semantic relations** between them, such as **synonymy**, such as  $e_i = (car, vehicle)$  or **hyperonymy**, such as  $e_j = (Toyota, car)$ . **Co-hyponyms**, such as  $(apple, pear)$  are semantic siblings – terms with a common hypernym i.e. “apple → fruit” and “pear → fruit”.

Individual nodes may be represented as a **synset** i.e. a clique of synonyms, such as  $\{car, vehicle, automobile\}$  or  $\{behemoth, hippopotamus\}$ . Each node usually represents a word in a given **word sense**, e.g. “jaguar (animal)” as opposed to “jaguar (car)” and may feature a human readable **definition** of this sense. At the same time, **mentions** of the word “jaguar” in a **text corpus** are not provided with explicit labels of word senses, such as “animal” or “car” unless someone marked them in a lexical resource as **examples** of word use in context. These, together with

graphical illustrations of word senses help to make definitions human-interpretable.

Figure 1-2 presents an overview of key methods for computational lexical semantics presented in this dissertation and their interrelations. To start a summary of these contributions, let us turn to another gap between two mentioned above methodological “realms” i.e., manually created vs data-driven lexical representations of words, namely the lack of explicit sense annotations in raw texts. At the very bottom of Figure 1-2 a textual context of the word “jaguar” is presented. A string “jaguar” may refer to several word senses, such as “jaguar\_(animal)” and “jaguar\_(car)” as described in a manually created taxonomy. However, there is obviously no identifier in text referring to one of these senses in the lexical resource. A **word sense disambiguation** (WSD) algorithm automatically identifies the most suitable meaning of a word given the context. Another related technology is **lexical substitution**. Instead of explicit linking to word senses it generates semantically suitable substitutes (usually, synonyms, co-hyponyms, or hypernyms) in the correct sense. For instance, here for the word “jaguar” substitutes “puma”, “cheetah”, and “cougar” are generated and not “Mercedes”, “BMW”, or “Audi”.

The interest of performing WSD is to link text to rich, precise and interpretable word sense representations from the manually created resource, which may feature elements such as definitions, images, hypernyms, and related words. At the same time, newly inserted word senses discovered automatically as described above, do not have such representations. That is why one of the contributions was to create a technology for automatic building such rich **human-interpretable multi-modal word sense representations**.

Finally, the use of word sense representations in machine learning models, especially in (deep) neural networks is complicated if these are represented in symbolic form of graphs (which correspond to sparse vector representation also known as “one-hot encoding”). Besides, for similarity computations between word senses low-dimensional dense vector format (also known as “embedding”) is preferable over sparse vector format or graph representations. That is why, procedures of **graph vectorization** were developed and tested on large scale

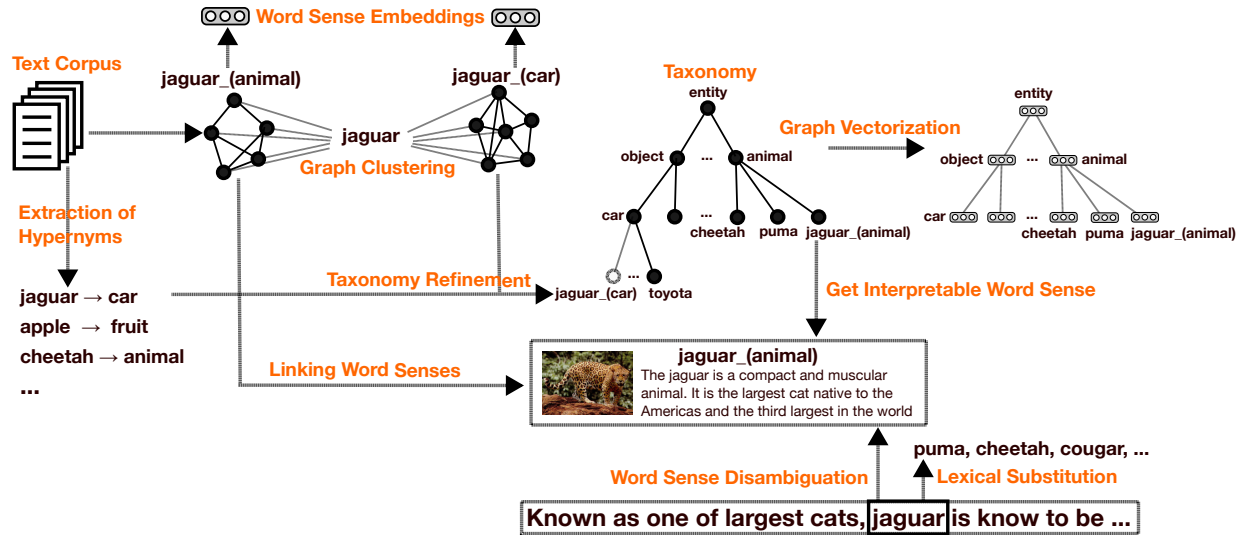


Figure 1-2: An “eco-system” of the developed approaches: an overview of the methods of computational lexical semantics presented in this dissertation and their interrelations.

lexical resources, such as Wordnet and DBpedia<sup>5</sup>. Taking as input a lexical resource network these output **node embeddings** of this graph. For similar purposes, procedures for obtaining **word sense embeddings** from the graph-based sense representations induced from text were proposed. These representations are further used to perform WSD based on similarity computations between context words and these newly learned vector representations of word senses.

To summarize, Figure 1-2 presents an overview of various methods for computational lexical semantics proposed in this dissertation and illustrate how they interact with one another. Namely, word senses induced from text corpus though **graph clustering** is the process known as **word sense induction**. Word senses mined from text though this procedure may already be present in the lexical resource, such as “jaguar\_(animal)”. In this case, a **linking of word senses** of these senses is required. Alternatively a word sense may cover a new meanings not present in the resource, but represented in text. In the latter case, this newly discovered word sense is added into the resource increasing its recall. Similar is done with semantic relations. The figure illustrates how hypernyms (also known as

<sup>5</sup><https://dbpedia.org>



“Is-A” or hierarchical relations) are extracted from text corpus and used to perform enrichment of an existing taxonomy.

Therefore, contributions presented in this dissertation cover a wide range of task related to lexical computational semantics: they form a solid methodological framework for learning, population, linking, disambiguation and vectorization of word senses and relations between them.

Methodologically, many developed methods are graph-based, more specifically graph clustering algorithms, including newly proposed, are used to process linguistic networks of various kinds. The use of graph representation is fairly natural as each lexical semantic resource can be represented as graph with nodes being word senses or terms and edges being semantic relations between them. As the modern NLP methods heavily rely on neural networks, dealing with such linguistic graphs required their vectorisation. Towards this end, methods for node embedding of linguistic graphs, such as WordNets and Knowledge Graphs (KG) were developed to solve various tasks, such as completion of linguistic resources and word sense disambiguation.

The proposed methods for learning and population from text are applicable to all common lexical linguistic resources, such as lexical semantic databases, e.g. Wordnet or Babelnet<sup>6</sup>, thesauri, taxonomies, as they all can be represented in the form of graph with nodes corresponding to word senses. Methods for induction of word senses from text mining new nodes of such graphs. Algorithms for linking of newly mined word senses with existing resources find correspondances between nodes of manually constructed and automatically constructed graphs. Methods for induction of semantic taxonomies, which form a backbone of pretty much every type of lexical semantic resource, are learning special kinds of directed graphs - trees. Crucially, most practical application of a sophisticated lexical resource, such as WordNet, involve mapping senses representation listed in this resource to tokens occurring in a raw text. Towards this end, developed methods for WSD search for optimal correspondences in terms of semantic coherence of nodes of lexical resource and tokens in text.

Therefore, the set of proposed methods deal with both graph and vector-based

---

<sup>6</sup><https://babelnet.org>

representation of word senses as they are highly complementary: for various use-cases one or another may be used. The contents of the thesis are arranged in 9 “content chapters”, covering a variety of topics. At the same time each chapter is related to the use of graph and/or vector based representations to a lexical-semantic processing task. Among all methodological techniques graph clustering shall be highlighted as the central one, applied in various contexts, such as induction of word senses, semantic frames or improving quality of hypernymy relations.

### **Object and goals of the dissertation.**

The purpose of the dissertation is the development of methods for computational lexical semantics which would bridge the gap between (i) the *precise well-interpretable* manually created lexical resources with *low lexical coverage* and (ii) *noisy non-interpretable* automatically induced from text distributional lexical representations with *high lexical coverage*. This includes (i) development of new algorithms for clustering of large linguistic networks constructed from both manually created lexical resources and graphs induced from text, (ii) development of methods for induction of lexical semantic structures of various kinds from text, most notably word senses and hypernymy relations, (iii) development of techniques for making the induced structures interpretable in the way they are in manually constructed resources, (iv) development of methods for effective disambiguation in context with respect to the induced sense representations, (v) development of effective vectorization of lexical semantic graphs for the use in various applications, (vi) development methods for automatic construction of lexical-semantic hierarchies.

### **The obtained results:**

1. Developing an algorithm for fuzzy graph clustering effective and efficient for processing of large lexical-semantic networks (Chapter 2).
2. Based on the developed fuzzy graph clustering algorithm, we propose a methods for induction of three lexical semantic structures: synsets, semantic frames, and semantic classes (Chapter 2).

3. Proposing a method for learning sense embeddings from word embeddings using graph clustering, and a word sense disambiguation method using the induced sense representations (Chapter 3).
4. Developing a method for inducing from text word sense representations using graph-based distributional methods and techniques for making these representations interpretable by automatic retrieval of hypernyms, images, and definitions (Chapter 4).
5. Proposing a framework for enriching lexical resources with distributional information, featuring algorithm for linking distributionally induced sense representations to manually created lexical resources, and algorithm for disambiguation of related words and hypernyms (Chapter 5).
6. Developing a model for generation of hypernyms of words based on projection learning with regularization of relation asymmetry (Chapter 6).
7. Proposing a method for post-processing of hypernymy relations using distributionally induced semantic classes: wrong hypernyms are removed while the missing ones are added (Chapter 7).
8. Proposing an algorithm for construction of taxonomic tree (composed of binary hypernymy relations between terms) using Euclidean and hyperbolic (Poincaré) vector representations (Chapter 8).
9. Proposing a model for learning node embeddings of linguistic networks using supervision from graph-based similarity metrics and show its applicability to word sense disambiguation task, inter alia (Chapter 9).
10. Proposing methods for neural lexical substitution which integrates information about the target word with the information about the context (Chapter 10).
11. Investigating distribution of lexical-semantic relations provided the neural lexical substitution models (Chapter 10).

**Author's contribution** includes the formal problem formulations and experimental design for all the mentioned above results, the design of the methods and algorithms mentioned, analysis and generalization of the results. The more specific author's contribution (and list of key collaborators) related to each paper are provided in the list of publications below.

**The novelty** of the proposed research lies in the development of new algorithms for computational lexical semantics. Most of the proposed methods deal with either distributional semantics methods, graph clustering algorithms or both of them. A novel graph clustering approach lies at the core of several newly proposed methods for lexical semantics. For instance, those for automatic induction of lexical semantic structures, such as synsets, semantic frames and semantic classes. Several contributions are related to automatic processing of hypernymy relationships between words. In particular, in the dissertation the author proposes:

- Algorithm for fuzzy graph clustering (Chapter 2);
- Methods for induction of synsets, semantic classes and frames (Chapter 2);
- Methods for learning word sense embeddings (Chapter 3);
- Method for inducing interpretable word sense representations from text (Chapter 4);
- Method for linking distributional word sense representations with lexical resources (Chapter 5);
- Model for generation of hypernyms (Chapter 6);
- Method for post-processing of noisy hypernymy relations (Chapter 7);
- Algorithm for construction of lexical semantic trees i.e. taxonomies (Chapter 8);
- Model for node embeddings of linguistic graphs (Chapter 9);
- Methods for neural lexical substitution (Chapter 10);

- Study of distribution of lexical semantic relations provided by neural lexical substitution models (Chapter 10).

Now let us turn to the summary of published research this dissertation is based on. The scope of dissertation is covered in 42 publications [1–42], among those:

- 5 papers are published in CORE A\* conferences [3, 5, 9, 10, 13];
- 6 papers are published in CORE A conferences [1, 2, 4, 7, 16];
- 5 articles are published in in Q1 journals [6, 8, 11, 12, 15];
- 1 paper is published in CORE A\* conference student track [28];
- 1 paper is published in CORE A conference demo track [18];
- 5 papers is published at CORE B conference [19, 20, 26, 27, 29];
- 11 papers indexed by Scopus published in proceedings of the main volumes of conferences [22–25, 30–34, 40, 41];
- 8 papers indexed by Scopus published in workshops co-located with top conferences (CORE A\*/A) [17, 21, 35–39, 42].

According to regulations of the Dissertation Council in Computer Sciences of Higher School of Economics and for brevity among the 42 papers topically fitting the dissertation only 20 key papers are listed below including *all* 16 first-tier, i.e. A\*/A/Q1, and *selected* 4 second-tier. The *remaining* 22 second-tier publications are cited in the bibliography and are openly available online. **The defence is performed based on 14 publications of these 20 listed below works.** Namely, based on the first 10 from the list of first-tier publications [1–10] and the 4 second-tier publications [17–20]. To avoid confusions, an note “used for defence” is included below where appropriate.

## First-tier publications:

1. [A. Panchenko](#), E. Ruppert, S. Faralli, S. P. Ponzetto, and C. Biemann, “**Unsupervised does not mean uninterpretable: The case for word sense induction and disambiguation,**” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, (Valencia, Spain), pp. 86–98, Association for Computational Linguistics, Apr. 2017  
<https://aclanthology.org/E17-1009>  
[CORE A] *used for defence; main co-author; the author of this thesis has proposed an interpretable unsupervised knowledge-free word sense disambiguation (WSD) method consisting of (1) a technique to disambiguation that relies on induced inventories as a pivot for learning sense feature representations, (2) a technique for making induced sense representations interpretable by labelling them with hypernyms and images.*
2. N. Arefyev, B. Sheludko, A. Podolskiy, and [A. Panchenko](#), “**Always Keep your Target in Mind: Studying Semantics and Improving Performance of Neural Lexical Substitution,**” in *Proceedings of the 28th International Conference on Computational Linguistics*, (Barcelona, Spain (Online)), pp. 1242–1255, International Committee on Computational Linguistics, Dec. 2020  
<https://aclanthology.org/2020.coling-main.107>  
[CORE A] *used for defence; main co-author; the author designed (in an inseparable cooperation with N. Arefyev) methods of target word injection for lexical substitution quality improvement; performed an analysis of types of semantic relations (synonyms, hypernyms, co-hyponyms, etc.) produced by neural substitution models.*
3. A. Kutuzov, M. Dorgham, O. Oliynyk, C. Biemann, and [A. Panchenko](#), “**Making Fast Graph-based Algorithms with Graph Metric Embeddings,**” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 3349–3355, Association for Computational Linguistics, July 2019  
<https://aclanthology.org/P19-1325>

- [**CORE A\***] *used for defence; main co-author; the author designed a method that learns dense vector embeddings of nodes based on a pre-defined graph-based similarity measure, e.g. the shortest path distance.*
4. D. Ustalov, N. Arefyev, C. Biemann, and A. Panchenko, “**Negative Sampling Improves Hypernymy Extraction Based on Projection Learning**,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, (Valencia, Spain), pp. 543–550, Association for Computational Linguistics, Apr. 2017  
<https://aclanthology.org/E17-2087>  
[**CORE A**] *used for defence; main co-author; the author (in an inseparable cooperation with D. Ustalov) designed an approach for hypernymy extraction based on projection learning, which makes use of both positive and negative training instances enforcing the asymmetry of the projection.*
5. R. Aly, S. Acharya, A. Ossa, A. Köhn, C. Biemann, and A. Panchenko, “**Every Child Should Have Parents: A Taxonomy Refinement Algorithm Based on Hyperbolic Term Embeddings**,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 4811–4817, Association for Computational Linguistics, July 2019  
<https://aclanthology.org/P19-1474>  
[**CORE A\***] *used for defence; main co-author; the author designed a method for refinement of semantic tree structures (taxonomies) using embeddings based on Euclidian and hyperbolic vector spaces.*
6. D. Ustalov, A. Panchenko, C. Biemann, and S. P. Ponzetto, “**Watset: Local-Global Graph Clustering with Applications in Sense and Frame Induction**,” *Computational Linguistics*, vol. 45, pp. 423–479, Sept. 2019  
<https://aclanthology.org/J19-3002>  
[**Q1**] *used for defence; main co-author; the author proposed (in an inseparable cooperation with D. Ustalov) a meta-algorithm for fuzzy graph clustering using hard clustering methods; methods for induction of synsets, semantic frames, and semantic classes based on based on the proposed algorithm.*

7. S. Faralli, A. Panchenko, C. Biemann, and S. P. Ponzetto, “**Linked Disambiguated Distributional Semantic Networks**,” in *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II* (P. Groth, E. Simperl, A. J. G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck, and Y. Gil, eds.), vol. 9982 of *Lecture Notes in Computer Science*, pp. 56–64, 2016  
[https://doi.org/10.1007/978-3-319-46547-0\\_{\\\_}7](https://doi.org/10.1007/978-3-319-46547-0_{\_}7)  
**[CORE A]** *used for defence; main co-author; the author developed (in inseparable cooperation with S. Faralli) methodology to automatically induce distributionally-based semantic representations from large amounts of text, and link them to a reference knowledge base.*
8. C. Biemann, S. Faralli, A. Panchenko, and S. P. Ponzetto, “**A framework for enriching lexical semantic resources with distributional semantics**,” *Nat. Lang. Eng.*, vol. 24, no. 2, pp. 265–312, 2018  
<https://doi.org/10.1017/S135132491700047X>  
**[Q1]** *used for defence; main co-author; the author has proposed (in inseparable cooperation with S. Faralli) a framework for enriching lexical semantic resources consisting of methods for combining information from distributional semantic models with manually constructed lexical semantic resources.*
9. D. Ustalov, A. Panchenko, and C. Biemann, “**Watset: Automatic Induction of Synsets from a Graph of Synonyms**,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Vancouver, Canada), pp. 1579–1590, Association for Computational Linguistics, July 2017  
<https://aclanthology.org/P17-1145>  
**[CORE A\*]** *used for defence; main co-author; the author has proposed (in inseparable cooperation with D. Ustalov) a novel approach that resolves ambiguities in the input graph to perform fuzzy clustering.*
10. D. Ustalov, A. Panchenko, A. Kutuzov, C. Biemann, and S. P. Ponzetto, “**Unsupervised Semantic Frame Induction using Triclustering**,” in *Proceedings of the 56th Annual Meeting of the Association for Computational*



*Linguistics (Volume 2: Short Papers)*, (Melbourne, Australia), pp. 55–62, Association for Computational Linguistics, July 2018

<https://aclanthology.org/P18-2010>

[CORE A\*] *used for defence; main co-author; the author has proposed (in inseparable cooperation with D. Ustalov) new approach to triclustering; proposed to apply triclustering algorithms for unsupervised frame induction; proposed new method for the evaluation of frame induction.*

11. Ö. Sevgili, A. Shelmanov, M. Y. Arkhipov, A. Panchenko, and C. Biemann, “**Neural entity linking: A survey of models based on deep learning**,” *Semantic Web*, vol. 13, no. 3, pp. 527–570, 2022

<https://doi.org/10.3233/SW-222986>

[Q1] *main co-author; the author has proposed the overall idea and structure of the survey; created formal definitions of tasks; guided the writing.*

12. S. Anwar, A. Shelmanov, N. Arefyev, A. Panchenko, and C. Biemann, “**Text augmentation for semantic frame induction and parsing**,” *Language Resources and Evaluation*, vol. 23, no. 3, pp. 527–556, 2023

<https://link.springer.com/article/10.1007/s10579-023-09679-8>

[Q1] *main co-author; the author designed a one-shot method for inducing frame-semantic structures using lexical substitution on frame-annotated sentences; designed experimental setting for semantic frame parsing and induction.*

13. A. Jana, D. Puzyrev, A. Panchenko, P. Goyal, C. Biemann, and A. Mukherjee, “**On the Compositionality Prediction of Noun Phrases using Poincaré Embeddings**,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 3263–3274, Association for Computational Linguistics, July 2019

<https://aclanthology.org/P19-1316>

[CORE A\*] *main co-author; the author designed a straightforward and efficient approach for combining distributional and hypernymy information using hyperbolic embeddings for the task of noun phrase compositionality prediction.*

14. I. Nikishina, V. Logacheva, A. Panchenko, and N. Loukachevitch, “**Studying**

**Taxonomy Enrichment on Diachronic WordNet Versions,”** in *Proceedings of the 28th International Conference on Computational Linguistics*, (Barcelona, Spain (Online)), pp. 3095–3106, International Committee on Computational Linguistics, Dec. 2020

<https://aclanthology.org/2020.coling-main.276>

[**CORE A**] *main co-author; the author designed several methods for taxonomy enrichment.*

15. I. Nikishina, M. Tikhomirov, V. Logacheva, Y. Nazarov, A. Panchenko, and N. V. Loukachevitch, “**Taxonomy enrichment with text and graph vector representations,**” *Semantic Web*, vol. 13, no. 3, pp. 441–475, 2022

<https://doi.org/10.3233/SW-212955>

[**Q1**] *main co-author; the author designed (in inseparable cooperation with I. Nikishina) methods for taxonomy enrichment using text and graph vectors.*

16. S. Faralli, A. Panchenko, C. Biemann, and S. P. Ponzetto, “**The ContrastMedium Algorithm: Taxonomy Induction From Noisy Knowledge Graphs With Just A Few Links,**” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, (Valencia, Spain), pp. 590–600, Association for Computational Linguistics, Apr. 2017

<https://aclanthology.org/E17-1056>

[**CORE A**] *main co-author; the author designed (in inseparable cooperation with S. Faralli) end-to-end pipeline for taxonomy induction from scratch using distributional semantic representations.*

### **Second-tier publications:**

17. M. Pelevina, N. Arefiev, C. Biemann, and A. Panchenko, “**Making Sense of Word Embeddings,**” in *Proceedings of the 1st Workshop on Representation Learning for NLP*, (Berlin, Germany), pp. 174–183, Association for Computational Linguistics, Aug. 2016

<https://aclanthology.org/W16-1620>

[**Scopus, highly cited**] *used for defence*; *main co-author; the author of this thesis*

*designed a novel method for learning sense embeddings from word embeddings and a word sense disambiguation (WSD) mechanism using these sense representations.*

18. [A. Panchenko](#), F. Marten, E. Ruppert, S. Faralli, D. Ustalov, S. P. Ponzetto, and C. Biemann, “**Unsupervised, Knowledge-Free, and Interpretable Word Sense Disambiguation**,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Copenhagen, Denmark), pp. 91–96, Association for Computational Linguistics, Sept. 2017  
<https://aclanthology.org/E17-1009>

[**CORE A, demo track**] *used for defence; main co-author; the author of this thesis has designed the first system for word sense induction and disambiguation, which is unsupervised, knowledge-free, and interpretable at the same time.*

19. [A. Panchenko](#), D. Ustalov, S. Faralli, S. P. Ponzetto, and C. Biemann, “**Improving Hypernymy Extraction with Distributional Semantic Classes**,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, (Miyazaki, Japan), European Language Resources Association (ELRA), May 2018  
<https://aclanthology.org/L18-1244>

[**CORE B**] *used for defence; main co-author; the author of this thesis has proposed an unsupervised method for post-processing of noisy hypernymy relations based on clustering of graphs of word senses induced from text.*

20. V. Logacheva, D. Teslenko, A. Shelmanov, S. Remus, D. Ustalov, A. Kutuzov, E. Artemova, C. Biemann, S. P. Ponzetto, and [A. Panchenko](#), “**Word Sense Disambiguation for 158 Languages using Word Embeddings Only**,” in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, (Marseille, France), pp. 5943–5952, European Language Resources Association, May 2020  
<https://aclanthology.org/2020.lrec-1.728>

[**CORE B**] *used for defence; main co-author; the author of this thesis proposed a new algorithm of unsupervised word sense induction, which creates sense inventories based on pre-trained word vectors which is based on ego-graph vector induction.*

### Reports at conferences and seminars:

In this section, we list conferences where the 42 papers relevant to the scope of this dissertation were presented, not only those 14 papers the defence is based on.<sup>7</sup>

1. **ACL-2019** [CORE A\*] [3,5,13,28,36]: The 57th Annual Meeting of the Association for Computational Linguistics, (Florence, Italy), Association for Computational Linguistics, July 2019.
2. **ACL-2018** [CORE A\*] [10]: The 56th Annual Meeting of the Association for Computational Linguistics (Melbourne, Australia), Association for Computational Linguistics, July 2018.
3. **ACL-2017** [CORE A\*] [9]: The 55th Annual Meeting of the Association for Computational Linguistics (Vancouver, Canada), Association for Computational Linguistics, July 2017.
4. **ACL-2016** [CORE A\*] [17]: The 54th Annual Meeting of the Association for Computational Linguistics (Berlin, Germany). Association for Computational Linguistics.
5. **IJCNLP-ACL-2021** [CORE A\*] [35]: The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Bankok, Thailand), Association for Computational Linguistics.
6. **COLING-2022** [CORE A] [42]: The 29th International Conference on Computational Linguistics (Gyeongju, Republic of Korea), International Committee on Computational Linguistics, October 2022.
7. **COLING-2020** [CORE A] [2, 14]: The 28th International Conference on Computational Linguistics, (Barcelona, Spain), International Committee on Computational Linguistics, December 2020.
8. **EACL-2017** [CORE A] [1,4,16,39]: The 15th Conference of the European Chapter of the Association for Computational Linguistics (Valencia, Spain), Association for Computational Linguistics, April 2017. [1]

---

<sup>7</sup>To obtain the exact rank of a publication please refer to the list above as some of the papers were presented at the associated workshops co-located with the main conference.

9. **EMNLP-2017** [CORE A] [18]: The 2017 Conference on Empirical Methods in Natural Language Processing (Copenhagen, Denmark), Association for Computational Linguistics, September 2017.
10. **ISWC-2016** [CORE A] [7]: The 15th International Semantic Web Conference, (Kobe, Japan), vol. 9982 of Lecture Notes in Computer Science, October, 2016.
11. **NAACL-2019** [CORE A] [37,38]: 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Minneapolis, Minnesota, USA), Association for Computational Linguistics, June 2019.
12. **NAACL-2016** [CORE A] [21]: The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (San Diego, California, USA), Association for Computational Linguistics.
13. **AAACL-2022** [CORE B] [40]: The 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Taipei, Taiwan), Association for Computational Linguistics, November 2022.
14. **LREC-2020** [CORE B] [20]: The 12th Language Resources and Evaluation Conference, (Marseille, France), European Language Resources Association, May 2020.
15. **LREC-2018** [CORE B] [19, 26, 27]: The 11th International Conference on Language Resources and Evaluation (LREC 2018), (Miyazaki, Japan), European Language Resources Association (ELRA), May 2018.
16. **LREC-2016** [CORE B] [29]: The 10th International Conference on Language Resources and Evaluation (LREC'16), (Portorož, Slovenia), pp. 2649–2655, European Language Resources Association (ELRA), May 2016.
17. **PaM-2020** [Scopus] [22]: The Probability and Meaning Conference (Gothenburg, Sweden), Association for Computational Linguistics, June 2020.
18. **RANLP-2019** [Scopus] [33]: The International Conference on Recent Advances in Natural Language Processing (Varna, Bulgaria), INCOMA Ltd., September 2019.

19. **GWC-2021** [Scopus] [41]: The 11th Global Wordnet Conference (Potchefstroom, South Africa), Global Wordnet Association, January 2021.
20. **AIST-2019** [Scopus/Q2] [32]: The 8th International Conference on Analysis of Images, Social Networks and Texts (Kazan, Russia), July 2019, vol. 11832 of Lecture Notes in Computer Science, Springer.
21. **AIST-2017** [Scopus/Q2] [30]: The 6th International Conference on Analysis of Images, Social Networks and Texts (Moscow, Russia), July 2017, vol. 10716 of Lecture Notes in Computer Science, Springer.
22. **Dialogue-2018** [Scopus] [24, 25]: The 24th International Conference on Computational Linguistics and Intellectual Technologies (Moscow, Russia), RGGU, June 2018.
23. **KONVENS-2018** [Scopus] [23]: The 14th Conference on Natural Language Processing (Vienna, Austria), September, 2018 Österreichische Akademie der Wissenschaften.
24. **KONVENS-2016** [Scopus] [31]: The 13th Conference on Natural Language Processing, (Bochum, Germany), September 2016, vol. 16 of Bochumer Linguistische Arbeitsberichte.

# Chapter 2

## Graph Clustering for Sense and Frame Induction

Materials of this chapter are based on papers [6, 9, 10] from the list of 14 publications the thesis is based on.

### 2.1 Introduction

In this chapter, we deal with the task of graph clustering applying to extraction of various linguistic structures, such as synsets.

Let  $G = (V, E)$  be an undirected simple **graph**, where  $V$  is a set of nodes and  $E \subseteq V^2$  is a set of undirected edges. We denote a subset of nodes  $C^i \subseteq V$  as a **cluster**. A **graph clustering** is a function  $\text{CLUSTER} : (V, E) \rightarrow C$  such that  $V = \bigcup_{C^i \in C} C^i$ . Two classes of graph clustering exist: **hard clustering** algorithms produce non-overlapping clusters, i.e.,  $C^i \cap C^j = \emptyset \iff i \neq j, \forall C^i, C^j \in C$  while **fuzzy clustering** permit cluster overlapping, i.e., a node can be a member of several clusters in  $C$ .

Below, a meta-algorithm for fuzzy graph clustering is presented. It creates an intermediate representation of the input graph that reflects the “ambiguity” of its nodes. Then, it uses hard clustering to discover clusters in this “disambiguated” intermediate graph.

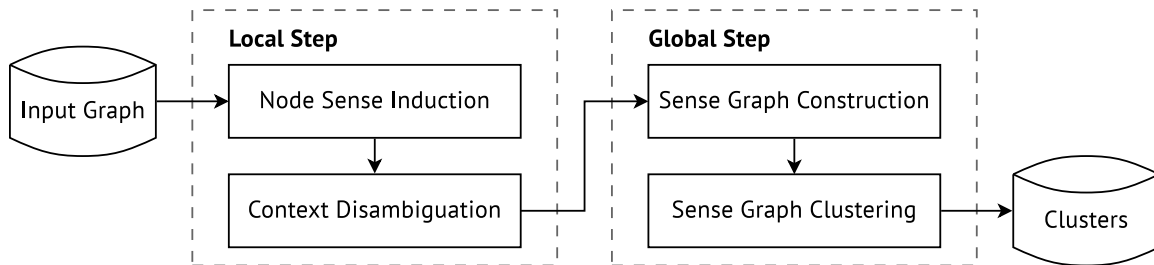


Figure 2-1: The outline of the algorithm showing the *local* step of node sense induction and context disambiguation, and the *global* step of sense graph constructing and clustering.

## 2.2 Method

In this section, a meta-algorithm for fuzzy graph clustering is presented. Given a graph connecting potentially ambiguous objects, e.g., words, it induces a set of unambiguous overlapping clusters (communities) by disambiguating and grouping the ambiguous objects. This is a meta-algorithm that uses existing *hard* clustering algorithms for graphs to obtain a *fuzzy* clustering, a.k.a. *soft* clustering.

### Outline of Fuzzy Graph Clustering Method

Algorithm constructs an intermediate representation of the input graph called a *sense graph*. This is achieved by node sense induction based on hard clustering of the input graph node neighborhoods. The sense graph has the edges established between the different *senses* of the input graph nodes. The global clusters of the input graph are obtained by applying a hard clustering algorithm to the sense graph.

An outline of our algorithm is depicted in Figure 2-1: it takes an undirected graph  $G = (V, E)$  as the input and outputs a set of clusters  $C$ . The algorithm has two steps: local and global. The *local* step, disambiguates the potentially ambiguous nodes in  $G$ . The *global* step, uses these disambiguated nodes to construct an intermediate sense graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  and produce the overlapping clustering  $C$ . Watset is parameterized by two graph partitioning algorithms  $\text{Cluster}_{\text{Local}}$  and  $\text{Cluster}_{\text{Global}}$ , and a context similarity measure  $\text{sim}$ . The complete pseudocode of Watset is presented in Algorithm 1. For the sake of illustration, while describing the approach, we will provide examples with words and their synonyms. However, Watset is not bound only to the lexical units and relationships, so our examples are given *without loss of generality*. Note also that Watset can be applied for both unweighted



---

**Algorithm 1** Watset, a Local-Global Meta-Algorithm for Fuzzy Graph Clustering.

---

**Input:** graph  $G = (V, E)$ ,hard clustering algorithms  $\text{Cluster}_{\text{Local}}$  and  $\text{Cluster}_{\text{Global}}$ ,context similarity measure  $\text{sim} : (\text{ctx}(a), \text{ctx}(b)) \rightarrow \mathbb{R}, \forall \text{ctx}(a), \text{ctx}(b) \subseteq V$ .**Output:** clusters  $\mathcal{C}$ .

```
1: for all  $u \in V$  do ▷ Local Step: Sense Induction
2:    $\text{senses}(u) \leftarrow \emptyset$ 
3:    $V_u \leftarrow \{v \in V : \{u, v\} \in E\}$  ▷ Note that  $u \notin V_u$ 
4:    $E_u \leftarrow \{\{v, w\} \in E : v, w \in V_u\}$ 
5:    $G_u \leftarrow (V_u, E_u)$ 
6:    $C_u \leftarrow \text{Cluster}_{\text{Local}}(G_u)$  ▷ Cluster the open neighborhood of  $u$ 
7:   for all  $C_u^i \in C_u$  do
8:      $\text{ctx}(u^i) \leftarrow C_u^i$ 
9:      $\text{senses}(u) \leftarrow \text{senses}(u) \cup \{u^i\}$ 
10:  $\mathcal{V} \leftarrow \bigcup_{u \in V} \text{senses}(u)$  ▷ Global Step: Sense Graph Nodes
11: for all  $\hat{u} \in \mathcal{V}$  do ▷ Local Step: Context Disambiguation
12:    $\widehat{\text{ctx}}(\hat{u}) \leftarrow \emptyset$ 
13:   for all  $v \in \text{ctx}(\hat{u})$  do
14:      $\hat{v} \leftarrow \arg \max_{v' \in \text{senses}(v)} \text{sim}(\text{ctx}(\hat{u}) \cup \{u\}, \text{ctx}(v'))$  ▷  $\hat{u}$  is a sense of  $u \in V$ 
15:      $\widehat{\text{ctx}}(\hat{u}) \leftarrow \widehat{\text{ctx}}(\hat{u}) \cup \{\hat{v}\}$ 
16:    $\mathcal{E} \leftarrow \{\{\hat{u}, \hat{v}\} \in \mathcal{V}^2 : \hat{v} \in \widehat{\text{ctx}}(\hat{u})\}$  ▷ Global Step: Sense Graph Edges
17:    $\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$  ▷ Global Step: Sense Graph Construction
18:    $\mathcal{C} \leftarrow \text{Cluster}_{\text{Global}}(\mathcal{G})$  ▷ Global Step: Sense Graph Clustering
19:    $C \leftarrow \{\{u \in V : \hat{u} \in \mathcal{C}^i\} \subseteq V : \mathcal{C}^i \in \mathcal{C}\}$  ▷ Remove the sense labels
20: return  $C$ 
```

---

and weighted graphs as soon as the underlying hard clustering algorithms  $\text{Cluster}_{\text{Local}}$  and  $\text{Cluster}_{\text{Global}}$  take edge weights into account.

### Local Step: Node Sense Induction and Disambiguation

The *local* step of Watset discovers the node senses in the input graph and uses this information to discover which particular senses of the nodes were connected via the edges of the input graph  $G$ .

**Node Sense Induction.** We induce node senses using the word neighborhood clustering approach by [43]. In particular, we assume that the removal of the nodes participating in many triangles separates a graph into several connected components. Each component corresponds to the sense of the target node, so this procedure is executed for every node independently. Figure 2-2 illustrates this approach for sense induction.

Given a node  $u \in V$ , we extract its open neighborhood  $G_u = (V_u, E_u)$  from the input

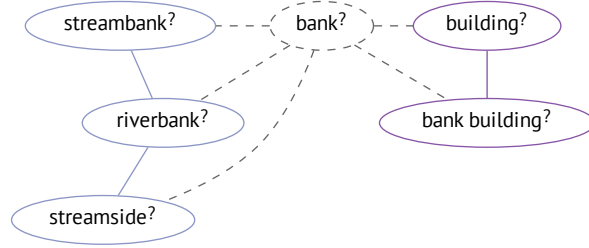


Figure 2-2: Clustering the neighborhood of the node “bank” of the input graph results in two clusters treated as the non-disambiguated sense contexts:  $bank^1 = \{streambank, riverbank, \dots\}$  and  $bank^2 = \{bank\ building, building, \dots\}$ .

Table 2.1: Example of induced senses for the node “bank” and the corresponding clusters (contexts).

Sense	Context
$bank^1$	$\{streambank, riverbank, \dots\}$
$bank^2$	$\{bank\ building, building, \dots\}$
$bank^3$	$\{bank\ company, \dots\}$
$bank^4$	$\{coin\ bank, penny\ bank, \dots\}$

graph  $G$ , such that the target node  $u$  is not included into  $V_u$  (lines 3–5):

$$V_u = \{v \in V : \{u, v\} \in E\}, \quad (2.1)$$

$$E_u = \{\{v, w\} \in E : v, w \in V_u\}. \quad (2.2)$$

Then, we run a hard graph clustering algorithm on  $G_u$  that assigns one node to one and only one cluster, yielding a clustering  $C_u$  (line 6). We treat each obtained cluster  $C_u^i \in C_u \subset V_u$  as representing a context for a different sense of the node  $u \in V$  (lines 7–9). We denote, e.g.,  $bank^1$ ,  $bank^2$  and other labels as the node *senses* referred to as senses(bank). In the example in Table 2.1,  $|\text{senses}(\text{bank})| = 4$ . Given a sense  $u^i \in \text{senses}(u)$ , we denote  $\text{ctx}(u^i) = C_u^i$  as a *context* of this sense of the node  $u \in V$ . Execution of this procedure for all the words in  $V$  results in the set of senses for the global step (line 10):

$$\mathcal{V} = \bigcup_{u \in V} \text{senses}(u). \quad (2.3)$$

**Disambiguation of Neighbors.** Although at the previous step we have induced node senses and mapped them to the corresponding contexts (Table 2.1), the elements of these

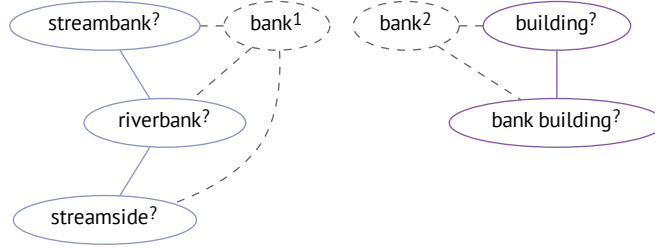


Figure 2-3: Contexts for two different senses of the node “bank”: only its senses  $bank^1$  and  $bank^2$  are currently known, while the other nodes in contexts need to be disambiguated.

Table 2.2: An example of context vectors for the node senses demonstrated in Figures 2-3 and 2-4. Since the graph is unweighted, one-hot encoding has been used. For matching purposes, the word “bank” is temporarily added into  $ctx(bank^2)$ .

Sense	bank	bank building	building	construction	edifice
$bank^2$	1	1	1	0	0
$building^1$	1	1	0	1	0
$building^2$	0	0	0	0	1

contexts do not contain sense information. For example, the context of  $bank^2$  in Figure 2-3 has two elements  $\{bank\ building^?,\ building^?\}$ , the sense labels of which are currently not known. We recover the sense labels of nodes in a context using the sense disambiguated as follows.

We represent each context as a vector in a vector space model [44] constructed for all the contexts. Since the graph  $G$  is simple and the context of any sense  $\hat{u} \in \mathcal{V}$  does not include the corresponding node  $u \in V$  (Table 2.1), we *temporarily* put it into the context during disambiguation. This prevents the situation of non-matching when the context of a candidate sense  $v' \in senses(v)$  has only one element and that element is  $u$ , i.e.,  $ctx(v') = \{u\}$ . We intentionally perform this insertion temporarily only during matching to prevent self-referencing. When a context  $ctx(\hat{u}) \subset V$  is transformed into a vector, we assign to each element  $v \in ctx(\hat{u})$  of this vector a weight equal to the weight of the edge  $\{u, v\} \in E$  of the input graph  $G$ . If  $G$  is unweighted, we assign 1 if and only if  $\{u, v\} \in E$ , otherwise 0 is assigned. Table 2.2 shows an example of the context vectors used for disambiguating the word  $building$  in the context of the sense  $bank^2$  in Figure 2-3. In this example the vectors essentially represent one-hot encoding as the example input graph is unweighted.

Then, given a sense  $\hat{u} \in \mathcal{V}$  of a node  $u \in V$  and the context of this sense  $ctx(\hat{u}) \subset V$ , we

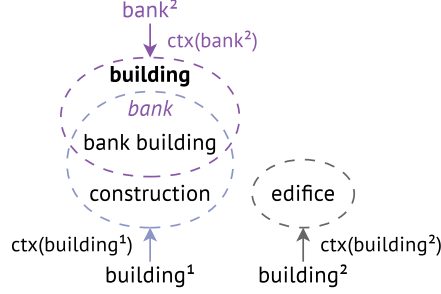


Figure 2-4: Matching the meaning of the ambiguous node “building” in the context of the sense  $bank^2$ . For matching purposes, the word “bank” is temporarily added into  $ctx(bank^2)$ .

*disambiguate* each node  $v \in ctx(\hat{u})$ . For that, we find the sense  $\hat{v} \in senses(v)$  the context  $ctx(\hat{v}) \subset V$  of which maximizes the similarity to the target context  $ctx(\hat{u})$ . We compute the similarity using a context similarity measure  $sim : (ctx(a), ctx(b)) \rightarrow \mathbb{R}, \forall ctx(a), ctx(b) \subseteq V$ . Typical choices for the similarity measure are dot product, cosine similarity, Jaccard index, etc. Hence, we *disambiguate* each context element  $v \in ctx(\hat{u})$ :

$$\hat{v} = \arg \max_{v' \in senses(v)} sim(ctx(\hat{u}) \cup \{u\}, ctx(v')). \quad (2.4)$$

An example in Figure 2-4 illustrates the node sense disambiguation process. The context of the sense  $bank^2$  is  $ctx(bank^2) = \{\text{building}, \text{bank building}\}$  and the disambiguation target is *building*. Having chosen cosine similarity as the context similarity measure, we compute the similarity between  $ctx(bank^2 \cup \{\text{bank}\})$  and the context of every sense of *building* in Table 2.2:  $\cos(ctx(bank^2) \cup \{\text{bank}\}, ctx(\text{building}^1)) = \frac{2}{3}$  and  $\cos(ctx(bank^2) \cup \{\text{bank}\}, ctx(\text{building}^2)) = 0$ . Therefore, for the word *building* in the context of  $bank^2$ , its first sense,  $\text{building}^1$ , should be used because its similarity value is higher.

Finally, we construct a disambiguated context  $\widehat{ctx}(\hat{u}) \subset \mathcal{V}$  which is a sense-aware representation of  $ctx(\hat{u})$ . This disambiguated context indicates which node senses were connected to  $\hat{u} \in \mathcal{V}$  in the input graph  $G$ . For that, in lines 13–15, we apply the disambiguation procedure defined in Equation (2.4) for every node  $v \in ctx(\hat{u})$ :

$$\widehat{ctx}(\hat{u}) = \{\hat{v} \in \mathcal{V} : v \in ctx(\hat{u})\}. \quad (2.5)$$

As the result of the *local* step, for each node  $u \in V$  in the input graph, we induce

the senses( $u$ )  $\subseteq \mathcal{V}$  of nodes and provide each sense  $\hat{u} \in \mathcal{V}$  with a disambiguated context  $\widehat{\text{ctx}}(\hat{u}) \subseteq \mathcal{V}$ .

**Global Step: Sense Graph Construction and Clustering**

The *global* step of Watset constructs an intermediate *sense graph* expressing the connections between the node senses discovered at the *local* step. We assume that the nodes  $\mathcal{V}$  of the sense graph are non-ambiguous, so running a hard clustering algorithm on this graph outputs clusters  $C$  covering the set of nodes  $V$  of the input graph  $G$ .

**Sense Graph Construction.** Using the set of node senses defined in Equation (2.3), we construct the sense graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  by establishing undirected edges between the senses connected through the disambiguated contexts (lines 16–17):

$$\mathcal{E} = \{\{\hat{u}, \hat{v}\} \in \mathcal{V}^2 : \hat{v} \in \widehat{\text{ctx}}(\hat{u})\}. \tag{2.6}$$

Note that this edge construction approach disambiguates the edges  $E$  such that if a pair of nodes was connected in the input graph  $G$ , then the corresponding sense nodes will be connected in the sense graph  $\mathcal{G}$ . As the result, the constructed sense graph  $\mathcal{G}$  is a sense-aware representation of the input graph  $G$ . In case  $G$  is weighted, we assign each edge  $\{\hat{u}, \hat{v}\} \in \mathcal{E}$  the same weight as the edge  $\{u, v\} \in E$  has in the input graph.

**Sense Graph Clustering.** Running a hard clustering algorithm on  $\mathcal{G}$  produces the set of sense-aware clusters  $\mathcal{C}$ , each sense-aware cluster  $\mathcal{C}^i \in \mathcal{C}$  is a subset of  $\mathcal{V}$  (line 18). In order to obtain the set of clusters  $C$  that covers the set of nodes  $V$  of the input graph  $G$ , we simply remove the sense labels from the elements of clusters  $\mathcal{C}$  (line 19):

$$C = \{\{u \in V : \hat{u} \in \mathcal{C}^i\} \subseteq V : \mathcal{C}^i \in \mathcal{C}\}. \tag{2.7}$$

Figure 2-5 illustrates the sense graph and its clustering on the example of the node “bank”. The construction of a sense graph requires disambiguation of the input graph nodes. Note that traditional approaches to graph-based sense induction, such as the ones proposed by [45–47], do not perform this step, but perform only local clustering of the graph since they do not aim at a global representation of clusters.

As the result of the *global* step, a set of clusters  $C$  of the input graph  $G$  is obtained using an intermediate sense-aware graph  $\mathcal{G}$ . The presented local-global graph clustering approach, Watset, makes it possible to naturally achieve a *soft* clustering of a graph using

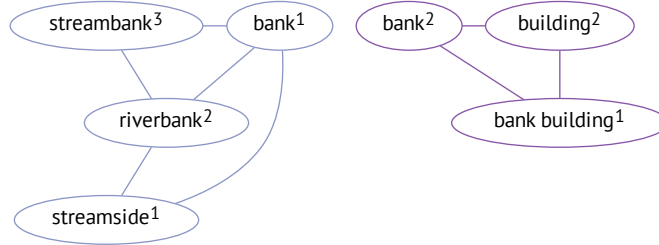


Figure 2-5: Clustering of the *sense graph*  $\mathcal{G}$  yields two clusters,  $\{bank^1, streambank^3, riverbank^2, \dots\}$  and  $\{bank^2, bank\ building^1, building^2, \dots\}$ ; if one removes the sense labels, the clusters will overlap resulting in a *soft* clustering of the input graph  $G$ .

Table 2.3: Various types of input linguistic graphs clustered by the Watset algorithm and the corresponding induced output symbolic linguistic structures.

Input Nodes	Input Edges	Output Linguistic Structure
Polysemous words	Synonymy relationships	Synsets composed of disambiguated words
Subject-Verb-Object (SVO) triples	Most distributionally similar SVO triples	Lexical semantic frames
Polysemous words	Most distributionally similar words	Semantic classes composed of disambiguated words

*hard* clustering algorithms only.

Its worth noting that the original paper [6] includes also description of a simplified version of Watset which allows to construct the sense graph  $\mathcal{G}$  in linear time  $O(|E|)$  by querying the node sense index to disambiguate the input edges  $E$  in a deterministic way. Other steps are identical to the original Watset algorithm presented here.

## 2.3 Results

Experiments show that, the algorithm shows competitive results in three applications: unsupervised synset induction from a synonymy graph, unsupervised semantic frame induction from dependency triples, and unsupervised semantic class induction from a distributional thesaurus. The algorithm is generic and can be also applied to other networks of linguistic data (see Table 2.3). Sample synsets induced by the Watset[MCL, MCL] method are presented in Table 2.4.

Further details, including theoretical and experimental algorithmic complexity analysis

Size	Synset
2	decimal point, dot
2	wall socket, power point
3	gullet, throat, food pipe
3	CAT, computed axial tomography, CT
4	microwave meal, ready meal, TV dinner, frozen dinner
4	mock strawberry, false strawberry, gurbir, Indian strawberry
5	objective case, accusative case, oblique case, object case, accusative
5	discipline, sphere, area, domain, sector
6	radio theater, dramatized audiobook, audio theater, radio play, radio drama, audio play
6	integrator, reconciler, consolidator, mediator, harmonizer, uniter
7	invite, motivate, entreat, ask for, incentify, ask out, encourage
7	curtail, crawl, yield, riding crop, harvest, crop, hunting crop

Table 2.4: Examples of extracted synses using the proposed graph clustering method from the graph of ambiguous synonyms.

and applications to real graphs can be found in [6, 9, 10].

# Chapter 3

## Word Sense Embeddings

Materials of this chapter are based on papers [17, 20] from the list of 14 publications the thesis is based on.

### 3.1 Introduction

In this chapter, graph clustering is applied to lexical-semantic networks generated from vector representations of words to obtain vector representations of word senses.

Word sense embedding learning task considered in this chapter is as following. Input is a set of **word vectors** (word embeddings) of an ambiguous vocabulary  $V$ :  $\forall v \in V \exists \mathbf{v} \in \mathbb{R}^d$ , where  $d$  is dimensionality of vector space. Outputs of the task are (i) a **word sense inventory**  $S$ :  $\forall v \in V \exists \{s_1, \dots, s_k\} : s_i \subset V$ , where  $k$  is the number of senses of word  $v$ , and (ii) a set of **word sense vectors**  $\forall s_i \exists \mathbf{s}_i \in \mathbb{R}^d$ .

A simple yet effective approach for learning such word sense embeddings is introduced. In contrast to existing techniques, which either directly learn sense representations from corpora or rely on sense inventories from lexical resources, our approach can induce a sense inventory from existing word embeddings via clustering of ego-networks of related words. An integrated WSD mechanism enables labelling of words in context with learned sense vectors, which gives rise to downstream applications.



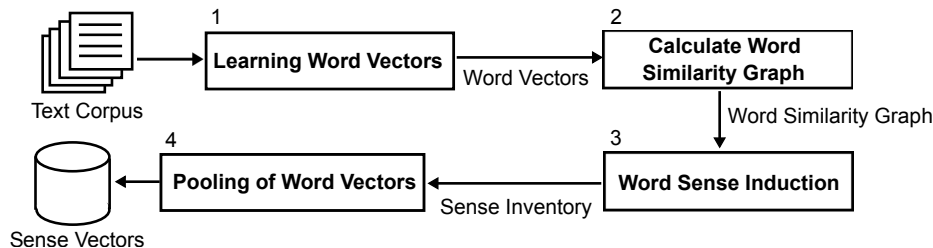


Figure 3-1: Schema of the word sense embeddings learning method SenseGram.

## 3.2 Method

The method consists of the four main stages depicted in Figure 3-1: (1) learning word embeddings; (2) building a graph of nearest neighbours based on vector similarities; (3) induction of word senses using ego-network clustering; and (4) aggregation of word vectors with respect to the induced senses.

The method can use existing word embeddings, sense inventories and word similarity graphs. To demonstrate such use-cases and to study the performance of the method in different settings, as variants of the complete pipeline presented in Figure 3-1, we experiment with two additional setups. The sense vectors are then constructed by averaging embeddings of words in each resulting cluster. In order to use these sense vectors for word sense disambiguation in text, the authors compute the probabilities of sense vectors of a word given its context or the similarity of the sense vectors to the context. Below we describe each of the stages of the method in detail.

### Learning Word Vectors

To learn word vectors, we use the *word2vec* [48] and *fastText* [49] but similar pre-trained word embeddings can be used. The final sense embeddings remain in the same vector space as these input word vectors.

### Calculating Word Similarity Graph

At this step, we build a graph of word similarities, such as (table, desk, 0.78). For each word we retrieve its 200 nearest neighbours. This number is motivated by prior studies [50, 51]: as observed, only few words have more strongly semantically related words. This graph is computed either based on word embeddings learned during the previous step or using semantic similarities provided by the *JoBimText* framework [50].

For similarities using word embeddings, such as *word2vec*, nearest neighbours of a term

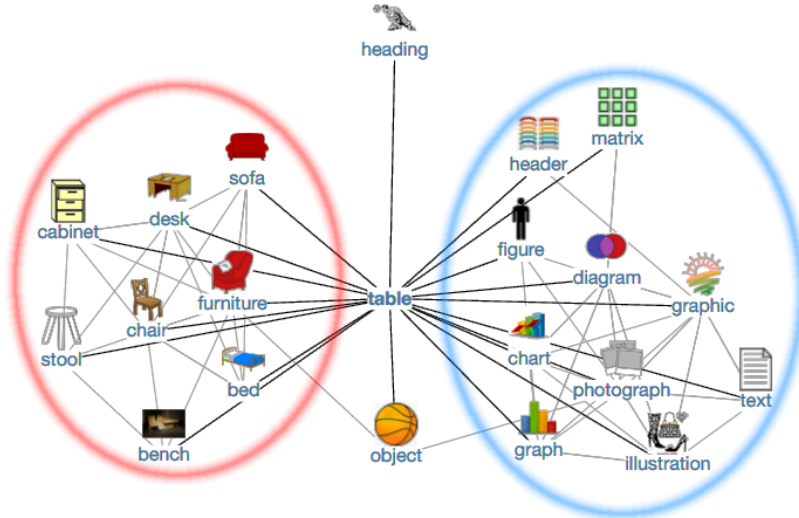


Figure 3-2: Visualization of the ego-network of “table” with furniture and data sense clusters. Note that the target “table” is excluded from clustering.

are terms with the highest cosine similarity of their respective vectors. For similarities using JoBimText (JBT) every word is represented as a bag of sparse dependency-based features extracted using the Malt parser and collapsed using an approach similar to [52]. Features are normalized using the LMI score [53] and further pruned down. Similarity of two words is equal to the number of common features.

### Word Sense Induction: a Baseline Algorithm

A word sense is represented by a word cluster. For instance the cluster “chair, bed, bench, stool, sofa, desk, cabinet” can represent the sense “table (furniture)”. To induce senses, first we construct an ego-network  $G$  of a word  $t$  and then perform graph clustering of this network. The identified clusters are interpreted as senses (see Figure 3-2). Words referring to the same sense tend to be tightly connected, while having fewer connections to words referring to different senses.

The sense induction presented in Algorithm 2 processes one word  $t$  of the word similarity graph  $T$  per iteration. First, we retrieve nodes  $V$  of the ego-network  $G$ : these are the  $N$  most similar words of  $t$  according to  $T$ . The target word  $t$  itself is not part of the ego-network. Second, we connect the nodes in  $G$  to their  $n$  most similar words from  $T$ . Finally, the ego-network is clustered with the Chinese Whispers algorithm [54]. This method is parameter free, thus we make no assumptions about the number of word senses.

The sense induction algorithm has three meta-parameters: the ego-network size ( $N$ ) of

---

**Algorithm 2** Word sense induction: Baseline Algorithm.

---

**input** :  $T$  – word similarity graph,  $N$  – ego-network size,  $n$  – ego-network connectivity,  $k$  – minimum cluster size

**output**: for each term  $t \in T$ , a clustering  $S_t$  of its  $N$  most similar terms

```
1 foreach  $t \in T$  do
2    $V \leftarrow N$  most similar terms of  $t$  from  $T$ 
    $G \leftarrow$  graph with  $V$  as nodes and no edges  $E$ 
3   foreach  $v \in V$  do
4      $V' \leftarrow n$  most similar terms of  $v$  from  $T$ 
       foreach  $v' \in V'$  do
5         if  $v' \in V$  then add edge  $(v, v')$  to  $E$ 
6       end
7   end
8    $S_t \leftarrow \text{ChineseWhispers}(G)$ 
    $S_t \leftarrow \{s \in S_t : |s| \geq k\}$ 
9 end
```

---

the target ego word  $t$ ; the ego-network connectivity ( $n$ ) is the maximum number of connections the neighbour  $v$  is allowed to have within the ego-network; the minimum size of the cluster  $k$ . The  $n$  parameter regulates the granularity of the inventory. In the experiments, we set the  $N$  to 200,  $n$  to 50, 100 or 200 and  $k$  to 5 or 15 to obtain different granulates, cf. [55]. Each word in a sense cluster has a weight which is equal to the similarity score between this word and the ambiguous word  $t$ .

### Word Sense Induction: an Improved Algorithm

One of the downsides of the described above algorithm is noise in the generated graph, namely, unrelated words and wrong connections. They hamper the separation of the graph. Another weak point is the imbalance in the nearest neighbour list, when a large part of it is attributed to the most frequent sense, not sufficiently representing the other senses. This can lead to construction of incorrect sense vectors.

We suggest an improved procedure of graph construction that uses the interpretability of vector addition and subtraction operations in word embedding space [56] while the previous algorithm only relies on the list of nearest neighbours in word embedding space. The key innovation is the use of vector subtraction to find pairs of most dissimilar graph nodes and construct the graph only from the nodes included in such “anti-edges”. Thus, the algorithm is based on *graph-based* word sense induction, but it also relies on *vector-based* operations between word embeddings to perform filtering of graph nodes. Analogously to the method

above, we construct a semantic relatedness graph from a list of nearest neighbours, but we filter this list using the following procedure:

1. Extract a list  $\mathcal{N} = \{w_1, w_2, \dots, w_N\}$  of  $N$  nearest neighbours for the target (ego) word vector  $w$ .
2. Compute a list  $\Delta = \{\delta_1, \delta_2, \dots, \delta_N\}$  for each  $w_i$  in  $\mathcal{N}$ , where  $\delta_i = w - w_i$ . The vectors in  $\delta$  contain the components of sense of  $w$  which are not related to the corresponding nearest neighbours from  $\mathcal{N}$ .
3. Compute a list  $\overline{\mathcal{N}} = \{\overline{w}_1, \overline{w}_2, \dots, \overline{w}_N\}$ , such that  $\overline{w}_i$  is in the top nearest neighbours of  $\delta_i$  in the embedding space. In other words,  $\overline{w}_i$  is a word which is the most similar to the target (ego) word  $w$  and least similar to its neighbour  $w_i$ . We refer to  $\overline{w}_i$  as an *anti-node* of  $w_i$ . The set of  $N$  nearest neighbours and their anti-nodes form a set of *anti-edges* i.e. pairs of most dissimilar nodes – those which should not be connected:  $\overline{E} = \{(w_1, \overline{w}_1), (w_2, \overline{w}_2), \dots, (w_N, \overline{w}_N)\}$ .

To clarify this, consider the target (ego) word  $w = \textit{python}$ , its top similar term  $w_1 = \textit{Java}$  and the resulting anti-node  $\overline{w}_i = \textit{snake}$  which is the top related term of  $\delta_1 = w - w_1$ . Together they form an anti-edge  $(w_i, \overline{w}_i) = (\textit{Java}, \textit{snake})$  composed of a pair of semantically dissimilar terms.

4. Construct  $V$ , the set of vertices of the semantic graph  $G = (V, E)$  from the list of anti-edges  $\overline{E}$ , with the following recurrent procedure:  $V = V \cup \{w_i, \overline{w}_i : w_i \in \mathcal{N}, \overline{w}_i \in \mathcal{N}\}$ , i.e. we add a word from the list of nearest neighbours *and* its anti-node only if both of them are nearest neighbours of the original word  $w$ . We do not add  $w$ 's nearest neighbours if their anti-nodes do not belong to  $\mathcal{N}$ . Thus, we add only words which can help discriminating between different senses of  $w$ .
5. Construct the set of edges  $E$  as follows. For each  $w_i \in \mathcal{N}$  we extract a set of its  $K$  nearest neighbours  $\mathcal{N}'_i = \{u_1, u_2, \dots, u_K\}$  and define  $E = \{(w_i, u_j) : w_i \in V, u_j \in V, u_j \in \mathcal{N}'_i, u_j \neq \overline{w}_i\}$ . In other words, we remove edges between a word  $w_i$  and its nearest neighbour  $u_j$  if  $u_j$  is also its anti-node. According to the hypothesis,  $w_i$  and  $\overline{w}_i$  belong to different senses of  $w$ , so they should not be connected (i.e. we never add anti-edges into  $E$ ). Therefore, we consider any connection between them as noise and remove it.

Note that  $N$  (the number of nearest neighbours for the target word  $w$ ) and  $K$  (the number of nearest neighbours of  $w_{ci}$ ) do not have to match. The difference between these parameters is the following.  $N$  defines how many words will be considered for the construction of ego-graph. On the other hand,  $K$  defines the degree of relatedness between words in the ego-graph — if  $K = 50$ , then we will connect vertices  $w$  and  $u$  with an edge only if  $u$  is in the list of 50 nearest neighbours of  $w$ . Increasing  $K$  increases the graph connectivity and leads to lower granularity of senses.

The described vertices selection procedure allows picking the most representative members of these clusters which are better at discriminating between the clusters. In addition to that, it helps dealing with the cases when one of the clusters is over-represented in the nearest neighbour list. In this case, many elements of such a cluster are not added to  $V$  because their anti-nodes fall outside the nearest neighbour list. This also improves the quality of clustering.

After the graph construction, the clustering is performed using the Chinese Whispers algorithm [46].

Figure 3-3 shows an example of the resulting pruned graph of for the word *Ruby* for  $N = 50$  nearest neighbours in terms of the fastText cosine similarity. In contrast to the baseline method described above where all 50 terms are clustered, in the method presented in this section we sparsify the graph by removing 13 nodes which were not in the set of the “anti-edges” i.e. pairs of most dissimilar terms out of these 50 neighbours. Examples of anti-edges i.e. pairs of most dissimilar terms for this graph include: (*Haskell*, *Sapphire*), (*Garnet*, *Rails*), (*Opal*, *Rubyist*), (*Hazel*, *RubyOnRails*), and (*Coffeescript*, *Opal*).

### Pooling of Word Vectors

At this stage, we calculate sense embeddings for each sense in the induced inventory. We assume that a word sense is a composition of words that represent the sense. We define a sense vector as a function of word vectors representing cluster items. Let  $W$  be a set of all words in the training corpus and let  $S_i = \{w_1, \dots, w_n\} \subseteq W$  be a sense cluster obtained during the previous step. Consider a function  $vec_w : W \rightarrow \mathbb{R}^m$  that maps words to their vectors and a function  $\gamma_i : W \rightarrow \mathbb{R}$  that maps cluster words to their weight in the cluster  $S_i$ . We experimented with two ways to calculate sense vectors: unweighted average of word vectors:

$$\mathbf{s}_i = \frac{\sum_{k=1}^n vec_w(w_k)}{n}; \quad (3.1)$$

Vector	Nearest Neighbours
table	tray, bottom, diagram, bucket, brackets, stack, basket, list, parenthesis, cup, trays, pile, playfield, bracket, pot, drop-down, cue, plate
table#0	leftmost#0, column#1, randomly#0, tableau#1, top-left#0, indent#1, bracket#3, pointer#0, footer#1, cursor#1, diagram#0, grid#0
table#1	pile#1, stool#1, tray#0, basket#0, bowl#1, bucket#0, box#0, cage#0, saucer#3, mirror#1, birdcage#0, hole#0, pan#1, lid#0

Table 3.1: Neighbours of the word “table” and its senses produced by the method. The neighbours of the initial vector belong to both senses, while those of sense vectors are sense-specific.

and weighted average of word vectors:

$$\mathbf{s}_i = \frac{\sum_{k=1}^n \gamma_i(w_k) \text{vec}_w(w_k)}{\sum_{k=1}^n \gamma_i(w_k)}. \quad (3.2)$$

Table 3.1 provides an example of weighted pooling results. While the original neighbours of the word “table” contain words related to both furniture and data, the neighbours of the sense vectors are either related to furniture or data, but not to both at the same time. Besides, each neighbour of a sense vector has a sense identifier as we calculate cosine between sense vectors, not word vectors.

### Word Sense Disambiguation

This section describes how sense vectors are used to disambiguate a word in a context. Given a target word  $w$  and its context words  $C = \{c_1, \dots, c_k\}$ , we first map  $w$  to a set of its sense vectors according to the inventory:  $S = \{\mathbf{s}_1, \dots, \mathbf{s}_n\}$ . We use two strategies to choose a correct sense taking vectors for context words either from the matrix of context embeddings or from the matrix of word vectors. The first one is based on sense probability in given context:

$$\mathbf{s}^* = \arg \max_i P(C|\mathbf{s}_i) = \arg \max_i \frac{1}{1 + e^{-\bar{\mathbf{c}}_c \cdot \mathbf{s}_i}}, \quad (3.3)$$

where  $\bar{\mathbf{c}}_c$  is the mean of context embeddings:  $k^{-1} \sum_{i=1}^k \text{vec}_c(c_i)$  and function  $\text{vec}_c : W \rightarrow \mathbb{R}^m$  maps context words to context embeddings. Using the mean of context embeddings to calculate sense probability is natural with the CBOW because this model optimizes exactly the same mean to have high scalar product with word embeddings for words occurred in context and low scalar product for random words [48].

The second disambiguation strategy is based on similarity between sense and context:

$$s^* = \arg \max_i \text{sim}(\mathbf{s}_i, C) = \arg \max_i \frac{\bar{\mathbf{c}}_w \cdot \mathbf{s}_i}{\|\bar{\mathbf{c}}_w\| \cdot \|\mathbf{s}_i\|}, \quad (3.4)$$

where  $\bar{\mathbf{c}}_w$  is the mean of word embeddings:  $\bar{\mathbf{c}}_w = k^{-1} \sum_{i=1}^k \text{vec}_w(c_i)$ . The latter method uses only word vectors ( $\text{vec}_w$ ) and require no context vectors ( $\text{vec}_c$ ). This is practical, as the standard implementation of *word2vec* does not save context embeddings and thus most pre-computed models provide only word vectors.

To improve WSD performance we also apply context filtering. Typically, only several words in context are relevant for sense disambiguation, like “chairs” and “kitchen” are for “table” in “They bought a table and chairs for kitchen.” For each word  $c_j$  in context  $C = \{c_1, \dots, c_k\}$  we calculate a score that quantifies how well it discriminates the senses:

$$\max_i f(\mathbf{s}_i, c_j) - \min_i f(\mathbf{s}_i, c_j), \quad (3.5)$$

where  $\mathbf{s}_i$  iterates over senses of the ambiguous word and  $f$  is one of the disambiguation strategies: either  $P(c_j|\mathbf{s}_i)$  or  $\text{sim}(\mathbf{s}_i, c_j)$ . The  $p$  most discriminative context words are used for disambiguation.

### Knowledge-Free Labelling of Induced Senses

We label each word cluster representing a sense to make them and the WSD results interpretable by humans. In the chapter below we show how hypernyms can be used label the clusters, e.g. “animal” in the “python (animal)”. Yet hypernyms are not available for some low-resourced languages. Therefore, we describe a simpler method to select a keyword which would help to interpret each cluster. For each graph node  $v \in V$  we count the number of anti-edges it belongs to:

$$\text{keyness}(v) = |\{(w_i, \bar{w}_i) : (w_i, \bar{w}_i) \in \bar{E} \wedge (v = w_i \vee v = \bar{w}_i)\}|. \quad (3.6)$$

A graph clustering yields a partition of  $V$  into  $n$  clusters:  $V = \{V_1, V_2, \dots, V_n\}$ . For each cluster  $V_i$  we define a *keyword*  $w_i^{\text{key}}$  as the word with the largest number of anti-edges  $\text{keyness}(\cdot)$  among words in this cluster.

### 3.3 Results

Main experiments were conducted for English language. Besides, this method was used to induce a collection of sense inventories for 158 languages on the basis of the original pre-trained fastText word embeddings by [57], enabling WSD in these languages.

The results suggest that, the presented algorithms on English WSI datasets and multi-lingual lexical similarity and relatedness task show that the performance of the method is comparable to state-of-the-art unsupervised WSD systems. Details of experimental results and their analysis can be found in [17, 20].

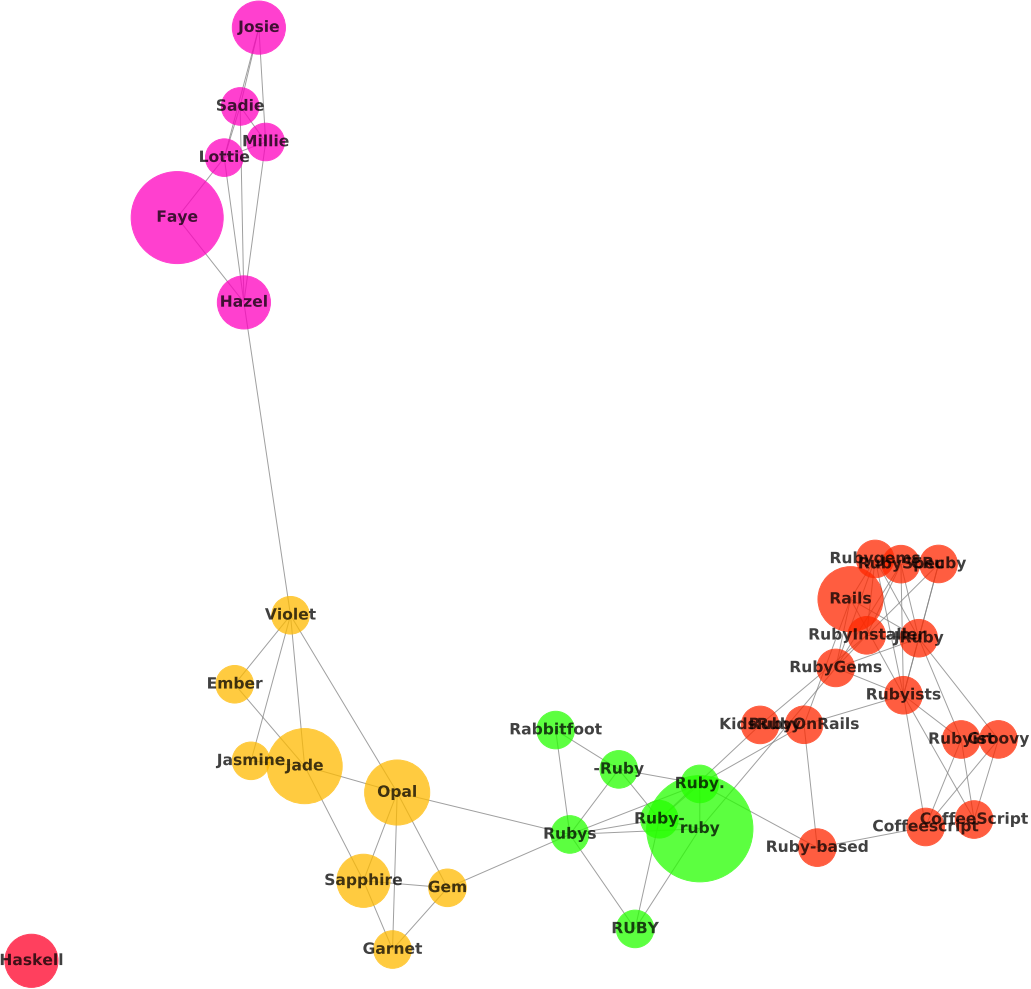


Figure 3-3: The graph of nearest neighbours of the word *Ruby* separated according several senses: programming languages, female names, gems, different spellings of the word *Ruby*. Node size denotes word importance with the largest node in the cluster being used as a keyword to interpret an induced word sense.



# Chapter 4

## Unsupervised Interpretable Word Sense Disambiguation

Materials of this chapter are based on papers [1] and [18] from the list of 14 publications the thesis is based on.

### 4.1 Introduction

In this chapter, an unsupervised, knowledge-free, and interpretable approach to word sense induction and disambiguation is proposed building on top of techniques presented in the previous two chapters.

The task formulation here as following. Input is (i) a **word sense inventory**  $S$ , and (ii) a **mention** of a word  $v$  in a **context**  $C$ . The output is (i) a **word sense identifier** of the word  $v$  corresponding to the mention in the context  $C$ , and (ii) a **human-readable representation** of the word sense  $s$ .

The current trend in NLP is the use of highly opaque models, e.g. neural networks and word embeddings. While these models yield state-of-the-art results on a range of tasks, their drawback is poor interpretability. On the example of word sense induction and disambiguation (WSID), it is shown that it is possible to develop an interpretable model that matches the state-of-the-art models in accuracy. Namely, an unsupervised, knowledge-free WSID approach is presented, which is interpretable at three levels: word sense inventory, sense feature representations, and disambiguation procedure. Experiments show that our

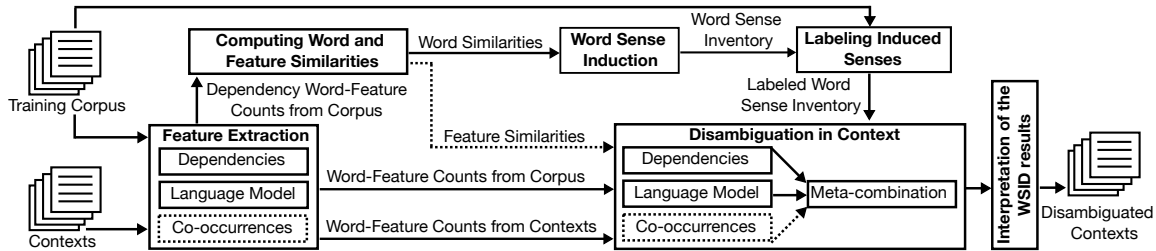


Figure 4-1: Outline of unsupervised interpretable method for word sense induction and disambiguation.

model performs on par with state-of-the-art word sense embeddings and other unsupervised systems while offering the possibility to justify its decisions in human-readable form.

A word sense disambiguation (WSD) system takes as input a target word  $t$  and its context  $C$ . The system returns an identifier of a word sense  $s_i$  from the word sense inventory  $\{s_1, \dots, s_n\}$  of  $t$ , where the senses are typically defined manually in advance. Despite significant progress in methodology during the two last decades [58–60], WSD is still not widespread in applications [61], which indicates the need for further progress. The difficulty of the problem largely stems from the lack of domain-specific training data. A fixed sense inventory, such as the one of WordNet [62], may contain irrelevant senses for the given application and at the same time lack relevant domain-specific senses. Word sense induction from domain-specific corpora is a supposed to solve this problem. However, most approaches to word sense induction and disambiguation, e.g. [63–65], rely on clustering methods and dense vector representations that make a WSD model uninterpretable as compared to knowledge-based WSD methods.

Interpretability of a statistical model is important as it lets us understand the reasons behind its predictions [66–68]. Interpretability of WSD models (1) lets a user understand why in the given context one observed a given sense (e.g., for educational applications); (2) performs a comprehensive analysis of correct and erroneous predictions, giving rise to improved disambiguation models.

The contribution of this chapter is an interpretable unsupervised knowledge-free WSD method. The novelty of our method is in (1) a technique to disambiguation that relies on induced inventories as a pivot for learning sense feature representations, (2) a technique for making induced sense representations interpretable by labeling them with hypernyms and images.

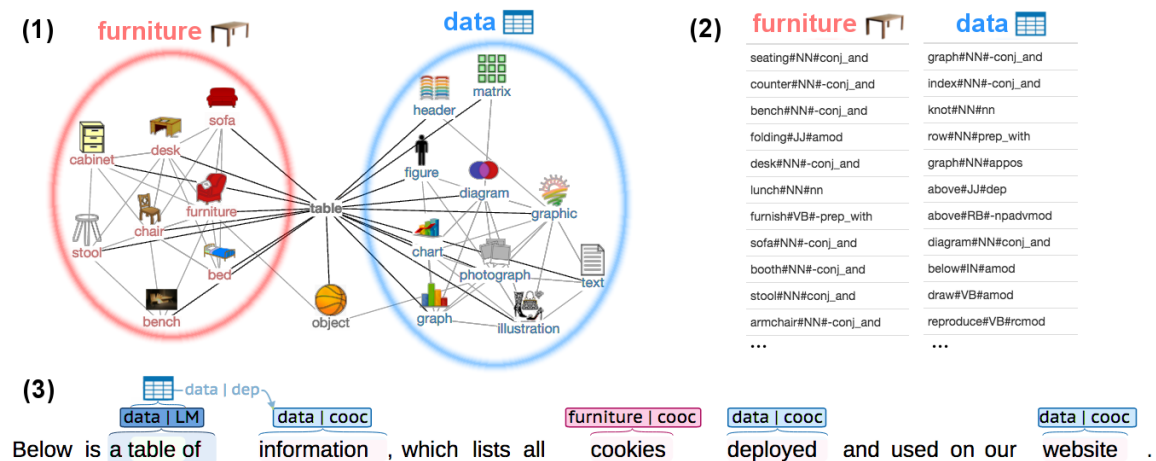


Figure 4-2: Interpretation of the senses of the word “table” at three levels by our method: (1) word sense inventory; (2) sense feature representation; (3) results of disambiguation in context. The sense labels (“furniture” and “data”) are obtained automatically based on cluster labeling with hypernyms. The images associated with the senses are retrieved using a search engine: “table data” and “table furniture”.

## 4.2 Method

Our unsupervised word sense disambiguation method consist of the five steps illustrated in Figure 4-1: extraction of context features (Section 3.1); computing word and feature similarities; word sense induction; labeling of clusters with hypernyms and images, disambiguation of words in context based on the induced inventory, and finally interpretation of the model. Feature similarity and co-occurrence computation steps (drawn with a dashed lines) are optional, since they did not consistently improve performance.

**Extraction of Context Features** The goal of this step is to extract word-feature counts from the input corpus. In particular, we extract features based on dependencies, word co-occurrences, and statistical language models. Also graph of word and feature similarities are computed and used.

**Word Sense Induction** We induce a sense inventory by clustering of ego-network of similar words. In our case, an inventory represents senses by a word cluster, such as “chair, bed, bench, stool, sofa, desk, cabinet” for the “furniture” sense of the word “table”. The sense induction processes one word  $t$  of the distributional thesaurus  $T$  per iteration. First, we retrieve nodes of the ego-network  $G$  of  $t$  being the  $N$  most similar words of  $t$  according

to  $T$  (see Figure 4-2 (1)). Note that the target word  $t$  itself is not part of the ego-network. Second, we connect each node in  $G$  to its  $n$  most similar words according to  $T$ . Finally, the ego-network is clustered with Chinese Whispers [54]. The  $n$  parameter regulates the granularity of the inventory: we experiment with  $n \in \{200, 100, 50\}$  and  $N = 200$ .

**Labeling Induced Senses with Hypernyms and Images** To improve interpretability of induced senses, we assign an image to each word in the cluster (see Figure 4-2) by querying the Bing image search API using the query composed of the target word and its hypernym, e.g. “jaguar car”. The first hit of this query is selected to represent the induced word sense.

---

**Algorithm 3** Unsupervised WSD with induced word sense inventory.

---

**input** : Word  $t$ , context features  $C$ , sense inventory  $I$ , word-feature table  $F$ , use largest cluster back-off  $LCB$ , use feature expansion  $FE$ .

**output**: Sense of the target word  $t$  in inventory  $I$  and confidence score.

```

10  $S \leftarrow \text{getSenses}(I, t)$ 
    if  $FE$  then
11    $C \leftarrow \text{featureExpansion}(C)$ 
12 end
13 foreach  $(\text{sense}, \text{cluster}) \in S$  do
14    $\alpha[\text{sense}] \leftarrow \{\}$ 
    foreach  $w \in \text{cluster}$  do
15     foreach  $c \in C$  do
16        $\alpha[\text{sense}] \leftarrow \alpha[\text{sense}] \cup F(w, c)$ 
17     end
18   end
19 end
20 if  $\max_{\text{sense} \in S} \text{mean}(\alpha[\text{sense}]) = 0$  then
21   if  $LCB$  then
22     return  $\arg \max_{(\text{cluster}) \in S} |\text{cluster}|$ 
23   else
24     return  $-1$  // reject to classify
25   end
26 else
27   return  $\arg \max_{(\text{sense}, \cdot) \in S} \text{mean}(\alpha[\text{sense}])$ 
28 end

```

---

**WSD with Induced Word Sense Inventory** To disambiguate a target word  $t$  in context, we extract context features  $C$  and pass them to Algorithm 3. We use the induced sense inventory  $I$  and select the sense that has the largest weighted feature overlap with context features or fall back to the largest cluster back-off when context features  $C$  do not match the learned sense representations.

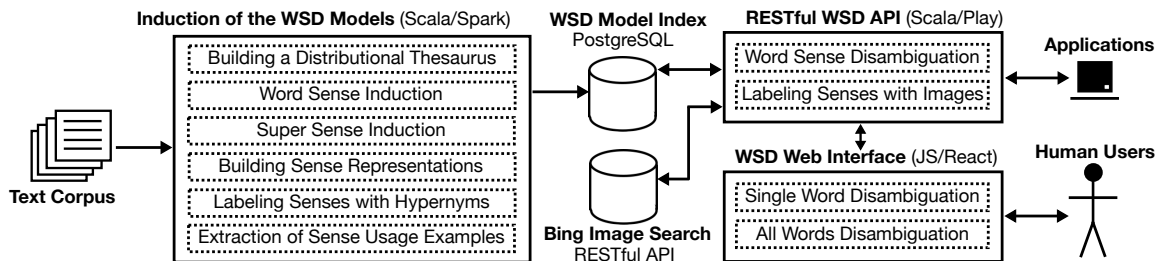


Figure 4-3: Software and functional architecture of the WSD system.

The algorithm starts by retrieving induced sense clusters of the target word (line 1). Next, the method starts to accumulate context feature weights of each *sense* in  $\alpha[sense]$ . Each word  $w$  in a *sense cluster* brings all its word-feature counts  $F(w, c)$ : see lines 5-12. Finally, a *sense* that maximizes mean weight across all context features is chosen (lines 13-21). Optionally, we can resort to the largest cluster back-off (LCB) strategy in case if no context features match sense representations.

## 4.3 Results

In experiments, two lexical sample collections suitable for evaluation of unsupervised WSD systems are used: Turk Bootstrap Word Sense Inventory (TWSI) dataset introduced by [69] and SemEval 2013 word sense induction dataset by [70].

The presented method yields performance comparable to the state-of-the-art unsupervised systems, including two methods based on word sense embeddings. Note, however, that none of the rivalling systems has a comparable level of interpretability to our approach. Further details of experimental results can be found in [1] and [18].

In addition to experimental results, an open source implementation of the method featuring a web demo of several pre-trained models was released.<sup>1</sup> System architecture is illustrated in Figure 4-3: it features API and a web application with user interface for interpretable WSD and sense inventory navigation. The application performs human-interpretable disambiguation of a text entered by a user in single word disambiguation mode (cf. Figure 4-4) and all words disambiguation mode (cf. Figure 4-5).

<sup>1</sup><http://www.jobimtext.org/wsd>

Sentence  
Jaguar is a large spotted predator of tropical America similar to the leopard. (A)


Word  
Jaguar (B)

Model  
Word Senses based on Cluster Word Features (C)

PREDICT SENSE    RANDOM SAMPLE

---

**Predicted senses for 'Jaguar'**

 1. jaguar (animal)  
Similarity score: 0.00184 / Confidence: 99.87% / Sense ID: jaguar#0 / BabelNet ID: bn:00033987n

Hypernyms: animal (D), wildlife, bird, mammal

Sample sentences  
The **jaguar**, a compact and well-muscled animal, is the largest cat in the New World.  
**Jaguar** may leap onto the back of the prey and sever the cervical vertebrae, immobilizing the target.

Cluster words (i)  
lion tiger leopard wolf monkey otter crocodile alligator deer cat elephant fox eagle owl snake

Context words (i)  
elephant: 0.012 tiger: 0.012 fox: 0.0099 wolf: 0.0097 cub: 0.0086 monkey: 0.0083 leopard: 0.0074 eagle: 0.0062  
den: 0.0043 elk: 0.0040 32078 more not shown

Matching features  
leopard: 0.0011 predator: 0.00040 spotted: 0.00038 large: 0.0000041 similar: 0.0000015 tropical: 5.6e-7 america: 2.0e-7

BABELNET LINK (F) ^ SHOW LESS (E)

Figure 4-4: Single word disambiguation mode: disambiguation of the word “Jaguar” (B) in the sentence “*Jaguar* is a large spotted predator of tropical America similar to the leopard.” (A) using the WSD disambiguation model based on cluster word features (C). The predicted sense is summarized with a hypernym and an image (D) and further represented with usage examples, semantically related words, and typical context clues. Each of these elements is extracted automatically. The reasons of the predictions are provided in terms of common sparse features of the input sentence and a sense representation (E). The induced senses are linked to BabelNet (F).

Sentence  
Jaguar is a large spotted predator of tropical America similar to the leopard. (A)

Model  
Word Senses based on Cluster Word Features (C)

DISAMBIGUATE SENTENCE    RANDOM SAMPLE

---

**Detected Entities**  
The system has detected these entities in the given sentence.




 animal Jaguar (D)	is a large spotted	 animal predator (D)	of tropical	 country America (D)
---	--------------------	--	-------------	---

Figure 4-5: All words disambiguation mode: results of disambiguation of nouns.

# Chapter 5

## Linking Word Sense

## Representations

Materials of this chapter are based on papers [7,8] from the list of 14 publications the thesis is based on.

### 5.1 Introduction

In this chapter, methods for linking of word sense representations induced automatically from text using the methods presented in the previous chapters to lexical-semantic networks constructed manually, e.g. Wordnets, are presented. The linking is performed on the level of individual word senses.

The task of **word sense linking** is as following. Inputs are (i) a **manually-constructed** lexical-semantic graph  $W$ , e.g. WordNet, and (ii) **distributionally-induced** lexical-semantic graph, e.g. the one introduced in the previous chapter,  $T = \{(j, R_j, H_j)\}$ , where  $j$  is a sense identifier, i.e. `mouse:1`,  $R_j$  the set of its semantically related senses, i.e. `{keyboard:1, computer:0, ...}`,  $H_j$  the set of its hypernym senses, i.e. `{equipment:3, ...}`. Output is a **mapping**  $M$ : a set of pairs e.g.  $(source, target)$  where  $source \in T.senses$  is a sense of  $T$  and  $target \in W.senses \cup source$  is the most suitable sense of  $W$ .

We present an approach to combining distributional semantic representations induced from text corpora with manually constructed lexical-semantic networks. While both kinds

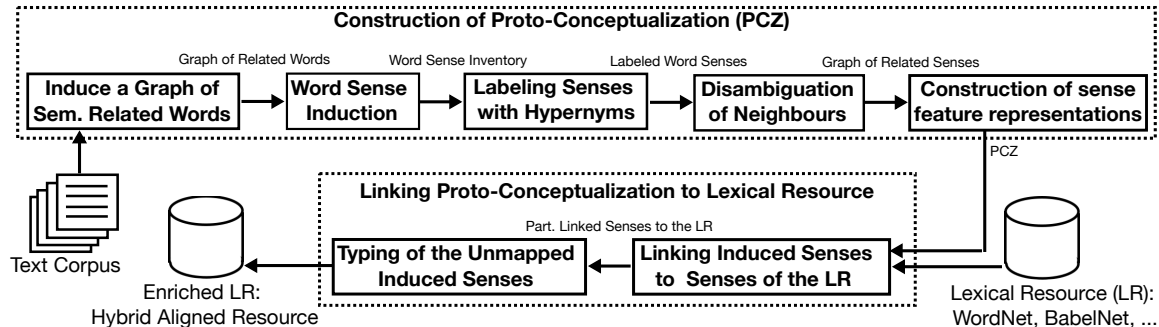


Figure 5-1: Overview of the proposed framework for enriching lexical resources: a distributional semantic model is used to construct a disambiguated distributional lexical semantic network (a proto-conceptualization, PCZ), which is subsequently linked to the lexical resource.

of semantic resources are available with high lexical coverage, our aligned resource combines the domain specificity and availability of contextual information from distributional models with the conciseness and high quality of manually crafted lexical networks. We start with a distributional representation of induced senses of vocabulary terms, which are accompanied with rich context information given by related lexical items. We then automatically disambiguate such representations to obtain a full-fledged proto-conceptualization, i.e. a typed graph of induced word senses. In a final step, this proto-conceptualization is aligned to a lexical ontology, resulting in a hybrid aligned resource. Moreover, unmapped induced senses are associated with a semantic type in order to connect them to the core resource.

## 5.2 Method

The construction of our hybrid aligned resource (HAR) builds upon methods used to link various manually constructed lexical resources to construct BabelNet [71] and UBY [72], among others. In our method, however, linking is performed between two networks that are structurally similar, but have been constructed in two completely different ways: one resource is built using an unsupervised bottom-up approach from text corpora, while the second is constructed in a top-down manner using manual labor, e.g., codified knowledge from human experts such as lexicographers (WordNet). In particular, the method consists of two major phases, as illustrated in Figure 5-1.

The upper part correspond to the method described in the chapters above i.e. building



---

**Algorithm 4** Linking induced senses to senses of a lexical resource.

---

**Input:**  $T = \{(j_i, R_{j_i}, H_{j_i})\}$ ,  $W$ ,  $th$ ,  $m$   
**Output:**  $M = (source, target)$

- 1:  $M = \emptyset$
- 2: **for all**  $(j_i, R_{j_i}, H_{j_i}) \in T.monosemousSenses$  **do**
- 3:    $C(j_i) = W.getSenses(j_i.lemma, j_i.POS)$
- 4:   **if**  $|C(j_i)| == 1$ , let  $C(j_i) = \{c_0\}$  **then**
- 5:     **if**  $sim(j_i, c_0, \emptyset) \geq th$  **then**
- 6:        $M = M \cup \{(j_i, c_0)\}$
- 7: **for**  $step = 1$ ;  $step \leq m$  ;  $step = step + 1$  **do**
- 8:    $M_{step} = \emptyset$
- 9:   **for all**  $(j_i, R_{j_i}, H_{j_i}) \in T.senses/M.senses$  **do**
- 10:      $C(j_i) = W.getSenses(j_i.lemma, j_i.POS)$
- 11:     **for all**  $c_k \in C(j_i)$  **do**
- 12:        $rank(c_k) = sim(j_i, c_k, M)$
- 13:       **if**  $rank(c_k)$  has a single top value for  $c_t$  **then**
- 14:         **if**  $rank(c_t) \geq th$  **then**
- 15:          $M_{step} = M_{step} \cup \{(j_i, c_t)\}$
- 16:      $M = M \cup M_{step}$
- 17: **for all**  $(j_i, R_{j_i}, H_{j_i}) \in T.senses/M.senses$  **do**
- 18:    $M = M \cup \{(j_i, j_i)\}$
- 19: **return**  $M$

---

interpretable word sense representations from text in an unsupervised manner. Below we describe how a corpus-induced semantic network (a proto-conceptualization) is linked to a manually created semantic network, represented by a lexical resource.

### Linking Induced Senses to Senses of the Lexical Resource

We link each sense in our proto-conceptualization (PCZ) to the most suitable sense (if any) of a Lexical Resource (LR, see Figure 5-1 step 3). There exist many algorithms for knowledge base linking [73]: here, we build upon simple, yet high-performing previous approaches to linking LRs that achieved state-of-the-art performance. These rely at their core on computing the overlap between the bags of words built from the LRs' concept lexicalizations, e.g., [72, 74] (*inter alia*). Specifically, we develop i) an iterative approach – so that the linking can benefit from the availability of linked senses from previous iterations – ii) leveraging the lexical content of the source and target resources. Algorithm 4 takes as input:

1. a PCZ  $T = \{(j_i, R_{j_i}, H_{j_i})\}$  where  $j_i$  is a sense identifier (i.e. `mouse:1`),  $R_{j_i}$  the set of its semantically related senses (i.e.  $R_{j_i} = \{\text{keyboard:1, computer:0, ...}\}$  and  $H_{j_i}$  the set of its hypernym senses (i.e.  $H_{j_i} = \{\text{equipment:3, ...}\}$ ;

2. a LR  $W$ : we experiment with: WordNet, a lexical database for English and BabelNet, a very large multilingual ‘encyclopedic dictionary’;
3. a threshold  $th$  over the similarity between pairs of concepts and a number  $m$  of iterations as a stopping criterion.

The algorithm outputs a mapping  $M$ , which consists of a set of pairs of the kind  $(source, target)$  where  $source \in T.senses$  is a sense of the input PCZ  $T$  and  $target \in W.senses \cup source$  is the most suitable sense of  $W$  or  $source$  when no such sense has been identified.

The algorithm starts by creating an empty mapping  $M$  (line 1). Then for each monosemous sense (e.g., Einstein:0 is the only sense in the PCZ for the term Einstein) it searches for a candidate monosemous sense (lines 2-6). If such monosemous candidate senses exist (line 4), we compare the two senses (line 5) with the following similarity function:

$$sim(j, c, M) = \frac{|T.BoW(j, M, W) \cap W.BoW(c)|}{|T.BoW(j, M, W)|}, \quad (5.1)$$

where

1.  $T.BoW(j, M, W)$  is the set of words containing all the terms extracted from related/hypernym senses of  $j$  and all the terms extracted from the related/hypernym (i.e., already linked in  $M$ ) synsets in  $W$ . For each synset from the LR, we use all synonyms and content words of the gloss.
2.  $W.BoW(c)$  contains the synonyms and the gloss content words for the synset  $c$  and all the related synsets of  $c$ .

Then a new link pair  $(j_i, c_0)$  is added to  $M$  if the similarity score between  $j_i$  and  $c_0$  meets or exceeds the threshold  $th$  (line 5). At this point, we collected a first set of disambiguated (monosemous) senses in  $M$  and start to iteratively disambiguate the remaining (polysemous) senses in  $T$  (lines 7-16). This iterative disambiguation process is similar to the one we described for the monosemous case (lines 2-6), with the main difference that, due to the polysemy of the candidates synsets, we instead use the similarity function to rank all candidate senses (lines 11-12) and select the top-ranked candidates for the mapping (lines 13-15). At the end of each iteration, we add all collected pairs to  $M$  (line

---

**Algorithm 5** Typing of the unmapped induced senses.

---

**Input:**  $M = (source, target), W$ **Output:**  $H = (source, type)$ 

```
1:  $H = \emptyset$ 
2: for all  $(source, target) \in M$  do
3:   if  $target \notin W$  then
4:      $Rank = 0$ 
5:     for all  $related \in R_{source}, \exists(related, trelated) \in M, trelated \in W$  do
6:       for all  $hop \in (1, 2, 3)$  do
7:         for all  $ancestor \in W.ancestors(trelated, hop)$  do
8:            $Rank(ancestor) = Rank(ancestor) + 1.0/hop$ 
9:         for all  $ntype \in Rank.top(top_h)$  do
10:         $H = H \cup (source, ntype)$ 
11: return  $H$ 
```

---

16). Finally, all unlinked  $j$  of  $T$ , i.e. induced senses that have no corresponding LR sense, are added to the mapping  $M$  (lines 17- 18).

### Typing of the Unmapped Induced Senses

An approach based on the bag-of-words from concept lexicalizations has the advantage of being simple, as well as high performing as we show later in the evaluation – cf. also findings from [74]. However, there could be still PCZ senses that cannot be mapped to the target lexical resource, e.g., because of vocabulary mismatches, sparse concepts’ lexicalizations, or because they are simply absent in the resource.

Consequently, in the last phase of our resource creation pipeline we link these ‘orphan’ PCZ senses (i.e., those from lines 17-18 of Algorithm 4), in order to obtain a unified resource, and propose a method to infer the type of those concepts that were not linked to the target lexical resource. For example, so far we were not able to find a BabelNet sense for the PCZ item Roddenberry:10 (the author of ‘Star Trek’). However, by looking at the linked related concepts that share the same BabelNet hypernym – e.g. the PCZ items Asimov:3 *is-a* author<sub>BabelNet</sub>, Tolkien:7 *is-a* author<sub>BabelNet</sub>, Heinlein:8 *is-a* author<sub>BabelNet</sub>, etc. – we can infer that Roddenberry:10 *is-a* author:1, since the latter was linked to the Babel synset author<sub>BabelNet</sub>.

The input of Algorithm 5 consist of the mapping  $M$  of a PCZ to a lexical resource  $W$  (cf. Algorithm 4). The output is a new mapping  $H$  containing pairs of the kind  $(source, type)$  where  $type$  is a type in  $W$  for the concept  $source \in PCZ$ . We first initialize the new mapping  $H$  as an empty set (line 1). Then for all the pairs  $(source, target)$  where the target is a

PCZ ID	WordNet ID	PCZ Related Terms	PCZ Context Clues
mouse:0	mouse:wn1	rat:0, rodent:0, monkey:0, ...	rat:conj_and, gray:amod, ...
mouse:1	mouse:wn4	keyboard:1, computer:0, printer:0 ...	click:-prep_of, click:-nn, ....
keyboard:0	keyboard:wn1	piano:1, synthesizer:2, organ:0 ...	play:-dobj, electric:amod, ..
keyboard:1	keyboard:wn1	keypad:0, mouse:1, screen:1 ...	computer, qwerty:amod ...

Table 5.1: Sample entries of the hybrid aligned resource (HAR) for the words *mouse* and *keyboard*. Trailing numbers indicate sense identifiers. To enrich WordNet sense representations we rely on related terms and context clues.

concept not included in the target lexical resource  $W$  (line 3), we compute a rank of all the ancestors of each related sense that has a counterpart *trelated* in  $W$  (lines 5-8). In other words, starting from linked related senses *trelated*, we traverse the taxonomy hierarchy (at most for 3 hops) in  $W$  and each time we encounter a sense *ancestor* we increment its rank by the inverse of the distance to *trelated*. Finally we add the pairs  $(source, ntype)$  to  $H$  for all the *ntype* at the top  $top_h$  in the *Rank*.

Finally, our final resource consists of: i) the proto-conceptualization (PCZ); ii) the mapping  $M$  of PCZ entries to the lexical resource (e.g., WordNet or BabelNet); iii) the mapping  $H$  of suggested types for the PCZ entries not mapped in  $M$ .

## 5.3 Results

Manual evaluations against ground-truth judgments for different stages of our method as well as an extrinsic evaluation on a knowledge-based word sense disambiguation benchmark all indicate the high quality of the new hybrid resource. Additionally, we show the benefits of enriching top-down lexical knowledge resources with bottom-up distributional information from text for addressing high-end knowledge acquisition tasks such as cleaning hypernym graphs and learning taxonomies from scratch.

Examples of the linked word senses are provided in Table 5.1 between Wordnet constructed manually and corpus-induced sense repository from sparse count-based distributional model (PCZ). Further examples, experimental results and their analysis can be found in [7, 8].

# Chapter 6

## Prediction of Hypernym Embeddings

Materials of this chapter are based on paper [4] from the list of 14 publications the thesis is based on.

### 6.1 Introduction

In this section, a method for extraction of hypernyms based on projection learning and word embeddings is presented.

Hypernymy is a hierarchical semantic relation between terms, such as (apple, fruit) or (jaguar, animal). In the former example, the word **apple** is a **hyponym**, and the word **fruit** is a **hypernym**. The task we consider in this chapter is given a hyponym to predict its corresponding hypernym(s).

In contrast to classification-based approaches, projection-based methods require no candidate hyponym-hypernym pairs. While it is natural to use both positive and negative training examples in supervised relation extraction, the impact of negative examples on hypernym prediction was not studied so far. In this chapter, it is shown that explicit negative examples used for regularization of the model significantly improve performance compared to the state-of-the-art approach of [75] on three datasets from different languages.

## 6.2 Method

The approach performs hypernymy extraction via regularized projection learning.

**Baseline Approach** The model of [75] is used as the baseline. In this approach, the projection matrix  $\Phi^*$  is obtained similarly to the linear regression problem, i.e., for the given row word vectors  $\vec{x}$  and  $\vec{y}$  representing correspondingly hyponym and hypernym, the square matrix  $\Phi^*$  is fit on the training set of positive pairs  $\mathcal{P}$ :

$$\Phi^* = \arg \min_{\Phi} \frac{1}{|\mathcal{P}|} \sum_{(\vec{x}, \vec{y}) \in \mathcal{P}} \|\vec{x}\Phi - \vec{y}\|^2, \quad (6.1)$$

where  $|\mathcal{P}|$  is the number of training examples and  $\|\vec{x}\Phi - \vec{y}\|$  is the distance between a pair of row vectors  $\vec{x}\Phi$  and  $\vec{y}$ . In the original method, the  $L^2$  distance is used. To improve performance,  $k$  projection matrices  $\Phi$  are learned one for each cluster of relations in the training set. One example is represented by a hyponym-hypernym offset. Clustering is performed using the  $k$ -means algorithm [76].

**Linguistic Constraints via Regularization** The nearest neighbors generated using distributional word vectors tend to contain a mixture of synonyms, hypernyms, co-hyponyms and other related words [77–79]. In order to explicitly provide examples of undesired relations to the model, we propose two improved versions of the baseline model: *asymmetric regularization* that uses inverted relations as negative examples, and *neighbor regularization* that uses relations of other types as negative examples. For that, we add a regularization term to the loss function:

$$\Phi^* = \arg \min_{\Phi} \frac{1}{|\mathcal{P}|} \sum_{(\vec{x}, \vec{y}) \in \mathcal{P}} \|\vec{x}\Phi - \vec{y}\|^2 + \lambda R, \quad (6.2)$$

where  $\lambda$  is the constant controlling the importance of the regularization term  $R$ .

**Asymmetric Regularization.** As hypernymy is an asymmetric relation, our first method enforces the asymmetry of the projection matrix. Applying the same transformation to the predicted hypernym vector  $\vec{x}\Phi$  should not provide a vector similar ( $\cdot$ ) to the initial hyponym vector  $\vec{x}$ . Note that, this regularizer requires only positive examples  $\mathcal{P}$ :

$$R = \frac{1}{|\mathcal{P}|} \sum_{(\vec{x}, \cdot) \in \mathcal{P}} (\vec{x}\Phi\Phi \cdot \vec{x})^2. \quad (6.3)$$

**Neighbor Regularization.** This approach relies on the negative sampling by explicitly providing the examples of semantically related words  $\vec{z}$  of the hyponym  $\vec{x}$  that penalizes the matrix to produce the vectors similar to them:

$$R = \frac{1}{|\mathcal{N}|} \sum_{(\vec{x}, \vec{z}) \in \mathcal{N}} (\vec{x}\Phi\Phi \cdot \vec{z})^2. \quad (6.4)$$

This regularizer requires negative samples  $\mathcal{N}$ . In our experiments, we use synonyms of hyponyms as  $\mathcal{N}$ , but other types of relations can be also used such as antonyms, meronyms or co-hyponyms. Certain words might have no synonyms in the training set. In such cases, we substitute  $\vec{z}$  with  $\vec{x}$ , gracefully reducing to the previous variation. Otherwise, on each training epoch, we sample a random synonym of the given word.

**Regularizers without Re-Projection.** In addition to the two regularizers described above, that rely on re-projection of the hyponym vector ( $\vec{x}\Phi\Phi$ ), we also tested two regularizers without re-projection, denoted as  $\vec{x}\Phi$ . The neighbor regularizer in this variation is defined as follows:

$$R = \frac{1}{|\mathcal{N}|} \sum_{(\vec{x}, \vec{z}) \in \mathcal{N}} (\vec{x}\Phi \cdot \vec{z})^2. \quad (6.5)$$

In our case, this regularizer penalizes relatedness of the predicted hypernym  $\vec{x}\Phi$  to the synonym  $\vec{z}$ . The asymmetric regularizer without re-projection is defined in a similar way.

**Training of the Models** To learn parameters of the considered models we used the Adam method [80] with the default meta-parameters as implemented in the TensorFlow framework [81]. We ran 700 training epochs passing a batch of 1024 examples to the optimizer. Projection matrices were initialized using the normal distribution  $\mathcal{N}(0, 0.1)$ .

## 6.3 Results

Evaluation of the proposed methods was done for one Russian and two English hypernymy datasets. The experiments in the context of the hypernymy prediction task for both languages show significant improvements of the proposed approach over the state-of-the-art model without negative sampling. Further experimental results and their analysis can be found in [4].

# Chapter 7

## Extracting of Hypernyms via Sense Graph Clustering

Materials of this chapter are based on papers [19] from the list of 14 publications the thesis is based on.

### 7.1 Introduction

In this chapter, it is shown how distributionally-induced semantic classes can be helpful for extracting hypernyms.

The task considered in this chapter is given a set of **noisy hypernyms**  $H = \{(w_i, w_j), (w_k, w_l), \dots, (w_y, w_z)\}$  to obtain an updated set of **cleansed hypernyms**  $H'$  which (i) do not contain wrong relations, (ii) adds missing correct relations .

Methods for inducing sense-aware semantic classes using distributional semantics are presented below. These classes are used for filtering noisy hypernymy relations. Denoising of hypernyms is performed by labelling each semantic class with its hypernyms. On the one hand, this allows us to filter out wrong extractions using the global structure of distributionally similar senses. On the other hand, we infer missing hypernyms via label propagation to cluster terms. We conduct a large-scale crowdsourcing study showing that processing of automatically extracted hypernyms using our approach improves the quality of the hypernymy extraction in terms of both precision and recall. Furthermore, the utility of the method is demonstrated in the domain taxonomy induction task, achieving



the state-of-the-art results on a SemEval-2016 task on taxonomy induction.

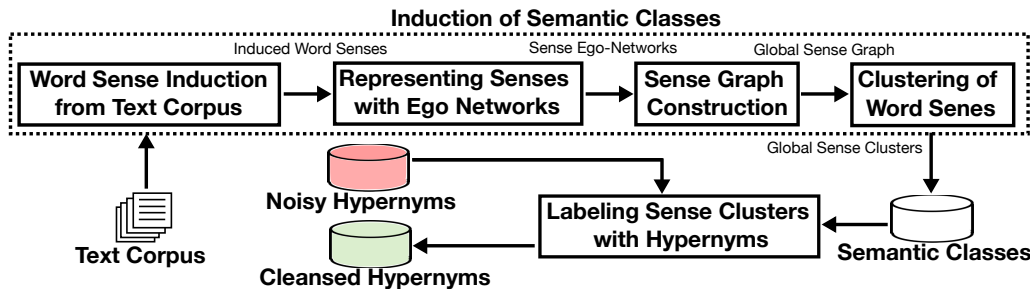


Figure 7-1: Outline of the approach: sense-aware distributional semantic classes are induced from a text corpus and then used to filter noisy hypernyms database (e.g. extracted by an external method from a text corpus).

## 7.2 Method

In this section, a method for unsupervised induction of distributional sense-aware semantic classes is presented. Also we show how to perform denoising of hypernyms using the induced structures. As illustrated in Figure 7-1, our method induces a sense inventory from a text corpus using the method of [82, 83], and clusters these senses.

Sample word senses from the induced semantic clusters are presented in Table 7.1. The difference of the induced sense inventory from the sense clustering is that word senses in the induced resource are specific to a given target word, e.g. words “apple” and “mango” have distinct “fruit” senses, represented by a list of related senses. On the other hand, Sense clusters represent a global and not a local clustering of senses, i.e. the “apple” in the “fruit” sense can be a member of only one cluster. This is similar to WordNet, where one sense can only belong to a single synset. Below we describe each step of our method.

### Word Sense Induction from a Text Corpus

Each word sense  $s$  in the induced sense inventory  $\mathcal{S}$  is represented by a list of neighbors  $\mathcal{N}(s)$ . Extraction of this network is performed using the method of [82] and involves three steps: (1) building a distributional thesaurus, i.e. a graph of related ambiguous terms [84]; (2) word sense induction via clustering of ego networks [85, 86] of related words using the Chinese Whispers graph clustering algorithm [46]; (3) disambiguation of related words and hypernyms. The word sense inventory used in our experiment was extracted from a 9.3 billion tokens corpus, which is a concatenation of Wikipedia, ukWac [87], LCC [88] and

Global Sense Cluster: Semantic Class, $c \subset \mathcal{S}$	Hypernyms, $\mathcal{H}(c) \subset \mathcal{S}$
peach#1, banana#1, pineapple#0, berry#0, blackberry#0, grape-fruit#0, strawberry#0, blueberry#0, fruit#0, grape#0, melon#0, orange#0, pear#0, plum#0, raspberry#0, watermelon#0, apple#0, apricot#0, watermelon#0, pumpkin#0, berry#0, <b>mangosteen#0</b> , ...	vegetable#0, fruit#0, crop#0, ingredient#0, food#0, .
C#4, Basic#2, Haskell#5, Flash#1, Java#1, Pascal#0, Ruby#6, PHP#0, Ada#1, Oracle#3, Python#3, Apache#3, Visual Basic#1, ASP#2, Delphi#2, SQL Server#0, CSS#0, AJAX#0, JavaScript#0, SQL Server#0, Apache#3, Delphi#2, Haskell#5, .NET#1, CSS#0, ...	programming language#3, technology#0, language#0, format#2, app#0

Table 7.1: Sample of the induced sense clusters representing “fruits” and “programming language” semantic classes. Similarly to the induced word senses, the semantic classes are labeled with hypernyms. In contrast to the induced word senses, which represent a local clustering of word senses (related to a given word) semantic classes represent a global sense clustering of word senses.

Gigaword [89]. Note that analogous graphs of senses can be obtained using word sense embeddings, see [65, 90]. Similarly to other distributional word graphs, the induced sense inventory sense network is scale-free, cf. [91]. Experiments show that a global clustering of this network can lead to a discovery of giant components, which are useless in our context as they represent no semantic class. To overcome this problem, we re-build the sense network.

### Representing Senses with Ego Networks

To perform a global clustering of senses, we represent each induced sense  $s$  by a second-order *ego network* [86]. An ego network is a graph consisting of all related senses  $\mathcal{R}(s)$  of the ego sense  $s$  reachable via a path of length one or two, defined as:

$$\{s_j : (s_j \in \mathcal{N}(s)) \vee (s_i \in \mathcal{N}(s) \wedge s_j \in \mathcal{N}(s_i))\}. \quad (7.1)$$

Each edge weight  $\mathcal{W}_s(s_i, s_j)$  between two senses is taken from the induced sense inventory network [82] and is equal to a distributional semantic relatedness score between  $s_i$  and  $s_j$ .

Senses in the induced sense inventory may contain a mixture of different senses introducing noise in a global clustering, e.g. “Python” in the animal sense is related to both car and snake senses. To minimize the impact of the word sense induction errors, we filter out ego networks with a highly segmented structure. Namely, we cluster each ego network with the Chinese Whispers algorithm and discard networks for which the cluster containing the target sense  $s$  contains less than 80% nodes of the respective network to

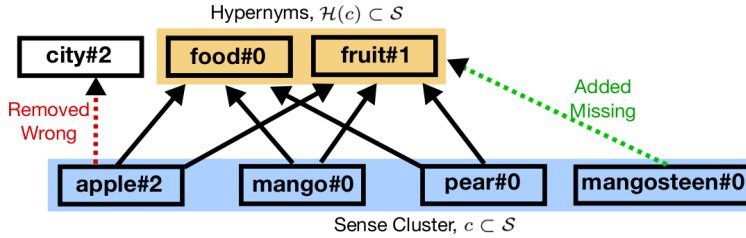


Figure 7-2: Post-processing of hypernymy relations using distributionally induced semantic classes, represented by clusters of induced word senses labeled with hypernyms. The word postfix, such as #1, is a sense ID. The wrong hypernyms outside the cluster labels are removed, while the missing ones not present in the noisy database of hypernyms are added.

ensure semantic coherence inside the word groups. Besides, all nodes of a network not appearing in the cluster containing the ego sense  $s$  are also discarded.

### Global Sense Graph Construction

The goal of this step is to merge ego networks of individual senses constructed at the previous step into a global graph. We compute weights of the edges of the global graph by counting the number of co-occurrences of the same edge in different networks:

$$\mathcal{W}(s_i, s_j) = \sum_{s \in \mathcal{S}} \mathcal{W}_s(s_i, s_j). \quad (7.2)$$

For filtering out noisy edges, we remove all edges with the weight less than a threshold  $t$ . Finally, we apply the function  $E(w)$  that re-scales edge weights. We tested identity function (count) and the natural logarithm (log):

$$\mathcal{W}(s_i, s_j) = \begin{cases} E(\mathcal{W}(s_i, s_j)) & \text{if } \mathcal{W}(s_i, s_j) \geq t, \\ 0 & \text{otherwise.} \end{cases} \quad (7.3)$$

### Clustering of Word Senses

The core of our method is the induction of semantic classes by clustering the global graph of word senses. We use the Chinese Whispers algorithm to make every sense appear only in one cluster  $c$ . Results of the algorithm are groups of strongly related word senses that represent different concepts.

We use two clustering versions in our experiments: the *fine-grained* model clusters 208,871 induced word senses into 1,870 semantic classes, and the *coarse-grained* model that

groups 18,028 word senses into 734 semantic classes. To find optimal parameters of our method, we compare the induced labeled sense clusters to lexical semantic knowledge from WordNet 3.1 [92] and BabelNet 3.7 [93].

### **Denoising Hypernyms using the Induced Distributional Semantic Classes**

By labeling the induced semantic classes with hypernyms we can remove wrong ones or add those that are missing as illustrated in Figure 7-2. Each sense cluster is labeled with the hypernyms, where the labels are the common hypernyms of the cluster word (cf. Table 7.1). Hypernyms that label no sense cluster are filtered out. In addition, new hypernyms can be generated as a result of labeling. Additional hypernyms are discovered by propagating cluster labels to the rare words without hypernyms, e.g. “mangosteen” in Figure 7-2. For labeling we used the tf-idf: hypernyms that appear in many senses  $s$  are down-weighted:

$$\text{tf-idf}(h) = \sum_{s \in c} \mathcal{H}(s) \cdot \log \frac{|\mathcal{S}|}{|h \in \mathcal{H}(s) : \forall s \in \mathcal{S}|}, \quad (7.4)$$

where  $\sum_{s \in c} \mathcal{H}(s)$  is a sum of weights for all hypernyms for each sense  $s$ , per each cluster  $c$ . We label each sense cluster  $c$  with its top five hypernyms  $\mathcal{H}(c)$ . Each hypernym is disambiguated using the method of [82]. Namely, we calculate the cosine similarity between the context (the current sense cluster) and the induced senses (local clusters of the ambiguous word).

Distributional representations of rare words, such as “mangosteen” can be less precise than those of frequent words. However, co-occurrence of a hyponym and a hypernym in a single sentence is not required in our approach, while it is the case for the path-based hypernymy extraction methods.

## **7.3 Results**

To evaluate the method three experiments were conducted. A large-scale crowdsourcing study indicated a high plausibility of extracted semantic classes according to human judgment. Besides, it was demonstrated that the approach helps to improve precision and recall of a hypernymy extraction method. Finally, it was shown that semantic classes can be used to improve domain taxonomy induction from text based on the SemEval-2016 dataset. Further experimental results and their analysis can be found in [19].

# Chapter 8

## Taxonomy Enrichment using Hyperbolic Embeddings

Materials of this chapter are based on paper [5] from the list of 14 publications the thesis is based on.

### 8.1 Introduction

The task of taxonomy enrichment is as following. Given a **noisy taxonomic graph**  $G = (V, E)$  where edges  $E$  represent hypernymy relations between words in  $V$  with errors of two types (i) absent edges:  $E_{abs} = \{(v_i, v_j) : v_i \text{ is-a } v_j \wedge (v_i, v_j) \notin E\}$  with a special case of orphan nodes  $V_{orh} = \{v_i \in V : \nexists (v_i, v_j) \in E\}$ ; and (ii) wrong edges:  $E_{wrg} = \{(v_i, v_j) \in E : v_i \text{ not-is-a } v_j\}$  build a **cleansed taxonomic graph**  $G' = (V, E')$  correcting the two edge errors by: (i) adding absent edges:  $E = E \cup \{E_{abs}\}$ ; (ii) removing wrong edges:  $E = E \setminus E_{wrg}$ . For orphan nodes only adding edges is needed. For connected nodes either adding absent additional edge is needed or a relocation if it is wrongly placed. The latter is a combination of removing a wrong edge and adding an absent edge to the deattached node in question.

In this chapter the use of Poincaré embeddings is introduced to improve existing state-of-the-art approaches to domain-specific taxonomy induction from text as a signal for both relocating wrong hyponym terms within a (pre-induced) taxonomy as well as for attaching disconnected terms in a taxonomy. This method substantially improves previous

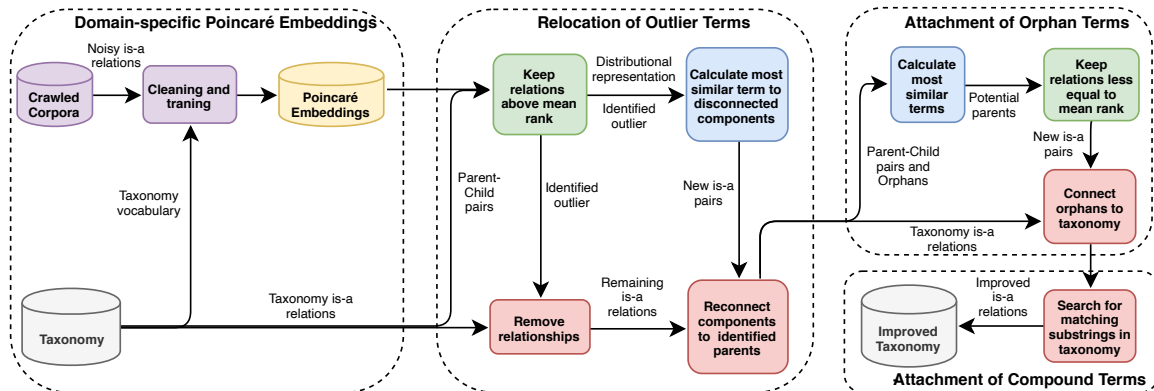


Figure 8-1: Outline of our taxonomy refinement method.

state-of-the-art results on taxonomy extraction. We demonstrate the superiority of Poincaré embeddings over distributional semantic representations, supporting the hypothesis that they can better capture hierarchical lexical-semantic relationships than embeddings in the Euclidean space.

## 8.2 Method

In this section, a method for taxonomy refinement using hyperbolic word embedding is presented. Embeddings using distributional semantics (i.e. `word2vec`) and Poincaré embeddings [94] are used to alleviate the largest error classes in taxonomy extraction: the existence of *orphans* – disconnected nodes that have an overall connectivity degree of zero and *outliers* – a child node that is assigned to a wrong parent. The rare case in which multiple parents can be assigned to a node has been ignored in the proposed refinement system. The first step consists of creating domain-specific Poincaré embeddings. They are then used to identify and relocate outlier terms in the taxonomy, as well as to attach unconnected terms to the taxonomy. In the last step, we further optimize the taxonomy by employing the endocentric nature of hyponyms. See Figure 8-1 for a schematic visualization of the refinement pipeline.

### Training Dataset Construction

To create domain-specific Poincaré embeddings, we use noisy hypernym relationships extracted from a combination of general and domain-specific corpora. For the general domain, we extracted text from English Wikipedia, Gigaword [95], ukWac [96] and LCC news corpora [97]. Noisy IS-A relations are extracted with lexical-syntactic patterns from

all corpora by applying PattaMaika,PatternSim [98], and WebISA [99] following [100].

The extracted noisy relationships of the common and domain-specific corpora are further processed separately and combined afterward. To limit the number of terms and relationships, we restrict the IS-A relationships on pairs for which both entities are part of the taxonomy’s vocabulary. Relations with a frequency of less than three are removed to filter noise. Besides further removing every reflexive relationship, only the more frequent pair of a symmetric relationship is kept. Hence, the set of cleaned relationships is transformed into being antisymmetric and irreflexive.

The same procedure is applied to relationships extracted from the general-domain corpus. They are then used to expand the set of relationships created from the domain-specific corpora.

### Hypernym-Hyponym Distance

Poincaré embeddings are trained on these cleaned IS-A relationships. For comparison, we also trained a model on noun pairs extracted from WordNet (P-WN). Pairs were only kept if both nouns were present in the vocabulary of the taxonomy. Finally, we trained the word2vec embeddings, connecting compound terms in the training corpus (Wikipedia) by ‘\_’ to learn representations for compound terms, i.e multiword units, for the input vocabulary.

In contrast to embeddings in the Euclidean space where the cosine similarity is commonly applied as a similarity measure, i.e.

$$d(\mathbf{u}, \mathbf{v}) = 1 - \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}, \quad (8.1)$$

Poincaré embeddings use a hyperbolic space, specifically the Poincaré ball model [101]. Hyperbolic embeddings are designed for modeling hierarchical relationships between words as they explicitly capture the hierarchy between words in the embedding space and are therefore a natural fit for inducing taxonomies. They were also successfully applied to hierarchical relations in image classification tasks [102]. The distance between two points  $\mathbf{u}, \mathbf{v} \in \mathcal{B}^d$  for a  $d$ -dimensional Poincaré Ball model is defined as:

$$d(\mathbf{u}, \mathbf{v}) = \operatorname{arcosh}\left(1 + 2\frac{\|\mathbf{u} - \mathbf{v}\|^2}{(1 - \|\mathbf{u}\|^2)(1 - \|\mathbf{v}\|^2)}\right). \quad (8.2)$$

This Poincaré distance enables us to capture the hierarchy and similarity between words simultaneously. It increases exponentially with the depth of the hierarchy. So while the

distance of a leaf node to most other nodes in the hierarchy is very high, nodes on abstract levels, such as the root, have a comparably small distance to all nodes in the hierarchy. The word2vec embeddings have no notion of hierarchy and hierarchical relationships cannot be represented with vector offsets across the vocabulary [103]. When applying word2vec, we use the observation that distributionally similar words are often co-hyponyms [104,105].

### **Relocation of Outlier Terms**

Poincaré embeddings are used to compute and store a rank  $rank(x, y)$  between every child and parent of the existing taxonomy, defined as the index of  $y$  in the list of sorted Poincaré distances of all entities of the taxonomy to  $x$ . Hypernym-hyponym relationships with a rank larger than the mean of all ranks are removed, chosen on the basis of tests on the 2015 TExEval data [106]. Disconnected components that have children are re-connected to the most similar parent in the taxonomy or to the taxonomy root. Previously or now disconnected isolated nodes are subject to orphan attachment.

Since distributional similarity does not capture parent-child relations, the relationships are not registered as parent-child but as co-hyponym relationships. Thus, we compute the distance to the closest co-hyponym (child of the same parent) for every node. This filtering technique is then applied to identify and relocate outliers.

### **Attachment of Orphan Terms**

We then attach orphans (nodes unattached in the input or due to the removal of relationships in the previous step) by computing the rank between every orphan and the most similar node in the taxonomy. This node is an orphan’s potential parent. Only hypernym-hyponym relationships with a rank lower or equal to the mean of all stored ranks are added to the taxonomy. For the word2vec system, a link is added between the parent of the most similar co-hyponym and the orphan.

### **Attachment of Compound Terms**

In case a representation for a compound noun term does not exist, we connect it to a term that is a substring of the compound. If no such term exists, the noun remains disconnected. Finally, the Tarjan algorithm [107] is applied to ensure that the refined taxonomy is asymmetric: In case a circle is detected, one of its links is removed at random.



Word	Parent patterns	Parent after refinement	Gold parent	Closest neighbors
second language acquisition	—	linguistics	linguistics	applied linguistics, semantics, linguistics
botany	—	genetics	plant science, ecology	genetics, evolutionary ecology, animal science
sweet potatoes	—	vegetables	vegetables	vegetables, side dishes, fruit
wastewater	water	waste	waste	marine pollution, waste, pollutant
water	waste, natural resources	natural resources	aquatic environment	continental shelf, management of resources
international relations	sociology, analysis, humanities	humanities	political science	economics, economic theory, geography

Table 8.1: Example words with respective parent(s) in the input taxonomy constructed using Hearst’ patterns approach and after refinement using our domain-specific Poincaré embeddings, as well as the word’s closest three neighbors (incl. orphans) in embeddings.

## 8.3 Results

Evaluation of the proposed method was performed on SemEval-2016 taxonomy enrichment dataset and output of three top systems. The refinement method is generically applicable to noisy taxonomies, yielding an improved taxonomy extraction system overall. Examples of predictions are available in Table 8.1. Experiments show that, the developed refinement method for improving existing taxonomies through the use of hyperbolic Poincaré embeddings consistently yield improvements over strong baselines and in comparison to word2vec as a representative for distributional vectors in the Euclidean space. It was further shown that Poincaré embeddings can be efficiently created for a specific domain from crawled text without the need for an existing database such as WordNet. This observation confirms the theoretical capability of Poincaré embeddings to learn hierarchical relations, which enables their future use in a wide range of semantic tasks.

Further experimental results and their analysis can be found in [5].

# Chapter 9

## Node Embeddings of Lexical-Semantic Graphs

Materials of this chapter are based on paper [3] from the list of 14 publications the thesis is based on.

### 9.1 Introduction

In this chapter methods for learning embedding of graph-based metrics are introduced.

The task of **graph metric learning** is as following. Given a graph  $G = (E, V)$  and a **graph metric**  $sim : E \times E \rightarrow [0; 1]$  to learn a matrix of **node embeddings**  $\mathbf{E} \in R^{|E| \times d}$ , where  $d$  is the embedding dimensionality, such that  $sim(e_i, e_j) \approx f(\mathbf{e}_i, \mathbf{e}_j)$ , where  $f$  is some vector-based distance, faster to compute than  $sim$ .

When operating on large graphs, such as transportation networks, social networks, or lexical resources, the need for estimating similarities between nodes arises. For many domain-specific applications, custom graph node similarity measures  $sim : V \times V \rightarrow \mathbb{R}$  have been defined on pairs of nodes  $V$  of a graph  $G = (V, E)$ . Examples include travel time, communities, or semantic distances for knowledge-based word sense disambiguation on WordNet [108]. For instance, the similarity  $s_{ij}$  between the `cup.n.01` and `mug.n.01` synsets in the WordNet is  $\frac{1}{4}$  according to the inverted shortest path distance as these two nodes are connected by the undirected path `cup`  $\rightarrow$  `container`  $\leftarrow$  `vessel`  $\leftarrow$  `drinking_vessel`  $\leftarrow$  `mug`.

A large variety of such node similarity measures have been described, many of which

are based on the notion of a random walk [109–111]. As given by the structure of the problem, most such measures are defined as traversals of edges  $E$  of the graph, which makes their computation prohibitively inefficient. To this end, we propose the *path2vec* model, which solves this problem by decoupling development and use of graph-based measures, and – in contrast to purely walk- based embeddings – is trainable to reflect custom node similarity measures. We represent nodes in a graph with dense embeddings that are good in approximating such custom, e.g. application-specific, pairwise node similarity measures. Similarity computations in a vector space are several orders of magnitude faster than computations directly operating on the graph.

## 9.2 Method

### Definition of the Model

*Path2vec* is a graph metric embeddings model learns embeddings of the graph nodes  $\{v_i, v_j\} \in V$  such that the dot products between pairs of the respective vectors ( $\mathbf{v}_i \cdot \mathbf{v}_j$ ) are close to the user-defined similarities between the nodes  $s_{ij}$ . In addition, the model reinforces the similarities  $\mathbf{v}_i \cdot \mathbf{v}_n$  and  $\mathbf{v}_j \cdot \mathbf{v}_m$  between the nodes  $v_i$  and  $v_j$  and all their respective adjacent nodes  $\{v_n : \exists(v_i, v_n) \in E\}$  and  $\{v_m : \exists(v_j, v_m) \in E\}$  to preserve local structure of the graph. The model preserves both **global** and **local** relations between nodes by minimizing

$$\mathcal{L} = \sum_{(v_i, v_j) \in B} ((\mathbf{v}_i^\top \mathbf{v}_j - s_{ij})^2 - \alpha(\mathbf{v}_i^\top \mathbf{v}_n + \mathbf{v}_j^\top \mathbf{v}_m)), \quad (9.1)$$

where  $s_{ij} = \text{sim}(v_i, v_j)$  is the value of a ‘gold’ similarity measure between a pair of nodes  $v_i$  and  $v_j$ ,  $\mathbf{v}_i$  and  $\mathbf{v}_j$  are the embeddings of the first and the second node,  $B$  is a training batch,  $\alpha$  is a regularization coefficient. The second term ( $\mathbf{v}_i \cdot \mathbf{v}_n + \mathbf{v}_j \cdot \mathbf{v}_m$ ) in the objective function is a regularizer that aids the model to simultaneously maximize the similarity between adjacent nodes while learning the similarity between the two target nodes (one adjacent node is randomly sampled for each target node).

We use negative sampling to form a training batch  $B$  adding  $p$  negative samples ( $s_{ij} = 0$ ) for each real ( $s_{ij} > 0$ ) training instance: each real node (synset) pair  $(v_i, v_j)$  with ‘gold’ similarity  $s_{ij}$  is accompanied with  $p$  ‘negative’ node pairs  $(v_i, v_k)$  and  $(v_j, v_l)$  with zero similarities, where  $v_k$  and  $v_l$  are randomly sampled nodes from  $V$ . Embeddings are

initialized randomly and trained using the *Adam* optimizer [112] with early stopping. Once the model is trained, the computation of node similarities is approximated with the dot product of the learned node vectors, making the computations efficient:  $\hat{s}_{ij} = \mathbf{v}_i \cdot \mathbf{v}_j$ .

### Relation to Similar Models

Our model bears resemblance to the Skip-gram model [113], where the vector dot product  $\mathbf{v}_i \cdot \tilde{\mathbf{v}}_j$  of vectors of pairs of words  $(v_i, v_j)$  from a training corpus is optimized to a high score close to 1 for observed samples, while the dot products of negative samples are optimized towards 0. In the Skip-gram model, the target is to minimize the log likelihood of the conditional probabilities of context words  $w_j$  given current words  $w_i$ :

$$\mathcal{L} = - \sum_{(v_i, v_j) \in B_p} \log \sigma(\mathbf{v}_i \cdot \tilde{\mathbf{v}}_j) - \sum_{(v_i, v_j) \in B_n} \log \sigma(-\mathbf{v}_i \cdot \tilde{\mathbf{v}}_j), \quad (9.2)$$

where  $B_p$  is the batch of positive training samples,  $B_n$  is the batch of the generated negative samples, and  $\sigma$  is the sigmoid function. At this, Skip-gram uses only **local** information, never creating the full co-occurrence count matrix. In our *path2vec* model, the target dot product values  $s_{ij}$  are not binary, but can take arbitrary values in the  $[0...1]$  range, as given by the custom distance metric. Further, we use only a single embedding matrix with vector representations of the graph nodes, not needing to distinguish target and context.

Another related model is Global Vectors (GloVe) [114], which learns co-occurrence probabilities in a given corpus. The objective function to be minimized in GloVe model is  $\mathcal{L} = \sum_{(v_i, v_j) \in B} f(s_{ij})(\mathbf{v}_i \cdot \tilde{\mathbf{v}}_j - \log s_{ij} + b_i + b_j)^2$ , where  $s_{ij}$  counts the co-occurrences of words  $v_i$  and  $v_j$ ,  $b_i$  and  $b_j$  are additional biases for each word, and  $f(s_{ij})$  is a weighting function handling rare co-occurrences. Like the Skip-gram, GloVe also uses two embedding matrices, but it relies only on **global** information, pre-aggregating global word co-occurrence counts.

### Computing Training Similarities

In general case, our model requires computing pairwise node similarities  $s_{ij}$  for training between all pairs of nodes in the input graph  $G$ . This step could be computationally expensive, but it is done only once to make computing of similarities fast. Besides, for some metrics, effective algorithms exist that compute all pairwise similarities at once, e.g. [115] algorithm for computing shortest paths distances with the worst-case performance of  $O(|V|^2 \log |V| + |V||E|)$ . As the input training dataset also grows quadratically in  $|V|$ ,

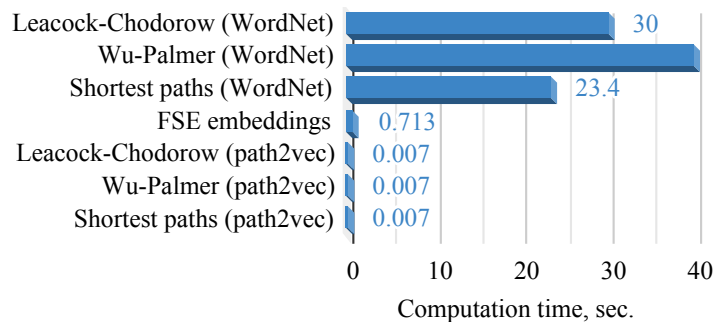


Figure 9-1: Similarity computation: graph vs vectors.

training time for large graphs can be slow. To address this issue, we found it useful to prune the input training set so that each node  $v_i \in V$  has only  $k \in [50; 200]$  most similar nodes. Such pruning does not lead to loss of effectiveness.

## 9.3 Results

Experiments were conducted to measure computational efficiency of the algorithm, as well as quality of approximation intrinsically (based on SimLex999 dataset) and extrinsically based on three SemEval WSD datasets. A comparison to baseline approaches, such as Deepwalk, node2vec or TransR were conducted. It was demonstrated that the approach generalizes well across graphs (WordNet, Freebase, and DBpedia). Besides, the method was integrated into a graph-based WSD algorithm, showing that its vectorized counterpart yields comparable F1 scores for the WSD task.

*Path2vec* enables a speed-up of up to four orders of magnitude for the computation of graph distances as compared to ‘direct’ graph measures (see Figure 9-1). Thus, the model is simple and general, hence it may be applied to any graph together with a node distance measure to speed up algorithms that employ graph distances.

Structured knowledge contained in language networks is useful for NLP applications but is difficult to use directly in neural architectures. In this chapter, a way to train embeddings that directly represent a graph-based similarity measure structure was proposed. The model, *path2vec*, relies on both global and local information from the graph and is simple, effective, and computationally efficient.

Further experimental results and their analysis can be found in [3].

# Chapter 10

## Lexical Substitution and Analysis of its Semantic Relation Types

Materials of this chapter are based on paper [2] from the list of 14 publications the thesis is based on.

### 10.1 Introduction

In this chapter methods for lexical substitution and its analysis are presented.

Lexical substitution is the task of generating words that can replace a given word in a given textual context. For instance, in the sentence “*My daughter purchased a new car*” the word *car* can be substituted by its synonym *automobile*, but also with co-hyponym *bike*, or even hypernym *motor vehicle* while keeping the original sentence grammatical.

More formally, the task of **lexical substitution** is formulated as following. Given a **sentence**  $S$  composed of a **context**  $C$  and a **target word**  $T$  find **lexical substitutes**: words/phrases which can be used to replace  $T$  without changing meaning of  $S$  as shown below:

- “We were not able to travel in the weather, and there was no phone.” → telephone;
- “What happened to the big, new garbage can at Church and Chambers Streets?” → bin, disposal, container.

Generation of plausible words that can replace a particular target word in a given context

is a powerful technology that can be used as a backbone of various NLP applications, such as word sense induction [116], lexical relation extraction [117], paraphrase generation, text simplification, textual data augmentation, etc. Note that the preferable type (e.g., synonym, hypernym, co-hyponym, etc.) of generated substitutes depends on the task at hand.

In this section, presented a study of lexical substitution methods employing both classic and more recent language and masked language models (LMs and MLMs), such as context2vec, ELMo, BERT, RoBERTa, XLNet. It is shown that already competitive results achieved by SOTA LMs/MLMs can be further substantially improved if information about the target word is injected properly. Besides, an analysis of lexical semantic relation types generated by these models is performed, as illustrated in Figure 10-1.

We were not able to travel in the weather , and there was no <b>phone</b> .										
GOLD	<b>telephone</b> (5)									
OOC	phone	<b>telephone</b>	phones	cellphone	fone	videophone	handset	telephones	p990i	cell-phone
XLNet	electricity	internet	phone	power	<b>telephone</b>	car	water	communication	radio	tv
XLNet+embs	phone	<b>telephone</b>	phones	cellphone	internet	radio	electricity	iphone	car	computer
What happened to the big , new garbage <b>can</b> at Church and Chambers Streets ?										
GOLD	<b>bin</b> (4)	<b>disposal</b> (1)	<b>container</b> (1)							
OOC	can	could	should	would	will	must	might	to	may	ll
XLNet	can	dump	<b>bin</b>	truck	<b>disposal</b>	pit	heap	pile	<b>container</b>	stand
XLNet+embs	can	could	will	<b>bin</b>	cannot	dump	may	truck	<b>disposal</b>	stand

Types of semantic relations: ■ synonym ■ co-hyponym ■ co-hyponym 3 ■ target ■ direct hypernym ■ transitive hypernym  
■ direct hyponym ■ transitive hyponym ■ unknown-relation ■ unknown-word

Figure 10-1: Examples of top substitutes provided by annotators (GOLD), the baseline (OOC), and two presented models (XLNet and XLNet+embs). The target word in each sentence is in bold, true positives are in bold also. The weights of gold substitutes are given in brackets. Each substitute is colored according to its lexical-semantic relation to the target word.

## 10.2 Method

To generate substitutes, we introduce several substitute probability estimators, which are models taking a text fragment and a target word position in it as input and producing a list of substitutes with their probabilities. To build our substitute probability estimators we employ the following LMs/MLMs: context2vec [118], ELMo [119], BERT [120], RoBERTa [121] and XLNet [122]. These models were selected to represent the progress in unsupervised pre-training with language modeling and similar tasks.

Given a target word occurrence, the basic approach for models like context2vec and ELMo is to encode its context and predict the probability distribution over possible center words in this particular context. This way, the model does not see the target word. For MLMs, the same result can be achieved by masking the target word. This basic approach employs the core ability of LMs/MLMs of predicting words that fit a particular context. However, these words are often not related to the target. The information about the target word can improve generated substitutes, but what is the best method of injecting this information is an open question.

Proposed is a method to introduce information about the original target word into neural lexical substitution models. Suppose we have an example  $LTR$ , where  $T$  is the target word, and  $C = (L, R)$  is its context (left and right, correspondingly). For instance, the occurrence of the target word *fly* in the sentence “*Let me fly away!*” will be represented as  $T = \text{“fly”}$ ,  $L = \text{“Let me”}$ ,  $R = \text{“away!”}$ .

The proposed method combines a distribution provided by a context-based substitute probability estimator  $P(s|C)$  with a distribution based on the proximity of possible substitutes to the target  $P(s|T)$ . The proximity is computed as the inner product between the respective embeddings, and the softmax function is applied to get a probability distribution. However, if we simply multiply these distributions, the second will have almost no effect because the first is very peaky. To align the orders of distributions, we use temperature softmax with carefully selected temperature hyperparameter:

$$P(s|T) \propto \exp\left(\frac{\langle emb_s, emb_T \rangle}{\mathcal{T}}\right). \quad (10.1)$$

Target word injection methods rely on word embeddings similarity and is denoted as “+embs”. The final distribution is obtained by the formula

$$P(s|C, T) \propto \frac{P(s|C)P(s|T)}{P(s)^\beta}. \quad (10.2)$$

For  $\beta = 1$ , this formula can be derived by applying the Bayes rule and assuming conditional independence of  $C$  and  $T$  given  $s$ . Other values of  $\beta$  can be used to penalize frequent words, more or less. The current methods are limited to generating only substitutes from the vocabulary of the underlying LM/MLM. Thus, we take word or subword embeddings of the same model we apply the injection to.



Word probabilities  $P(s)$  are retrieved from word frequencies for all models except ELMo. Following [123], for ELMo, we calculate word probabilities from word ranks in the ELMo vocabulary (which is ordered by word frequencies) based on Zipf-Mandelbrot distribution.

In addition to this method simpler injection strategies were tested e.g. dynamic patterns, duplicate input or original input. These showed weaker results. Different LMs/MLMs are employed to obtain context-based substitute probability distribution  $P(s|C)$ . For each of them, we experiment with different target injection methods: context2vec, ELMo, BERT/RoBERTa, XLNet.

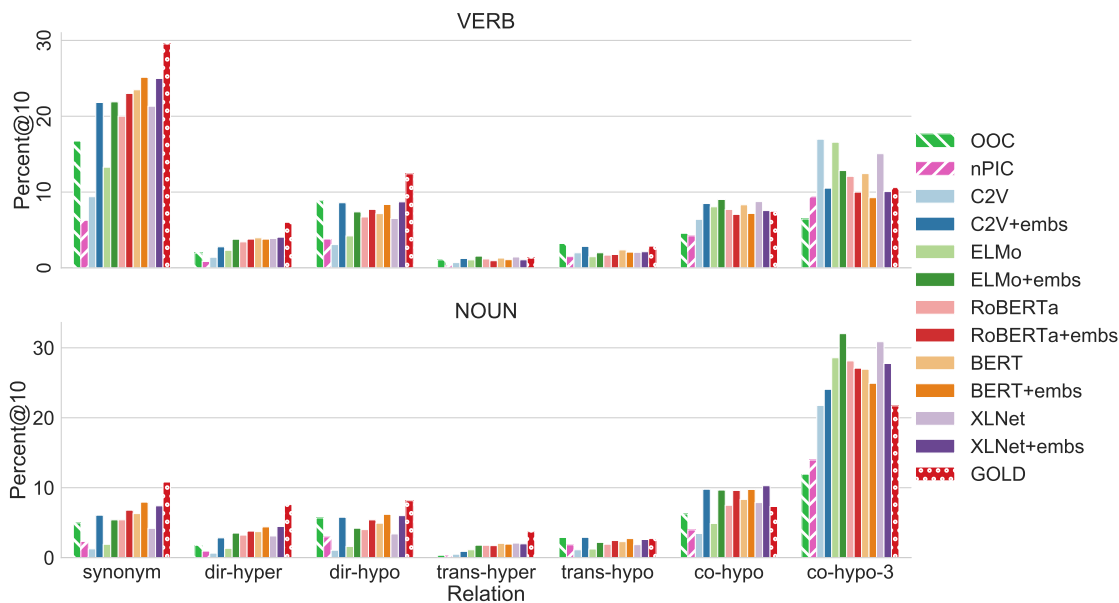


Figure 10-2: Proportions of substitutes related to the target by various semantic relations according to WordNet. We took top 10 substitutes from each model and all substitutes from the gold standard.

### 10.3 Results

Several existing and new target word injection methods are compared for each LM/MLM using both intrinsic evaluation on lexical substitution datasets (SemEval-2007 and CoInCo) and extrinsic evaluation on word sense induction (WSI) datasets (SemEval-2010 and SemEval-2013). Experimental results on two WSI datasets show that the proposed lexical substitution method with injection of information about target word (+embs)

consistently outperforms baseline methods, such as C2V, ELMo, BERT, RoBERTa, XLNet, without such information and ranks favourably to the previous SOTA results.

Besides, this work presented an analysis of the types of semantic relations, such as synonymy, hypernymy, and co-hyponymy, between target words and their substitutes generated by different models as illustrated in Figure 10-2. Results suggest that, co-hyponyms dominate among substitutes for nouns while synonyms dominate for verbs with co-hyponyms representing second largest type.

Further experimental results and their analysis can be found in [2].

# Chapter 11

## Conclusion

The results of this thesis are based on work presented in a series of research papers and journal articles at various international venues on methods and algorithms of computational lexical semantics with application to extraction of word senses, hypernymy relations between words, and semantic frames [1–42]. More specifically points defended in this dissertation are based on 14 publications [1–10, 17–20].

The set of results touched in the mentioned publications provide a comprehensive computational framework for processing of meaning of words and phrases and relations between them (the core topic of lexical semantics). Two types of computer-readable representations of lexical units (e.g., words and phrases) and semantic relations between them are currently established: (i) resources constructed manually, such as WordNet, BabelNet, or FrameNet and, (ii) automatically induced lexical semantic representation from raw text, such as sparse or dense word embeddings, and relations extracted with rules or statistical taggers. These two representations of lexical units and relations between them were traditionally separated. Some NLP methods would rely solely on precise yet limited in recall manually constructed lexical resources, while others would only on distributional methods which are more prone to inconsistencies of semantic representation yet provide higher recall as they are obtained from huge unlabelled text corpora. In this dissertation we made several methodological contributions related to both representations and their combination.

All in all, contributions presented in this thesis can be grouped as following:

- **Graph clustering algorithm for linguistic networks:** A fuzzy meta algorithm

for graph clustering for the needs of large linguistic graph processing. The algorithm was applied to extract synsets, semantic frames, and semantic classes.

- **Word sense embeddings:** Methods for learning sense embeddings using graph clustering and a mechanism for disambiguation of words with respect to these representations.
- **Inducing interpretable word sense representations:** Methods for inducing word sense representations and making them interpretable by automatic retrieval of hypernyms, images, and definitions.
- **Alignment of word sense representations:** Technology for alignment of manually created and automatically induced word senses. This set of methods allows enrichment of precise low coverage manual representations with high coverage word senses induced from text.
- **Disambiguation of word senses in context:** Methods for disambiguation of word meaning in context were proposed in form of classic WSD (word sense disambiguation) setup where given a word-context pair a sense identifier has to be predicted, and in the form of LexSub (lexical substitution setup) where given word-context pair a synonym fit to meaning of the context has to be predicted.
- **Induction of semantic trees:** Methods for dealing with special kind of lexical semantic resources composed of hierarchical relations, namely taxonomies, were proposed. Methods for population of existing, e.g. manually constructed, semantic trees with new nodes, e.g. new words and phrases not covered by vocabulary of such taxonomy. Besides, an approach based on clustering of graphs of automatically induced word senses was devised to perform cleansing of automatically extracted hypernymy relations.
- **Vectorisation of lexical semantic networks:** Methods for node embeddings based on graph-based measures. Application for completion of missing edges of lexical databases and knowledge graphs, and for word sense disambiguation.

Therefore, the methodological contributions cover both aspects of automatic construction of distributional and symbolic representations of word senses. Besides,

methods for linking these representation and disambiguation in context are presented. The methodology is heavily based on graph-based methods, notably graph clustering, showing that its a versatile tool in various task of computational lexical semantics. Most developed methods rely on graph representations as lexical semantic resources are naturally represented as graphs (with nodes being word senses and semantic relations being edges). At the same time, vector representations are also used demonstrating the usefulness of duality of graph-vector representation metaphor for various tasks of computational lexical semantics.

The “eco-system” of the developed methods and techniques illustrated in Figure 1-2, from one hand can be used to automate work manual labour of lexicographers creating and keeping up-to date various lexical-semantic resources. On the other hand, the developed methods can be used for improve readability of the neural models though linking vector-based representations inherently used by these models to interpretable graph-based representations. Besides, such linking of graph-based structures can improve performance in applications yielding additional representations.

A promising direction of the future work is to investigate more large language models (LLMs), such as T5, GPT, or LLaMa for the task of generation and completion of lexical semantic resources and other tasks related to modelling meaning of words senses and relations between them. Some results of work in this direction were already published by the author in [124–126], which suggest that using LLMs to solve tasks discussed in this thesis is a promising direction for future research and developments.

# Bibliography

- [1] [A. Panchenko](#), E. Ruppert, S. Faralli, S. P. Ponzetto, and C. Biemann, “**Unsupervised does not mean uninterpretable: The case for word sense induction and disambiguation**,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, (Valencia, Spain), pp. 86–98, Association for Computational Linguistics, Apr. 2017.
- [2] N. Arefyev, B. Sheludko, A. Podolskiy, and [A. Panchenko](#), “**Always Keep your Target in Mind: Studying Semantics and Improving Performance of Neural Lexical Substitution**,” in *Proceedings of the 28th International Conference on Computational Linguistics*, (Barcelona, Spain (Online)), pp. 1242–1255, International Committee on Computational Linguistics, Dec. 2020.
- [3] A. Kutuzov, M. Dorgham, O. Oliynyk, C. Biemann, and [A. Panchenko](#), “**Making Fast Graph-based Algorithms with Graph Metric Embeddings**,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 3349–3355, Association for Computational Linguistics, July 2019.
- [4] D. Ustalov, N. Arefyev, C. Biemann, and [A. Panchenko](#), “**Negative Sampling Improves Hypernymy Extraction Based on Projection Learning**,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, (Valencia, Spain), pp. 543–550, Association for Computational Linguistics, Apr. 2017.
- [5] R. Aly, S. Acharya, A. Ossa, A. Köhn, C. Biemann, and [A. Panchenko](#), “**Every Child Should Have Parents: A Taxonomy Refinement Algorithm Based on Hyperbolic Term Embeddings**,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 4811–4817, Association for Computational Linguistics, July 2019.
- [6] D. Ustalov, [A. Panchenko](#), C. Biemann, and S. P. Ponzetto, “**Watset: Local-Global Graph Clustering with Applications in Sense and Frame Induction**,” *Computational Linguistics*, vol. 45, pp. 423–479, Sept. 2019.
- [7] S. Faralli, [A. Panchenko](#), C. Biemann, and S. P. Ponzetto, “**Linked Disambiguated Distributional Semantic Networks**,” in *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II* (P. Groth, E. Simperl, A. J. G. Gray, M. Sabou, M. Krötzsch,

- F. Lécué, F. Flöck, and Y. Gil, eds.), vol. 9982 of *Lecture Notes in Computer Science*, pp. 56–64, 2016.
- [8] C. Biemann, S. Faralli, [A. Panchenko](#), and S. P. Ponzetto, “**A framework for enriching lexical semantic resources with distributional semantics**,” *Nat. Lang. Eng.*, vol. 24, no. 2, pp. 265–312, 2018.
- [9] D. Ustalov, [A. Panchenko](#), and C. Biemann, “**Watset: Automatic Induction of Synsets from a Graph of Synonyms**,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Vancouver, Canada), pp. 1579–1590, Association for Computational Linguistics, July 2017.
- [10] D. Ustalov, [A. Panchenko](#), A. Kutuzov, C. Biemann, and S. P. Ponzetto, “**Unsupervised Semantic Frame Induction using Triclustering**,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, (Melbourne, Australia), pp. 55–62, Association for Computational Linguistics, July 2018.
- [11] Ö. Sevgili, A. Shelmanov, M. Y. Arkhipov, [A. Panchenko](#), and C. Biemann, “**Neural entity linking: A survey of models based on deep learning**,” *Semantic Web*, vol. 13, no. 3, pp. 527–570, 2022.
- [12] S. Anwar, A. Shelmanov, N. Arefyev, [A. Panchenko](#), and C. Biemann, “**Text augmentation for semantic frame induction and parsing**,” *Language Resources and Evaluation*, vol. 23, no. 3, pp. 527–556, 2023.
- [13] A. Jana, D. Puzyrev, [A. Panchenko](#), P. Goyal, C. Biemann, and A. Mukherjee, “**On the Compositionality Prediction of Noun Phrases using Poincaré Embeddings**,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 3263–3274, Association for Computational Linguistics, July 2019.
- [14] I. Nikishina, V. Logacheva, [A. Panchenko](#), and N. Loukachevitch, “**Studying Taxonomy Enrichment on Diachronic WordNet Versions**,” in *Proceedings of the 28th International Conference on Computational Linguistics*, (Barcelona, Spain (Online)), pp. 3095–3106, International Committee on Computational Linguistics, Dec. 2020.
- [15] I. Nikishina, M. Tikhomirov, V. Logacheva, Y. Nazarov, [A. Panchenko](#), and N. V. Loukachevitch, “**Taxonomy enrichment with text and graph vector representations**,” *Semantic Web*, vol. 13, no. 3, pp. 441–475, 2022.
- [16] S. Faralli, [A. Panchenko](#), C. Biemann, and S. P. Ponzetto, “**The ContrastMedium Algorithm: Taxonomy Induction From Noisy Knowledge Graphs With Just A Few Links**,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, (Valencia, Spain), pp. 590–600, Association for Computational Linguistics, Apr. 2017.
- [17] M. Pelevina, N. Arefiev, C. Biemann, and [A. Panchenko](#), “**Making Sense of Word Embeddings**,” in *Proceedings of the 1st Workshop on Representation Learning for*

- NLP*, (Berlin, Germany), pp. 174–183, Association for Computational Linguistics, Aug. 2016.
- [18] [A. Panchenko](#), F. Marten, E. Ruppert, S. Faralli, D. Ustalov, S. P. Ponzetto, and C. Biemann, “**Unsupervised, Knowledge-Free, and Interpretable Word Sense Disambiguation**,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Copenhagen, Denmark), pp. 91–96, Association for Computational Linguistics, Sept. 2017.
- [19] [A. Panchenko](#), D. Ustalov, S. Faralli, S. P. Ponzetto, and C. Biemann, “**Improving Hypernymy Extraction with Distributional Semantic Classes**,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, (Miyazaki, Japan), European Language Resources Association (ELRA), May 2018.
- [20] V. Logacheva, D. Teslenko, A. Shelmanov, S. Remus, D. Ustalov, A. Kutuzov, E. Artemova, C. Biemann, S. P. Ponzetto, and [A. Panchenko](#), “**Word Sense Disambiguation for 158 Languages using Word Embeddings Only**,” in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, (Marseille, France), pp. 5943–5952, European Language Resources Association, May 2020.
- [21] [A. Panchenko](#), S. Faralli, E. Ruppert, S. Remus, H. Naets, C. Fairon, S. P. Ponzetto, and C. Biemann, “**TAXI at SemEval-2016 Task 13: a Taxonomy Induction Method based on Lexico-Syntactic Patterns, Substrings and Focused Crawling**,” in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, (San Diego, California), pp. 1320–1327, Association for Computational Linguistics, June 2016.
- [22] S. Anwar, A. Shelmanov, [A. Panchenko](#), and C. Biemann, “**Generating Lexical Representations of Frames using Lexical Substitution**,” in *Proceedings of the Probability and Meaning Conference (PaM 2020)*, (Gothenburg), pp. 95–103, Association for Computational Linguistics, June 2020.
- [23] D. Ustalov, [A. Panchenko](#), C. Biemann, and S. P. Ponzetto, “**Unsupervised Sense-Aware Hypernymy Extraction**,” in *Proceedings of the 14th Conference on Natural Language Processing, KONVENS 2018, Vienna, Austria, September 19-21, 2018* (A. Barbaresi, H. Biber, F. Neubarth, and R. Osswald, eds.), pp. 192–201, Österreichische Akademie der Wissenschaften, 2018.
- [24] [A. Panchenko](#), A. Lopukhina, D. Ustalov, K. Lopukhin, N. Arefyev, A. Leontyev, and N. V. Loukachevitch, “**RUSSE’2018: A Shared Task on Word Sense Induction for the Russian Language**,” in *Proceedings of the 24th International Conference on Computational Linguistics and Intellectual Technologies (Dialogue’2018)*. Moscow, Russia., pp. 192–201, RGGU, 2018.
- [25] N. Arefyev, P. Ermolaev, and [A. Panchenko](#), “**How much does a word weigh? Weighting word embeddings for word sense induction**,” in *Proceedings of the 24th International Conference on Computational Linguistics and Intellectual Technologies (Dialogue’2018)*. Moscow, Russia., pp. 201–212, RGGU, 2018.



- [26] D. Ustalov, D. Teslenko, [A. Panchenko](#), M. Chernoskutov, C. Biemann, and S. P. Ponzetto, “**An Unsupervised Word Sense Disambiguation System for Under-Resourced Languages**,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, (Miyazaki, Japan), European Language Resources Association (ELRA), May 2018.
- [27] S. Faralli, [A. Panchenko](#), C. Biemann, and S. P. Ponzetto, “**Enriching Frame Representations with Distributionally Induced Senses**,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, (Miyazaki, Japan), European Language Resources Association (ELRA), May 2018.
- [28] Ö. Sevgili, [A. Panchenko](#), and C. Biemann, “**Improving Neural Entity Disambiguation with Graph Embeddings**,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, (Florence, Italy), pp. 315–322, Association for Computational Linguistics, July 2019.
- [29] [A. Panchenko](#), “**Best of Both Worlds: Making Word Sense Embeddings Interpretable**,” in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, (Portorož, Slovenia), pp. 2649–2655, European Language Resources Association (ELRA), May 2016.
- [30] D. Ustalov, M. Chernoskutov, C. Biemann, and [A. Panchenko](#), “**Fighting with the Sparsity of Synonymy Dictionaries for Automatic Synset Induction**,” in *Analysis of Images, Social Networks and Texts - 6th International Conference, AIST 2017, Moscow, Russia, July 27-29, 2017, Revised Selected Papers* (W. M. P. van der Aalst, D. I. Ignatov, M. Y. Khachay, S. O. Kuznetsov, V. S. Lempitsky, I. A. Lomazova, N. V. Loukachevitch, A. Napoli, [A. Panchenko](#), P. M. Pardalos, A. V. Savchenko, and S. Wasserman, eds.), vol. 10716 of *Lecture Notes in Computer Science*, pp. 94–105, Springer, 2017.
- [31] [A. Panchenko](#), J. Simon, M. Riedl, and C. Biemann, “**Noun Sense Induction and Disambiguation using Graph-Based Distributional Semantics**,” in *Proceedings of the 13th Conference on Natural Language Processing, KONVENS 2016, Bochum, Germany, September 19-21, 2016* (S. Dipper, F. Neubarth, and H. Zinsmeister, eds.), vol. 16 of *Bochumer Linguistische Arbeitsberichte*, 2016.
- [32] D. Puzyrev, A. Shelmanov, [A. Panchenko](#), and E. Artemova, “**Noun Compositionality Detection Using Distributional Semantics for the Russian Language**,” in *Analysis of Images, Social Networks and Texts - 8th International Conference, AIST 2019, Kazan, Russia, July 17-19, 2019, Revised Selected Papers* (W. M. P. van der Aalst, V. Batagelj, D. I. Ignatov, M. Y. Khachay, V. V. Kuskova, A. Kutuzov, S. O. Kuznetsov, I. A. Lomazova, N. V. Loukachevitch, A. Napoli, P. M. Pardalos, M. Pelillo, A. V. Savchenko, and E. Tutubalina, eds.), vol. 11832 of *Lecture Notes in Computer Science*, pp. 218–229, Springer, 2019.
- [33] N. Arefyev, B. Sheludko, and [A. Panchenko](#), “**Combining Lexical Substitutes in Neural Word Sense Induction**,” in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, (Varna, Bulgaria), pp. 62–70, INCOMA Ltd., Sept. 2019.

- [34] A. Kutuzov, M. Dorgham, O. Oliynyk, C. Biemann, and A. Panchenko, “**Learning Graph Embeddings from WordNet-based Similarity Measures**,” in *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (\*SEM 2019)*, (Minneapolis, Minnesota), pp. 125–135, Association for Computational Linguistics, June 2019.
- [35] A. Razzhigaev, N. Arefyev, and A. Panchenko, “**SkoltechNLP at SemEval-2021 Task 2: Generating Cross-Lingual Training Data for the Word-in-Context Task**,” in *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, (Online), pp. 157–162, Association for Computational Linguistics, Aug. 2021.
- [36] D. Puzyrev, A. Shelmanov, A. Panchenko, and E. Artemova, “**A Dataset for Noun Compositionality Detection for a Slavic Language**,” in *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, (Florence, Italy), pp. 56–62, Association for Computational Linguistics, Aug. 2019.
- [37] N. Arefyev, B. Sheludko, A. Davletov, D. Kharchev, A. Nevidomsky, and A. Panchenko, “**Neural GRANNy at SemEval-2019 Task 2: A combined approach for better modeling of semantic relationships in semantic frame induction**,” in *Proceedings of the 13th International Workshop on Semantic Evaluation*, (Minneapolis, Minnesota, USA), pp. 31–38, Association for Computational Linguistics, June 2019.
- [38] S. Anwar, D. Ustalov, N. Arefyev, S. P. Ponzetto, C. Biemann, and A. Panchenko, “**HHMM at SemEval-2019 Task 2: Unsupervised Frame Induction using Contextualized Word Embeddings**,” in *Proceedings of the 13th International Workshop on Semantic Evaluation*, (Minneapolis, Minnesota, USA), pp. 125–129, Association for Computational Linguistics, June 2019.
- [39] A. Panchenko, S. Faralli, S. P. Ponzetto, and C. Biemann, “**Using Linked Disambiguated Distributional Networks for Word Sense Disambiguation**,” in *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, (Valencia, Spain), pp. 72–78, Association for Computational Linguistics, Apr. 2017.
- [40] I. Nikishina, I. Andrianov, A. Vakhitova, and A. Panchenko, “TaxFree: a visualization tool for candidate-free taxonomy enrichment,” in *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations* (W. Buntine and M. Liakata, eds.), (Taipei, Taiwan), pp. 39–47, Association for Computational Linguistics, Nov. 2022.
- [41] I. Nikishina, N. Loukachevitch, V. Logacheva, and A. Panchenko, “**Evaluation of Taxonomy Enrichment on Diachronic WordNet Versions**,” in *Proceedings of the 11th Global Wordnet Conference*, (University of South Africa (UNISA)), pp. 126–136, Global Wordnet Association, Jan. 2021.
- [42] I. Nikishina, A. Vakhitova, E. Tutubalina, and A. Panchenko, “**Cross-Modal Contextualized Hidden State Projection Method for Expanding of**

- Taxonomic Graphs,”** in *Proceedings of TextGraphs-16: Graph-based Methods for Natural Language Processing*, (Gyeongju, Republic of Korea), pp. 11–24, Association for Computational Linguistics, Oct. 2022.
- [43] B. Dorow and D. Widdows, “Discovering Corpus-Specific Word Senses,” in *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 2*, EACL ’03, (Budapest, Hungary), pp. 79–82, Association for Computational Linguistics, 2003.
- [44] G. Salton, A. Wong, and C. S. Yang, “A Vector Space Model for Automatic Indexing,” *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.
- [45] J. Véronis, “HyperLex: lexical cartography for information retrieval,” *Computer Speech & Language*, vol. 18, no. 3, pp. 223–252, 2004.
- [46] C. Biemann, “Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems,” in *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, (New York, NY, USA), pp. 73–80, Association for Computational Linguistics, 2006.
- [47] D. Hope and B. Keller, “MaxMax: A Graph-Based Soft Clustering Algorithm Applied to Word Sense Induction,” in *Computational Linguistics and Intelligent Text Processing: 14th International Conference, CICLing 2013, Samos, Greece, March 24-30, 2013, Proceedings, Part I*, vol. 7816 of *Lecture Notes in Computer Science*, pp. 368–381, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.
- [48] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *Workshop at International Conference on Learning Representations (ICLR)*, (Scottsdale, AZ, USA), pp. 1310–1318, 2013.
- [49] A. Joulin, E. Grave, P. Bojanowski, M. Nickel, and T. Mikolov, “Fast linear model for knowledge graph embeddings,” *arXiv preprint arXiv:1710.10881*, 2017.
- [50] C. Biemann and M. Riedl, “Text: Now in 2D! a framework for lexical expansion with contextual similarity,” *Journal of Language Modelling*, vol. 1, no. 1, pp. 55–95, 2013.
- [51] A. Panchenko, *Similarity measures for semantic relation extraction*. PhD thesis, Universit’e catholique de Louvain, Louvain-la-Neuve, Belgium, Louvain-la-Neuve, Belgium, 2013.
- [52] E. Ruppert, J. Klesy, M. Riedl, and C. Biemann, “Rule-based Dependency Parse Collapsing and Propagation for German and English,” in *Proceedings of the GSCL 2015*, (Duisburg, Germany), pp. 58–66, 2015.
- [53] K. W. Church and P. Hanks, “Word association norms, mutual information, and lexicography,” *Computational linguistics*, vol. 16, no. 1, pp. 22–29, 1990.
- [54] C. Biemann, “Chinese Whispers: an efficient graph clustering algorithm and its application to natural language processing problems,” in *Proceedings of the first workshop on graph based methods for natural language processing*, TextGraphs-1, (New York City, NY, USA), pp. 73–80, Association for Computational Linguistics, 2006.

- [55] C. Biemann, “Co-occurrence Cluster Features for Lexical Substitutions in Context,” in *Proceedings of the 5th Workshop on TextGraphs in conjunction with ACL 2010*, (Uppsala, Sweden), 2010.
- [56] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” in *Advances in Neural Information Processing Systems 26*, NIPS 2013, pp. 3111–3119, Harrahs and Harveys, NV, USA: Curran Associates, Inc., 2013.
- [57] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, “Learning Word Vectors for 157 Languages,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, (Miyazaki, Japan), pp. 3483–3487, European Language Resources Association (ELRA), 2018.
- [58] N. Ide and J. Véronis, “Introduction to the special issue on word sense disambiguation: the state of the art,” *Computational linguistics*, vol. 24, no. 1, pp. 2–40, 1998.
- [59] E. Agirre and P. G. Edmonds, *Word sense disambiguation: Algorithms and applications*, vol. 33. Springer Science & Business Media, 2007.
- [60] A. Moro and R. Navigli, “SemEval-2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking,” in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, (Denver, CO, USA), pp. 288–297, Association for Computational Linguistics, 2015.
- [61] R. Navigli, “Word sense disambiguation: A survey,” *ACM Computing Surveys (CSUR)*, vol. 41, no. 2, p. 10, 2009.
- [62] G. A. Miller, “WordNet: a lexical database for English,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [63] H. Schütze, “Automatic Word Sense Discrimination,” *Computational Linguistics*, vol. 24, no. 1, pp. 97–123, 1998.
- [64] J. Li and D. Jurafsky, “Do Multi-Sense Embeddings Improve Natural Language Understanding?,” in *Conference on Empirical Methods in Natural Language Processing (EMNLP’2015)*, EMNLP’15, (Lisboa, Portugal), pp. 1722–1732, Association for Computational Linguistics, 2015.
- [65] S. Bartunov, D. Kondrashkin, A. Osokin, and D. Vetrov, “Breaking Sticks and Ambiguities with Adaptive Skip-gram,” in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS’2016)*, AISTATS’16, (Cadiz, Spain), pp. 130–138, 2016.
- [66] A. Vellido, J. D. Martín, F. Rossi, and P. J. Lisboa, “Seeing is believing: The importance of visualization in real-world machine learning applications,” in *Proceedings of the 19th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN’2011)*, (Bruges, Belgium), pp. 219–226, 2011.
- [67] A. A. Freitas, “Comprehensible classification models: a position paper,” *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 1, pp. 1–10, 2014.

- [68] J. Li, X. Chen, E. Hovy, and D. Jurafsky, “Visualizing and Understanding Neural Models in NLP,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (San Diego, CA, USA), pp. 681–691, Association for Computational Linguistics, June 2016.
- [69] C. Biemann, “Turk Bootstrap Word Sense Inventory 2.0: A Large-Scale Resource for Lexical Substitution,” in *Proceedings of the 8th International Conference on Language Resources and Evaluation*, (Istanbul, Turkey), pp. 4038–4042, European Language Resources Association, 2012.
- [70] D. Jurgens and I. Klapaftis, “Semeval-2013 Task 13: Word Sense Induction for Graded and Non-graded Senses,” in *Proceedings of the 2nd Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval’2013)*, vol. 2, (Montreal, Canada), pp. 290–299, Association for Computational Linguistics, 2013.
- [71] S. P. Ponzetto and R. Navigli, “Knowledge-rich word sense disambiguation rivaling supervised systems,” in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL’10*, (Uppsala, Sweden), pp. 1522–1531, Association for Computational Linguistics, 2010.
- [72] I. Gurevych, J. ECKLE-KOHLER, S. Hartmann, M. Matuschek, C. M. Meyer, and C. Wirth, “UBY - A Large-Scale Unified Lexical-Semantic Resource Based on LMF,” in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, EACL’12*, (Avignon, France), pp. 580–590, Association for Computational Linguistics, 2012.
- [73] S. Pavel and J. Euzenat, “Ontology Matching: State of the Art and Future Challenges,” *IEEE Transaction on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 158–176, 2013.
- [74] R. Navigli and S. P. Ponzetto, “BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network,” *Artificial Intelligence*, vol. 193, pp. 217–250, 2012.
- [75] R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu, “Learning Semantic Hierarchies via Word Embeddings,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Baltimore, MD, USA), pp. 1199–1209, Association for Computational Linguistics, 2014.
- [76] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, (Berkeley, California, USA), pp. 281–297, University of California Press, 1967.
- [77] T. Wandmacher, “How semantic is Latent Semantic Analysis?,” in *Proceedings of RÉCITAL 2005*, (Dourdan, France), pp. 525–534, 2005.
- [78] K. Heylen, Y. Peirsman, D. Geeraerts, and D. Speelman, “Modelling Word Similarity: an Evaluation of Automatic Synonymy Extraction Algorithms,” in *Proceedings of the*

- Sixth International Conference on Language Resources and Evaluation (LREC'08)*, LREC 2008, (Marrakech, Morocco), pp. 3243–3249, European Language Resources Association (ELRA), 2008.
- [79] A. Panchenko, “Comparison of the Baseline Knowledge-, Corpus-, and Web-based Similarity Measures for Semantic Relations Extraction,” in *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, (Edinburgh, UK), pp. 11–21, Association for Computational Linguistics, 2011.
- [80] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *CoRR*, vol. abs/1412.6980, 2014.
- [81] M. Abadi et al., “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” *CoRR*, vol. abs/1603.04467, 2016.
- [82] S. Faralli, A. Panchenko, C. Biemann, and S. P. Ponzetto, “Linked Disambiguated Distributional Semantic Networks,” in *The Semantic Web – ISWC 2016: 15th International Semantic Web Conference, Proceedings, Part II*, Lecture Notes in Computer Science, pp. 56–64, Kobe, Japan: Springer International Publishing, 2016.
- [83] C. Biemann, S. Faralli, A. Panchenko, and S. P. Ponzetto, “A framework for enriching lexical semantic resources with distributional semantics,” *Natural Language Engineering*, pp. 1–48, 2018.
- [84] C. Biemann and M. Riedl, “Text: now in 2D! A framework for lexical expansion with contextual similarity,” *Journal of Language Modelling*, vol. 1, no. 1, pp. 55–95, 2013.
- [85] D. Widdows and B. Dorow, “A graph model for unsupervised lexical acquisition,” in *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pp. 1–7, Association for Computational Linguistics, 2002.
- [86] M. Everett and S. P. Borgatti, “Ego network betweenness,” *Social networks*, vol. 27, no. 1, pp. 31–38, 2005.
- [87] A. Ferraresi, E. Zanchetta, M. Baroni, and S. Bernardini, “Introducing and evaluating ukWaC, a very large web-derived corpus of English,” in *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, (Marrakech, Morocco), pp. 47–54, European Language Resources Association (ELRA), 2008.
- [88] M. Richter, U. Quasthoff, E. Hallsteinsdóttir, and C. Biemann, “Exploiting the Leipzig Corpora Collection,” in *Proceedings of IS-LTC'06*, (Ljubljana, Slovenia), European Language Resources Association (ELRA), 2006.
- [89] D. Graff and C. Cieri, “English Gigaword corpus,” *Linguistic Data Consortium*, 2003.
- [90] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum, “Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1059–1069, Association for Computational Linguistics, 2014.
- [91] M. Steyvers and J. B. Tenenbaum, “The Large-scale structure of semantic networks: Statistical analyses and a model of semantic growth,” *Cognitive science*, vol. 29, no. 1, pp. 41–78, 2005.

- [92] C. Fellbaum, *WordNet: An Electronic Database*. Cambridge, MA: MIT Press, 1998.
- [93] R. Navigli and S. P. Ponzetto, “BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network,” *Artificial Intelligence*, vol. 193, pp. 217–250, 2012.
- [94] M. Nickel and D. Kiela, “Poincaré Embeddings for Learning Hierarchical Representations,” in *Advances in Neural Information Processing Systems 30*, (Long Beach, CA, USA), pp. 6338–6347, 2017.
- [95] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda, “English gigaword fourth edition,” in *Linguistic Data Consortium*, (Philadelphia, PA, USA), 2009.
- [96] A. Ferraresi, E. Zanchetta, M. Baroni, and S. Bernardini, “Introducing and evaluating ukWaC, a very large web-derived corpus of English,” in *Proceedings of the 4th Web as Corpus Workshop. Can we beat Google?*, (Marrakech, Morocco), pp. 47–54, 2008.
- [97] D. Goldhahn, T. Eckart, and U. Quasthoff, “Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages,” in *Proceedings of the Eight International Conference on Language Resources and Evaluation*, (Istanbul, Turkey), pp. 759–765, 2012.
- [98] A. Panchenko, O. Morozova, and H. Naets, “A semantic similarity measure based on lexico-syntactic patterns,” in *Proceedings of KONVENS 2012*, (Vienna, Austria), pp. 174–178, September 2012.
- [99] J. Seitner, C. Bizer, K. Eckert, S. Faralli, R. M. und Heiko Paulheim, and S. P. Ponzetto, “A Large DataBase of Hypernymy Relations Extracted from the Web,” in *Proceedings of the 10th International Conference on Language Resources and Evaluation*, (Portorož, Slovenia), pp. 360–367, 2016.
- [100] A. Panchenko, S. Faralli, E. Ruppert, S. Remus, H. Naets, C. Fairon, S. P. Ponzetto, and C. Biemann, “TAXI at SemEval-2016 Task 13: a taxonomy Induction Method based on Lexico-syntactic Patterns, Substrings and Focused Crawling,” in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, SemEval@NAACL-HLT’16, (San Diego, CA, USA), pp. 1320–1327, Association for Computational Linguistics, 2016.
- [101] J. Stillwell, *Sources of hyperbolic geometry*. No. History of Mathematics, Volume 10 in 1, American Mathematical Society, 1996.
- [102] V. Khrukov, L. Mirvakhabova, E. Ustinova, I. Oseledets, and V. Lempitsky, “Hyperbolic Image Embeddings,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [103] R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu, “Learning semantic hierarchies via word embeddings,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, vol. 1, (Baltimore, MD, USA), pp. 1199–1209, Association for Computational Linguistics, 2014.
- [104] K. Heylen, Y. Peirsman, D. Geeraerts, and D. Speelman, “Modelling word similarity: an evaluation of automatic synonymy extraction algorithms,” in *Proceedings of*

- the sixth international language resources and evaluation*, (Marrakech, Morocco), pp. 3243–3249, 2008.
- [105] J. Weeds, D. Clarke, J. Reffin, D. Weir, and B. Keller, “Learning to distinguish hypernyms and co-hyponyms,” in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, (Dublin, Ireland), pp. 2249–2259, Dublin City University and Association for Computational Linguistics, 2014.
- [106] G. Bordea, P. Buitelaar, S. Faralli, and R. Navigli, “Semeval-2015 task 17: Taxonomy Extraction Evaluation (TExEval),” in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, (Denver, CO, USA), pp. 902–910, 2015.
- [107] R. Tarjan, “Depth first search and linear graph algorithms,” *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.
- [108] G. A. Miller, “WordNet: A Lexical Database for English,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [109] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, “Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation,” *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 3, pp. 355–369, 2007.
- [110] M. T. Pilehvar and R. Navigli, “From senses to texts: An all-in-one graph-based approach for measuring semantic similarity,” *Artificial Intelligence*, vol. 228, pp. 95–128, 2015.
- [111] B. Lebichot, G. Guex, I. Kivimäki, and M. Saerens, “A Constrained Randomized Shortest-Paths Framework for Optimal Exploration,” *arXiv preprint arXiv:1807.04551*, 2018.
- [112] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, (San Diego, CA, USA), 2015.
- [113] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” in *Advances in Neural Information Processing Systems 26*, (Lake Tahoe, NV, USA), pp. 3111–3119, Curran Associates, Inc., 2013.
- [114] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global Vectors for Word Representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, 2014.
- [115] D. B. Johnson, “Efficient algorithms for shortest paths in sparse networks,” *Journal of the ACM (JACM)*, vol. 24, no. 1, pp. 1–13, 1977.
- [116] A. Amrami and Y. Goldberg, “Word Sense Induction with Neural biLM and Symmetric Patterns,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 4860–4867, Association for Computational Linguistics, Oct.-Nov. 2018.



- [117] T. Schick and H. Schütze, “Rare Words: A Major Problem for Contextualized Embeddings and How to Fix it by Attentive Mimicking,” in *AAAI*, pp. 8766–8774, 2020.
- [118] O. Melamud, J. Goldberger, and I. Dagan, “context2vec: Learning Generic Context Embedding with Bidirectional LSTM,” in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, (Berlin, Germany), pp. 51–61, Association for Computational Linguistics, Aug. 2016.
- [119] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep Contextualized Word Representations,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 2227–2237, Association for Computational Linguistics, June 2018.
- [120] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.
- [121] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” *ArXiv*, vol. abs/1907.11692, 2019.
- [122] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” in *Advances in neural information processing systems*, pp. 5753–5763, 2019.
- [123] N. Arefyev, B. Sheludko, and A. Panchenko, “Combining Lexical Substitutes in Neural Word Sense Induction,” in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP’19)*, RANLP ’19, (Varna, Bulgaria), pp. 62–70, 2019.
- [124] V. Moskvoretskii, E. Neminova, A. Lobanova, A. Panchenko, and I. Nikishina, “TaxoLLaMA: WordNet-based model for solving multiple lexical semantic tasks,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (L.-W. Ku, A. Martins, and V. Srikumar, eds.), (Bangkok, Thailand), pp. 2331–2350, Association for Computational Linguistics, Aug. 2024.
- [125] V. Moskvoretskii, A. Panchenko, and I. Nikishina, “Are large language models good at lexical semantics? a case of taxonomy learning,” in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)* (N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, and N. Xue, eds.), (Torino, Italia), pp. 1498–1510, ELRA and ICCL, May 2024.
- [126] P. Chernomorchenko, A. Panchenko, and I. Nikishina, “Leveraging taxonomic information from large language models for hyponymy prediction,” in *Analysis*

*of Images, Social Networks and Texts* (D. I. Ignatov, M. Khachay, A. Kutuzov, H. Madoyan, I. Makarov, I. Nikishina, A. Panchenko, M. Panov, P. M. Pardalos, A. V. Savchenko, E. Tsymbalov, E. Tutubalina, and S. Zagoruyko, eds.), (Cham), pp. 49–63, Springer Nature Switzerland, 2024.