

# Methods and Algorithms of Computational Lexical Semantics with Applications to Extraction of Senses, Hypernyms, and Frames

Prof. Dr. **Alexander Panchenko**  
Skolkovo Institute of Science and Technology (Skoltech)

**Skoltech**

14.02.2024 ❤

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Download this Presentation



<https://alexanderpanchenko.github.io/presentations/ds.pdf>

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Object and goals of the dissertation

## Purpose

The purpose of the dissertation is the development of methods for **computational lexical semantics**

# Object and goals of the dissertation

## Purpose

The purpose of the dissertation is the development of methods for **computational lexical semantics**

- which would bridge the gap between the **precise well-interpretable** manually created lexical resources with **low coverage** and

# Object and goals of the dissertation

## Purpose

The purpose of the dissertation is the development of methods for **computational lexical semantics**

- which would bridge the gap between the **precise well-interpretable** manually created lexical resources with **low coverage** and
- **less precise and non-interpretable** automatically induced from text distributional lexical representations.

# Object and goals of the dissertation

## Goals

- 1 development of new algorithms for **processing large linguistic networks** constructed from both manually created lexical resources and graphs induced from text,

# Object and goals of the dissertation

## Goals

- 1 development of new algorithms for **processing large linguistic networks** constructed from both manually created lexical resources and graphs induced from text,
- 2 development of method for **induction** of lexical semantic structures from text, e.g. word senses,

# Object and goals of the dissertation

## Goals

- 1 development of new algorithms for **processing large linguistic networks** constructed from both manually created lexical resources and graphs induced from text,
- 2 development of method for **induction** of lexical semantic structures from text, e.g. word senses,
- 3 development of techniques for making the induced structures **interpretable** in the way they are in manually constructed resources,

# Object and goals of the dissertation

## Goals

- 1 development of new algorithms for **processing large linguistic networks** constructed from both manually created lexical resources and graphs induced from text,
- 2 development of method for **induction** of lexical semantic structures from text, e.g. word senses,
- 3 development of techniques for making the induced structures **interpretable** in the way they are in manually constructed resources,
- 4 development of methods for effective **disambiguation** in context with respect to the induced sense representations,

# Object and goals of the dissertation

## Goals

- 1 development of new algorithms for **processing large linguistic networks** constructed from both manually created lexical resources and graphs induced from text,
- 2 development of method for **induction** of lexical semantic structures from text, e.g. word senses,
- 3 development of techniques for making the induced structures **interpretable** in the way they are in manually constructed resources,
- 4 development of methods for effective **disambiguation** in context with respect to the induced sense representations,
- 5 development of effective **vectorization** of lexical semantic graphs for the use in various application.

# The obtained results

- 1 Algorithm for fuzzy graph clustering;

# The obtained results

- 1 Algorithm for fuzzy graph clustering;
- 2 Methods for induction of synsets, semantic classes and semantic frames;

# The obtained results

- 1 Algorithm for fuzzy graph clustering;
- 2 Methods for induction of synsets, semantic classes and semantic frames;
- 3 Methods for learning word sense embeddings;

# The obtained results

- 1 Algorithm for fuzzy graph clustering;
- 2 Methods for induction of synsets, semantic classes and semantic frames;
- 3 Methods for learning word sense embeddings;
- 4 Method for inducing interpretable word sense representations from text;

# The obtained results

- 1 Algorithm for fuzzy graph clustering;
- 2 Methods for induction of synsets, semantic classes and semantic frames;
- 3 Methods for learning word sense embeddings;
- 4 Method for inducing interpretable word sense representations from text;
- 5 Method for linking distributional word sense representations with lexical resources;

# The obtained results

- 1 Algorithm for fuzzy graph clustering;
- 2 Methods for induction of synsets, semantic classes and semantic frames;
- 3 Methods for learning word sense embeddings;
- 4 Method for inducing interpretable word sense representations from text;
- 5 Method for linking distributional word sense representations with lexical resources;
- 6 Model for generation of hypernyms;

# The obtained results

- 6 Method for post-processing of noisy hypernymy relations;

## The obtained results

- 6 Method for post-processing of noisy hypernymy relations;
- 7 Algorithm for construction of lexical semantic trees i.e. taxonomies;

# The obtained results

- 6 Method for post-processing of noisy hypernymy relations;
- 7 Algorithm for construction of lexical semantic trees i.e. taxonomies;
- 8 Model for node embeddings of linguistic graphs;

# The obtained results

- 6 Method for post-processing of noisy hypernymy relations;
- 7 Algorithm for construction of lexical semantic trees i.e. taxonomies;
- 8 Model for node embeddings of linguistic graphs;
- 9 Methods for neural lexical substitution;

# The obtained results

- 6 Method for post-processing of noisy hypernymy relations;
- 7 Algorithm for construction of lexical semantic trees i.e. taxonomies;
- 8 Model for node embeddings of linguistic graphs;
- 9 Methods for neural lexical substitution;
- 10 Study of distribution of lexical semantic relations provided by neural lexical substitution models.

# Content

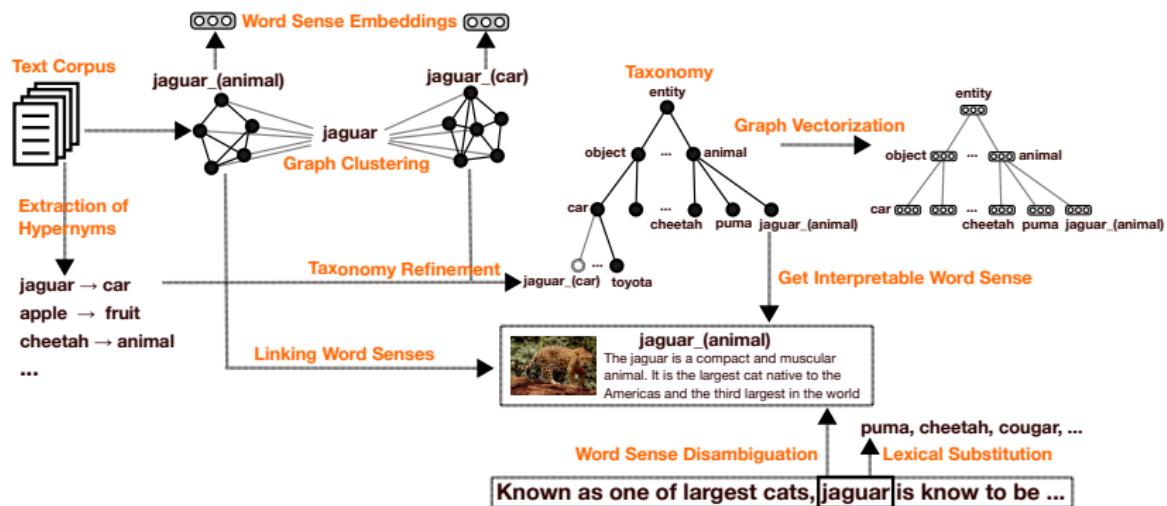


Figure: Overview of various methods for computational lexical semantics presented in this dissertation and their interrelations.

# The scope of dissertation is covered in 42 publications

## History

- Timeline:
  - The publication range from **2016** until **2023**

# The scope of dissertation is covered in 42 publications

## History

- Timeline:

- The publication range from **2016** until **2023**
- The research work started **February 2015**
- Dissertation preparation ended in **February 2024**.

# The scope of dissertation is covered in 42 publications

## History

- Timeline:
  - The publication range from **2016** until **2023**
  - The research work started **February 2015**
  - Dissertation preparation ended in **February 2024**.
- Thesis was prepared at Skoltech **based on publications prepared at:**

# The scope of dissertation is covered in 42 publications

## History

- Timeline:
  - The publication range from **2016** until **2023**
  - The research work started **February 2015**
  - Dissertation preparation ended in **February 2024**.
- Thesis was prepared at Skoltech **based on publications prepared at:**
  - Technical University of Darmstadt (TUDA): 2015-2016,
  - University of Hamburg (UHH): 2017-2019,
  - Skolkovo Institute of Science and Technology (Skoltech): 2019-2023,
  - Artificial Intelligence Research Institute (AIRI): 2023.

The scope of dissertation is covered in 42 publications

- 5 papers are published in **CORE A\***  
conferences [3, 5, 9, 10, 13];

## The scope of dissertation is covered in 42 publications

- **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];

## The scope of dissertation is covered in 42 publications

- **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 **Q1** journals [6, 8, 11, 12, 15];

## The scope of dissertation is covered in 42 publications

- **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 **Q1** journals [6, 8, 11, 12, 15];
- **1** paper is published in **CORE A\*** conference student track [28];

## The scope of dissertation is covered in 42 publications

- 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 **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];

## The scope of dissertation is covered in 42 publications

- **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 **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 [20, 26, 19, 27, 29];

## The scope of dissertation is covered in 42 publications

- 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 **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 [20, 26, 19, 27, 29];
- 11 papers indexed by **Scopus** published in proceedings of the  
main volumes of conferences  
[22, 23, 24, 25, 30, 31, 32, 33, 34, 40, 41];

## The scope of dissertation is covered in 42 publications

- 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 **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 [20, 26, 19, 27, 29];
- 11 papers indexed by **Scopus** published in proceedings of the main volumes of conferences [22, 23, 24, 25, 30, 31, 32, 33, 34, 40, 41];
- 8 papers indexed by **Scopus** published in workshops co-located with top conferences (CORE A\*/A) [35, 17, 21, 36, 37, 38, 39, 42].

## Selected 14 publications

- The defence and thesis summary is based on **14 publications** of these 42 overall published works.
- **10 first-tier** publications [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and
- **4 second-tier** publications [17, 18, 19, 20].

## Research was presented at various international venues

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

## Research was presented at various international venues

- 9 **EMNLP-2017** [CORE A] [18]: The 2017 Conference on Empirical Methods in Natural Language Processing (Copenhagen, Denmark)
- 10 **ISWC-2016** [CORE A] [7]: The 15th International Semantic Web Conference, (Kobe, Japan)
- 11 **NAACL-2019** [CORE A] [37, 38]: 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Minneapolis, Minnesota, USA)
- 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)
- 13 **AACL-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)
- 14 **LREC-2020** [CORE B] [20]: The 12th Language Resources and Evaluation Conference, (Marseille, France)
- 15 **LREC-2018** [CORE B] [26, 27, 19]: The 11th International Conference on Language Resources and Evaluation (LREC 2018), (Miyazaki, Japan), European Language Resources Association (ELRA), May 2018

## Research was presented at various international venues

- 16 **LREC-2016** [CORE B] [29]: The 10th International Conference on Language Resources and Evaluation (LREC16), (Portoroz, Slovenia).
- 17 **PaM-2020** [Scopus] [22]: The Probability and Meaning Conference (Gothenburg, Sweden)
- 18 **RANLP-2019** [Scopus] [33]: The International Conference on Recent Advances in Natural Language Processing (Varna, Bulgaria)
- 19 **GWC-2021** [Scopus] [41]: The 11th Global Wordnet Conference (Potchefstroom, South Africa)
- 20 **AIST-2019** [Scopus/Q2] [32]: The 8th International Conference on Analysis of Images, Social Networks and Texts (Kazan, Russia)
- 21 **AIST-2017** [Scopus/Q2] [30]: The 6th International Conference on Analysis of Images, Social Networks and Texts (Moscow, Russia)
- 22 **Dialogue-2018** [Scopus] [25, 24]: The 24th International Conference on Computational Linguistics and Intellectual Technologies (Moscow, Russia)
- 23 **KONVENS-2018** [Scopus] [23]: The 14th Conference on Natural Language Processing (Vienna, Austria).
- 24 **KONVENS-2016** [Scopus] [31]: The 13th Conference on Natural Language Processing, (Bochum, Germany)

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Local-Global Graph Clustering Algorithm

- Based on publications [6, 9, 10].

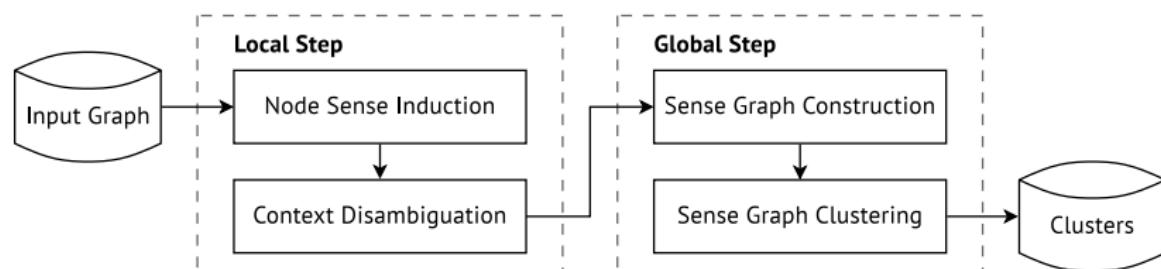
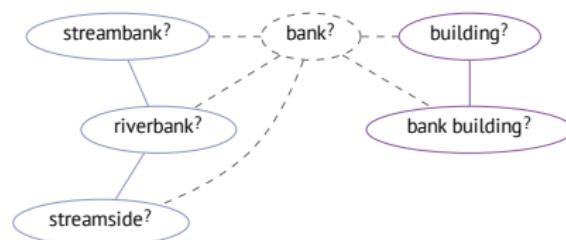


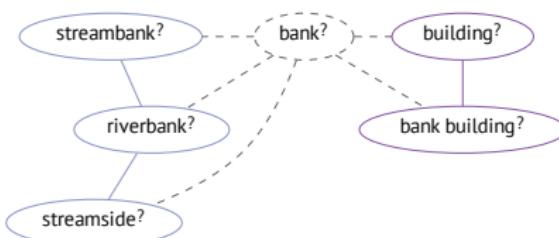
Figure: The outline of the algorithm showing the *local* step of node sense induction and context disambiguation, and the *global* step of sense graph constructing and clustering.

# Local-Global Graph Clustering Algorithm

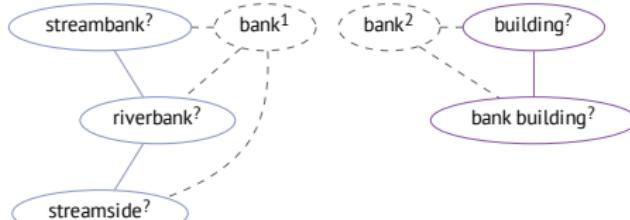


**Figure:** Clustering the neighborhood of the node “bank” of the input graph results in two clusters treated as the non-disambiguated sense contexts:  $\text{bank}^1 = \{\text{streambank}, \text{riverbank}, \dots\}$  and  $\{\text{bank}^2 = \text{bank building}, \text{building}, \dots\}$ .

# Local-Global Graph Clustering Algorithm



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



# Local-Global Graph Clustering Algorithm: Watset

**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  $C$ .

# Local-Global Graph Clustering Algorithm: Watset

**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  $C$ .

```
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:      $\text{senses}(u) \leftarrow \text{senses}(u) \cup \{u^i\}$ 
```

# Local-Global Graph Clustering Algorithm: Watset

**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  $C$ .

```

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

```

# Local-Global Graph Clustering Algorithm: Watset

**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  $C$ .

```

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
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:   $\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$ 
18:   $C \leftarrow \text{Cluster}_{\text{Global}}(\mathcal{G})$ 
19:   $C \leftarrow \{\{u \in V : \hat{u} \in C^i\} \subseteq V : C^i \in C\}$           ▷ Global Step: Sense Graph Construction
20: return  $C$                                       ▷ Global Step: Sense Graph Clustering
    
```

◀ □ ▶ ⏪ ⏩ Remove the sense labels ⏪ ⏩

# Local-Global Graph Clustering Algorithm: Simplified Watset\*

**Input:** graph  $G = (V, E)$ , hard clustering algorithms  $\text{Cluster}_{\text{Local}}$  and  $\text{Cluster}_{\text{Global}}$ .

**Output:** clusters  $C$ .

# Local-Global Graph Clustering Algorithm: Simplified Watset\*

**Input:** graph  $G = (V, E)$ , hard clustering algorithms  $\text{Cluster}_{\text{Local}}$  and  $\text{Cluster}_{\text{Global}}$ .

**Output:** clusters  $C$ .

```
1:  $\mathcal{V} \leftarrow \emptyset$ 
2: for all  $u \in V$  do                                ▷ Local Step: Sense Induction
   3:    $V_u \leftarrow \{v \in V : \{u, v\} \in E\}$ 
   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:     for all  $v \in C_u^i$  do
   9:        $\text{senses}[u][v] \leftarrow i$                   ▷ Node  $v$  is connected to the  $i$ -th sense of  $u$ 
10:       $\mathcal{V} \leftarrow \mathcal{V} \cup \{u^i\}$ 
```

# Local-Global Graph Clustering Algorithm: Simplified Watset\*

**Input:** graph  $G = (V, E)$ , hard clustering algorithms  $\text{Cluster}_{\text{Local}}$  and  $\text{Cluster}_{\text{Global}}$ .

**Output:** clusters  $C$ .

- 1:  $\mathcal{V} \leftarrow \emptyset$
- 2: **for all**  $u \in V$  **do**
  - 3:    $V_u \leftarrow \{v \in V : \{u, v\} \in E\}$
  - 4:    $E_u \leftarrow \{\{v, w\} \in E : v, w \in V_u\}$
  - 5:    $G_u \leftarrow (V_u, E_u)$
  - 6:    $C_u \leftarrow \text{Cluster}_{\text{Local}}(G_u)$
  - 7:   **for all**  $C_u^i \in C_u$  **do**
    - 8:     **for all**  $v \in C_u^i$  **do**
      - 9:        $\text{senses}[u][v] \leftarrow i$
    - 10:     $\mathcal{V} \leftarrow \mathcal{V} \cup \{u^i\}$
  - 11:  $\mathcal{E} \leftarrow \{u^{\text{senses}[u][v]}, v^{\text{senses}[v][u]}\} \in \mathcal{V}^2 : \{u, v\} \in E\}$
  - 12:  $\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$

▷ Local Step: Sense Induction

▷ Note that  $u \notin V_u$

▷ Cluster the open neighborhood of  $u$

▷ Node  $v$  is connected to the  $i$ -th sense of  $u$

▷ Global Step: Sense Graph Edges

▷ Global Step: Sense Graph Construction

# Local-Global Graph Clustering Algorithm: Simplified Watset\*

**Input:** graph  $G = (V, E)$ , hard clustering algorithms  $\text{Cluster}_{\text{Local}}$  and  $\text{Cluster}_{\text{Global}}$ .

**Output:** clusters  $C$ .

```

1:  $\mathcal{V} \leftarrow \emptyset$                                 ▷ Local Step: Sense Induction
2: for all  $u \in V$  do                            ▷ Note that  $u \notin V_u$ 
3:    $V_u \leftarrow \{v \in V : \{u, v\} \in E\}$ 
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:     for all  $v \in C_u^i$  do
9:        $\text{senses}[u][v] \leftarrow i$                   ▷ Node  $v$  is connected to the  $i$ -th sense of  $u$ 
10:       $\mathcal{V} \leftarrow \mathcal{V} \cup \{u^i\}$ 
11:  $\mathcal{E} \leftarrow \{\{u^{\text{senses}[u][v]}, v^{\text{senses}[v][u]}\} \in \mathcal{V}^2 : \{u, v\} \in E\}$  ▷ Global Step: Sense Graph Edges
12:  $\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$                       ▷ Global Step: Sense Graph Construction
13:  $C \leftarrow \text{Cluster}_{\text{Global}}(\mathcal{G})$         ▷ Global Step: Sense Graph Clustering
14:  $c \leftarrow \{\{u \in V : u \in C^i\} \subseteq \mathcal{V} : C^i \in C\}$  ▷ Remove the sense labels
15: return  $C$ 
```

\* Simplified version of the algorithm proposed by Dmitry Ustalov.

# Local-Global Graph Clustering Algorithm: Simplified Watset\*

Table: Node sense identifier tracking in Simplified Watset.

Source	Target	Index
bank	streambank	1
	riverbank	1
	streamside	1
	building	2
	bank building	2
streambank	bank	3
	riverbank	3
...		

\* Simplified version of the algorithm proposed by Dmitry Ustalov.

# Local-Global Graph Clustering Algorithm

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

Input Nodes	Input Edges	Output Structure	Linguistic
Polysemous words	Synonymy relationships	Synsets composed of disambiguated words	of
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	

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Word Sense Embeddings

- Based on publications [17, 20].

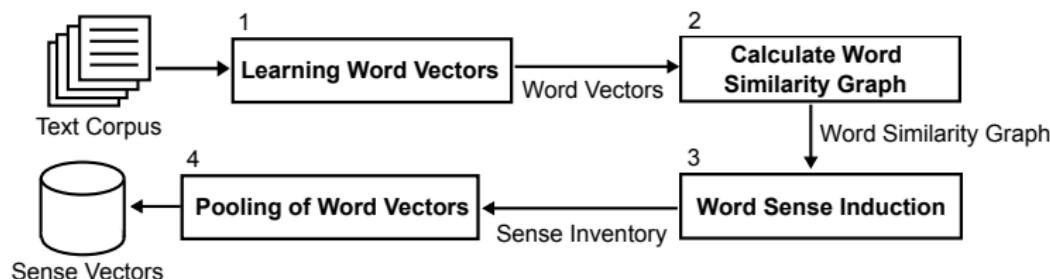


Figure: Schema of the word sense embeddings learning method “SenseGram”.

# Word Sense Embeddings

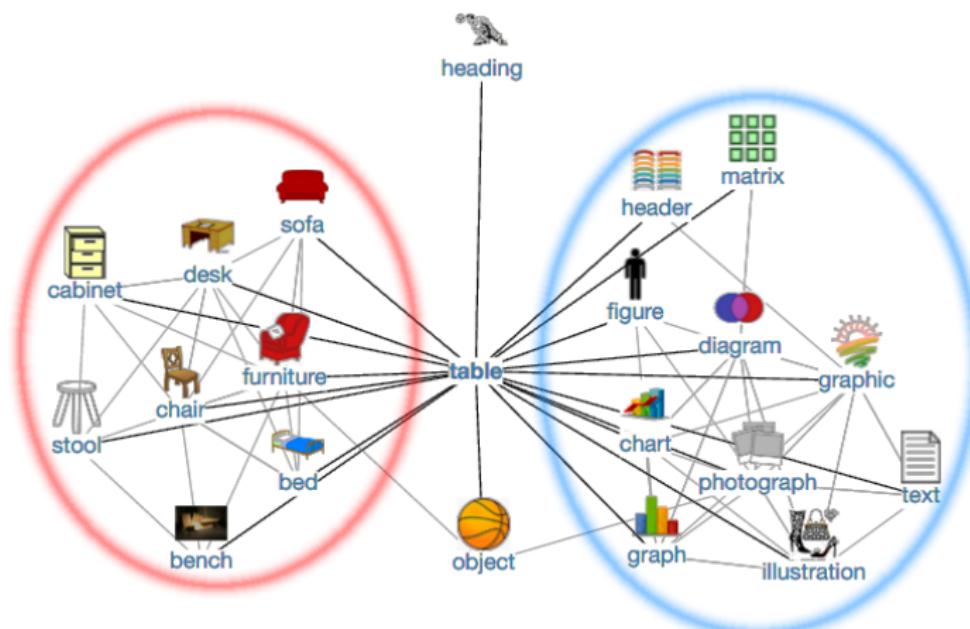


Figure: Visualization of the ego-network of “table”. The target “table” is excluded from clustering.

## 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

## 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  
10 foreach  $t \in T$  do  
11    $V \leftarrow N$  most similar terms of  $t$  from  $T$   
     $G \leftarrow$  graph with  $V$  as nodes and no edges  $E$ 
```

## Word Sense Induction: Baseline Algorithm

---

```
19 input :  $T$  – word similarity graph,  $N$  – ego-network size,  $n$  – ego-network
20           connectivity,  $k$  – minimum cluster size
21 output: for each term  $t \in T$ , a clustering  $S_t$  of its  $N$  most similar terms
22
23 19 foreach  $t \in T$  do
24       $V \leftarrow N$  most similar terms of  $t$  from  $T$ 
25       $G \leftarrow$  graph with  $V$  as nodes and no edges  $E$ 
26
27      20 foreach  $v \in V$  do
28           $V' \leftarrow n$  most similar terms of  $v$  from  $T$ 
29          foreach  $v' \in V'$  do
30              if  $v' \in V$  then add edge  $(v, v')$  to  $E$ 
31
32          21 end
33
34      22 end
35
36 23 end
```

## 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
28 foreach  $t \in T$  do
29    $V \leftarrow N$  most similar terms of  $t$  from  $T$ 
    $G \leftarrow$  graph with  $V$  as nodes and no edges  $E$ 
30   foreach  $v \in V$  do
31      $V' \leftarrow n$  most similar terms of  $v$  from  $T$ 
     foreach  $v' \in V'$  do
       if  $v' \in V$  then add edge  $(v, v')$  to  $E$ 
32   end
33 end
34
35    $S_t \leftarrow \text{ChineseWhispers}(G)$ 
    $S_t \leftarrow \{s \in S_t : |s| \geq k\}$ 
36 end
```

## Word Sense Induction: Improved Graph Construction

- 1 Extract a list  $\mathcal{N} = \{w_1, w_2, \dots, w_N\}$  of  $N$  nearest neighbours for the target (ego) word vector  $w$ .

## Word Sense Induction: Improved Graph Construction

- 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  $\delta_i = w - w_i$ .

## Word Sense Induction: Improved Graph Construction

- 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  $\delta_i = w - w_i$ .
- 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.

## Word Sense Induction: Improved Graph Construction

- 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  $\delta_i = w - w_i$ .
- 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.
  - $\overline{w_i}$  - is an **anti-node** of  $w_i$ : most similar to the target (ego) word  $w$  and least similar to its neighbour  $w_i$ .

## Word Sense Induction: Improved Graph Construction

- 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  $\delta_i = w - w_i$ .
- 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.
  - $\overline{w_i}$  - is an **anti-node** of  $w_i$ : most similar to the target (ego) word  $w$  and least similar to its neighbour  $w_i$ .
  - The set of  $N$  nearest neighbours and their anti-nodes form a set of **anti-edges** i.e. pairs of most dissimilar nodes:

## Word Sense Induction: Improved Graph Construction

- 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  $\delta_i = w - w_i$ .
- 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.
  - $\overline{w_i}$  - is an **anti-node** of  $w_i$ : most similar to the target (ego) word  $w$  and least similar to its neighbour  $w_i$ .
  - The set of  $N$  nearest neighbours and their anti-nodes form a set of **anti-edges** i.e. pairs of most dissimilar nodes:
  - These should not be connected:  
$$\overline{E} = \{(w_1, \overline{w_1}), (w_2, \overline{w_2}), \dots, (w_N, \overline{w_N})\}.$$

## Word Sense Induction: Improved Graph Construction

- 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  $\delta_i = w - w_i$ .
- 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.
  - $\overline{w_i}$  - is an **anti-node** of  $w_i$ : most similar to the target (ego) word  $w$  and least similar to its neighbour  $w_i$ .
  - The set of  $N$  nearest neighbours and their anti-nodes form a set of **anti-edges** i.e. pairs of most dissimilar nodes:
  - These should not be connected:  
$$\overline{E} = \{(w_1, \overline{w_1}), (w_2, \overline{w_2}), \dots, (w_N, \overline{w_N})\}$$
.
  - $w = \text{python}$ , its top similar term  $w_1 = \text{Java}$  and the anti-node  $\overline{w_i} = \text{snake}$  form an anti-edge  $(w_i, \overline{w_i}) = (\text{Java}, \text{snake})$ .

## Word Sense Induction: Improved Graph Construction

- 4 Construct graph  $G = (V, E)$  from the list of anti-edges  $\overline{E}$ , with the following recurrent procedure:

$$V = V \cup \{w_i, \overline{w_i} : w_i \in \mathcal{N}, \overline{w_i} \in \mathcal{N}\}$$

## Word Sense Induction: Improved Graph Construction

- 4 Construct graph  $G = (V, E)$  from the list of anti-edges  $\overline{E}$ , with the following recurrent procedure:

$$V = V \cup \{w_i, \overline{w}_i : w_i \in \mathcal{N}, \overline{w}_i \in \mathcal{N}\}$$

- Add a word from neighbours *and* its anti-node only if both of them are nearest neighbours of the original word  $w$ .

## Word Sense Induction: Improved Graph Construction

- 4 Construct graph  $G = (V, E)$  from the list of anti-edges  $\overline{E}$ , with the following recurrent procedure:

$$V = V \cup \{w_i, \overline{w}_i : w_i \in \mathcal{N}, \overline{w}_i \in \mathcal{N}\}$$

- Add a word from neighbours *and* its anti-node only if both of them are nearest neighbours of the original word  $w$ .
- Do not add  $w$ 's nearest neighbours if their anti-nodes do not belong to  $\mathcal{N}$ .

## Word Sense Induction: Improved Graph Construction

- 4 Construct graph  $G = (V, E)$  from the list of anti-edges  $\overline{E}$ , with the following recurrent procedure:

$$V = V \cup \{w_i, \overline{w}_i : w_i \in \mathcal{N}, \overline{w}_i \in \mathcal{N}\}$$

- Add a word from neighbours *and* its anti-node only if both of them are nearest neighbours of the original word  $w$ .
- Do not add  $w$ 's nearest neighbours if their anti-nodes do not belong to  $\mathcal{N}$ .
- Intuition: add only words which can help discriminating between different senses of  $w$ .

## Word Sense Induction: Improved Graph Construction

- 4 Construct graph  $G = (V, E)$  from the list of anti-edges  $\overline{E}$ , with the following recurrent procedure:

$$V = V \cup \{w_i, \overline{w}_i : w_i \in \mathcal{N}, \overline{w}_i \in \mathcal{N}\}$$

- Add a word from neighbours *and* its anti-node only if both of them are nearest neighbours of the original word  $w$ .
- Do not add  $w$ 's nearest neighbours if their anti-nodes do not belong to  $\mathcal{N}$ .
- Intuition: add only words which can help discriminating between different senses of  $w$ .

- 5 Construct the set of edges  $E$ : For each  $w_i \in \mathcal{N}$  extract a set of its  $K$  nearest neighbours  $\mathcal{N}'_i = \{u_1, u_2, \dots, u_K\}$  and define  $E = \{(w_i, u_j) : w_i \in V, u_j \in V, u_j \in \mathcal{N}'_i, u_j \neq \overline{w}_i\}$ .

## Word Sense Induction: Improved Graph Construction

- 4 Construct graph  $G = (V, E)$  from the list of anti-edges  $\overline{E}$ , with the following recurrent procedure:

$$V = V \cup \{w_i, \overline{w}_i : w_i \in \mathcal{N}, \overline{w}_i \in \mathcal{N}\}$$

- Add a word from neighbours *and* its anti-node only if both of them are nearest neighbours of the original word  $w$ .
- Do not add  $w$ 's nearest neighbours if their anti-nodes do not belong to  $\mathcal{N}$ .
- Intuition: add only words which can help discriminating between different senses of  $w$ .

- 5 Construct the set of edges  $E$ : For each  $w_i \in \mathcal{N}$  extract a set of its  $K$  nearest neighbours  $\mathcal{N}'_i = \{u_1, u_2, \dots, u_K\}$  and define  $E = \{(w_i, u_j) : w_i \in V, u_j \in V, u_j \in \mathcal{N}'_i, u_j \neq \overline{w}_i\}$ .
- Do not add edges corresponding to anti-nodes if any.

# Word Sense Embeddings

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

# Word Sense Embeddings

- Get Sense Embeddings by pooling of word vectors:

$$\mathbf{s}_i = \frac{\sum_{w \in S_t} \gamma_i(w_k) \text{vec}_w(w_k)}{\sum_{s \in S_t} \gamma_i(w_k)}.$$

# Word Sense Embeddings

- Get Sense Embeddings by pooling of word vectors:

$$\mathbf{s}_i = \frac{\sum_{w \in S_t} \gamma_i(w_k) \text{vec}_w(w_k)}{\sum_{s \in S_t} \gamma_i(w_k)}.$$

- Word Sense Disambiguation:

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

$$\text{where } \bar{\mathbf{c}}_w = k^{-1} \sum_{i=1}^k \text{vec}_w(c_i).$$

# Word Sense Embeddings

- Get Sense Embeddings by pooling of word vectors:

$$\mathbf{s}_i = \frac{\sum_{w \in S_t} \gamma_i(w_k) \text{vec}_w(w_k)}{\sum_{s \in S_t} \gamma_i(w_k)}.$$

- Word Sense Disambiguation:

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

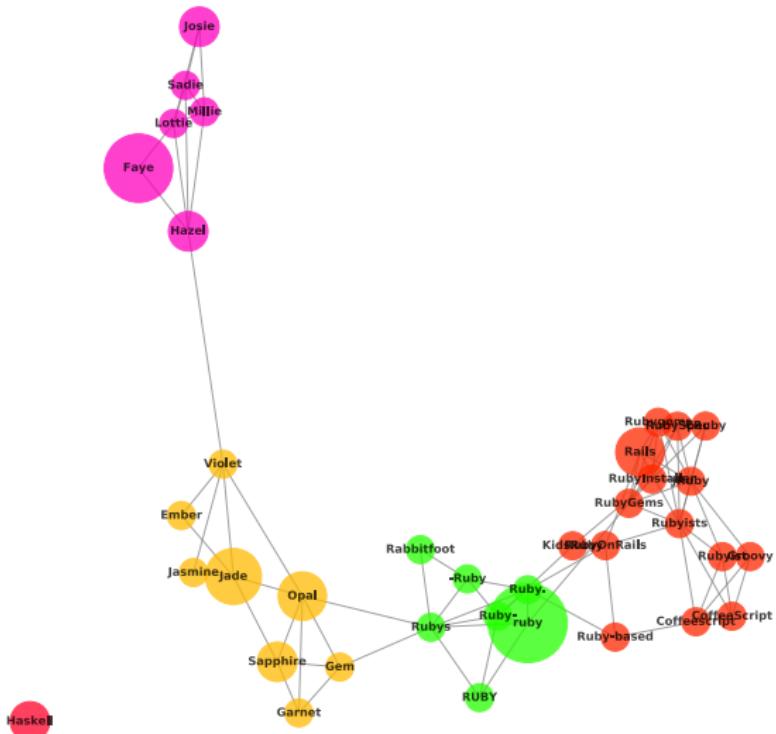
where  $\bar{\mathbf{c}}_w = k^{-1} \sum_{i=1}^k \text{vec}_w(c_i)$ .

- Knowledge-Free Labelling of Induced Sense Clusters:

$$\text{keyness}(v) = |\{(w_i, \overline{w_i}) : (w_i, \overline{w_i}) \in \overline{E} \wedge (v = w_i \vee v = \overline{w_i})\}|,$$

.. is number of anti-edges among words in this cluster.

# Word Sense Embeddings



# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Unsupervised Interpretable Word Sense Disambiguation

- Based on publications [1] and [18].

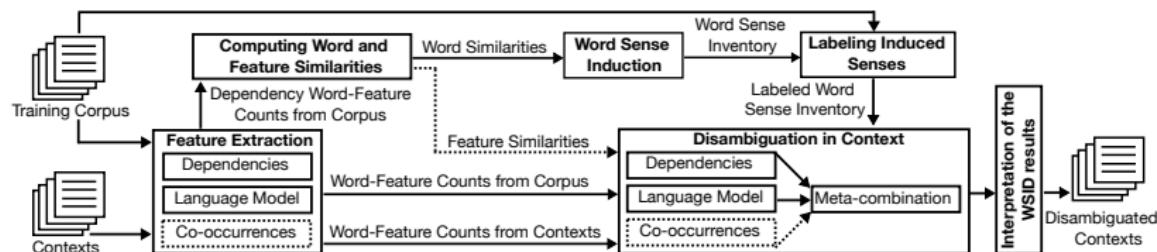
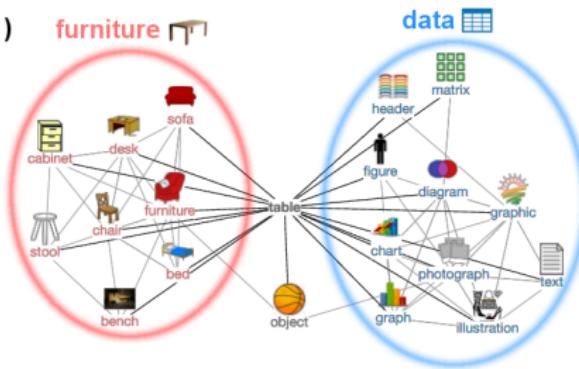


Figure: Outline of unsupervised interpretable method for word sense induction and disambiguation.

# Unsupervised Interpretable Word Sense Disambiguation

(1) furniture



(2) data

furniture	data
seating#NN#conj_and	graph#NN#-conj_and
counter#NN#-conj_and	index#NN#-conj_and
bench#NN#-conj_and	knot#NN#nn
folding#JJ#amod	row#NN#prep_with
desk#NN#-conj_and	graph#NN#appos
lunch#NN#nn	above#JJ#dep
furnish#VB#-prep_with	above#RB#-npadvmod
sofa#NN#-conj_and	diagram#NN#conj_and
booth#NN#-conj_and	below#N#amod
stool#NN#conj_and	drew#VB#amod
armchair#NN#-conj_and	reproduce#VB#rcmod
...	...

(3)

Below is a table of **data | LM**, **data | cooc**, **furniture | cooc**, **data | cooc**, **data | cooc**. information, which lists all cookies deployed and used on our website.

**Figure:** Interpretation of the senses of the word “table” at three levels: (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”.

# Unsupervised Interpretable Word Sense Disambiguation

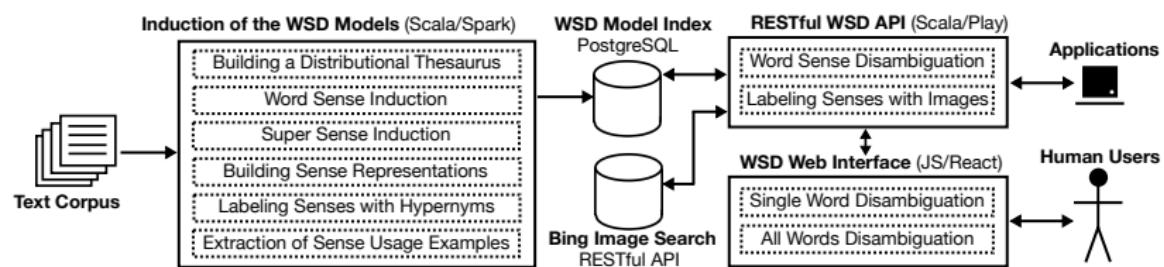


Figure: Software and functional architecture of the WSD system.

# Unsupervised Interpretable Word Sense Disambiguation

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 wildlife bird mammal D

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 lion tiger leopard wolf monkey otter crocodile alligator deer cat elephant fox eagle owl snake E

Context words 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 F

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 G ^ SHOW LESS H

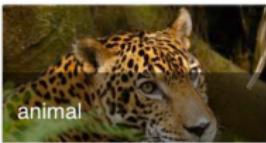
# Unsupervised Interpretable Word Sense Disambiguation

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.

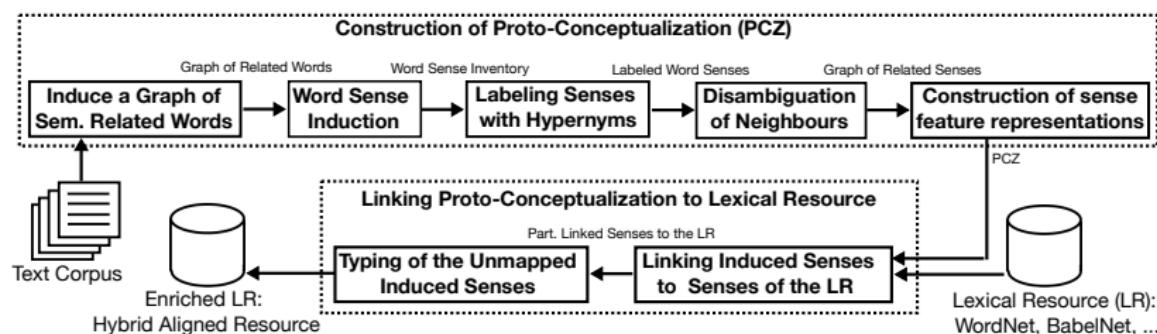
 animal Jaguar <span style="color:red; border:1px solid red; border-radius:50%; padding:2px 5px;">D</span>	 animal predator <span style="color:red; border:1px solid red; border-radius:50%; padding:2px 5px;">D</span>	 country America <span style="color:red; border:1px solid red; border-radius:50%; padding:2px 5px;">D</span>
is a large spotted	of tropical	

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 **Linking Word Sense Representations**
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Linking Word Sense Representations

- Based on publications [7, 8].



**Figure:** 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.

# Linking Word Sense Representations

- 1 **LR**  $W$ : lexical resource e.g. WordNet or BabelNet;

# Linking Word Sense Representations

- 1 **LR**  $W$ : lexical resource e.g. WordNet or BabelNet;
- 2 **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.  $H_{j_i} = \{\text{equipment:3, ...}\}$ );

# Linking Word Sense Representations

- 1 **LR**  $W$ : lexical resource e.g. WordNet or BabelNet;
- 2 **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.  $R_{j_i} = \{\text{keyboard:1}, \text{computer:0}, \dots\}$ ) and  $H_{j_i}$  the set of its hypernym senses (i.e.  $H_{j_i} = \{\text{equipment:3}, \dots\}$ );
- 3 **Threshold**  $th$  over the similarity between pairs of concepts and a number  $m$  of iterations as a stopping criterion.

# Linking Word Sense Representations

- 1 **LR**  $W$ : lexical resource e.g. WordNet or BabelNet;
- 2 **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.  $R_{j_i} = \{\text{keyboard:1}, \text{computer:0}, \dots\}$ ) and  $H_{j_i}$  the set of its hypernym senses (i.e.  $H_{j_i} = \{\text{equipment:3}, \dots\}$ );
- 3 **Threshold**  $th$  over the similarity between pairs of concepts and a number  $m$  of iterations as a stopping criterion.
- 4 **Mapping**  $M$ : set of pairs of the kind  $(source, target)$  where  $source \in T.\text{senses}$  is a sense of the input PCZ  $T$  and  $target \in W.\text{senses} \cup source$  is the most suitable sense of  $W$ .

# Linking Word Sense Representations

- 1 **LR**  $W$ : lexical resource e.g. WordNet or BabelNet;
- 2 **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.  $R_{j_i} = \{\text{keyboard:1}, \text{computer:0}, \dots\}$ ) and  $H_{j_i}$  the set of its hypernym senses (i.e.  $H_{j_i} = \{\text{equipment:3}, \dots\}$ );
- 3 **Threshold**  $th$  over the similarity between pairs of concepts and a number  $m$  of iterations as a stopping criterion.
- 4 **Mapping**  $M$ : set of pairs of the kind  $(source, target)$  where  $source \in T.\text{senses}$  is a sense of the input PCZ  $T$  and  $target \in W.\text{senses} \cup source$  is the most suitable sense of  $W$ .
- 5 Compare the two senses with similarity function:

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

# Linking Word Sense Representations

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

**Output:**  $M = (\text{source}, \text{target})$

$$M = \emptyset$$

# Linking Word Sense Representations

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

**Output:**  $M = (\text{source}, \text{target})$

$M = \emptyset$

**for all**  $(j_i, R_{j_i}, H_{j_i}) \in T.\text{monosemousSenses}$  **do**

$C(j_i) = W.\text{getSenses}(j_i.\text{lemma}, j_i.\text{POS})$

**if**  $|C(j_i)| == 1$ , let  $C(j_i) = \{c_0\}$  **then**

**if**  $\text{sim}(j_i, c_0, \emptyset) \geq th$  **then**

$M = M \cup \{(j_i, c_0)\}$

# Linking Word Sense Representations

```

Input:  $T = \{(j_i, R_{j_i}, H_{j_i})\}$ ,  $W$ ,  $th$ ,  $m$ 
Output:  $M = (\text{source}, \text{target})$ 
 $M = \emptyset$ 
for all  $(j_i, R_{j_i}, H_{j_i}) \in T.\text{monosemousSenses}$  do
     $C(j_i) = W.getSenses(j_i.\text{lemma}, j_i.\text{POS})$ 
    if  $|C(j_i)| == 1$ , let  $C(j_i) = \{c_0\}$  then
        if  $\text{sim}(j_i, c_0, \emptyset) \geq th$  then
             $M = M \cup \{(j_i, c_0)\}$ 
for  $\text{step} = 1$ ;  $\text{step} \leq m$ ;  $\text{step} = \text{step} + 1$  do
     $M_{\text{step}} = \emptyset$ 
    for all  $(j_i, R_{j_i}, H_{j_i}) \in T.\text{senses}/M.\text{senses}$  do
         $C(j_i) = W.getSenses(j_i.\text{lemma}, j_i.\text{POS})$ 
        for all  $c_k \in C(j_i)$  do
             $\text{rank}(c_k) = \text{sim}(j_i, c_k, M)$ 
        if  $\text{rank}(c_k)$  has a single top value for  $c_t$  then
            if  $\text{rank}(c_t) \geq th$  then
                 $M_{\text{step}} = M_{\text{step}} \cup \{(j_i, c_t)\}$ 
     $M = M \cup M_{\text{step}}$ 

```

# Linking Word Sense Representations

```

Input:  $T = \{(j_i, R_{j_i}, H_{j_i})\}$ ,  $W$ ,  $th$ ,  $m$ 
Output:  $M = (\text{source}, \text{target})$ 
 $M = \emptyset$ 
for all  $(j_i, R_{j_i}, H_{j_i}) \in T.\text{monosemousSenses}$  do
     $C(j_i) = W.getSenses(j_i.\text{lemma}, j_i.\text{POS})$ 
    if  $|C(j_i)| == 1$ , let  $C(j_i) = \{c_0\}$  then
        if  $\text{sim}(j_i, c_0, \emptyset) \geq th$  then
             $M = M \cup \{(j_i, c_0)\}$ 
for  $\text{step} = 1$ ;  $\text{step} \leq m$ ;  $\text{step} = \text{step} + 1$  do
     $M_{\text{step}} = \emptyset$ 
    for all  $(j_i, R_{j_i}, H_{j_i}) \in T.\text{senses}/M.\text{senses}$  do
         $C(j_i) = W.getSenses(j_i.\text{lemma}, j_i.\text{POS})$ 
        for all  $c_k \in C(j_i)$  do
             $\text{rank}(c_k) = \text{sim}(j_i, c_k, M)$ 
        if  $\text{rank}(c_k)$  has a single top value for  $c_t$  then
            if  $\text{rank}(c_t) \geq th$  then
                 $M_{\text{step}} = M_{\text{step}} \cup \{(j_i, c_t)\}$ 
     $M = M \cup M_{\text{step}}$ 
for all  $(j_i, R_{j_i}, H_{j_i}) \in T.\text{senses}/M.\text{senses}$  do
     $M = M \cup \{(j_i, j_i)\}$ 
return  $M$ 

```

# Linking Word Sense Representations

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:** 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.

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Prediction of Hypernym Embeddings

- Based on publications [4].

## Introduction

- **Hypernymy** represent hierarchical relations between terms:
  - (apple, **is-a**, fruit): apple = **hyponym**, fruit = **hypernym**;
  - (jaguar, **is-a**, animal).

# Prediction of Hypernym Embeddings

- Based on publications [4].

## Introduction

- **Hypernymy** represent hierarchical relations between terms:
  - (apple, **is-a**, fruit): apple = **hyponym**, fruit = **hypernym**;
  - (jaguar, **is-a**, animal).
- **Baseline approach** [43]. A projection matrix  $\Phi^*$  is obtained:
  - given vectors  $\vec{x}$  and  $\vec{y}$  representing hyponym and hypernym

# Prediction of Hypernym Embeddings

- Based on publications [4].

## Introduction

- **Hypernymy** represent hierarchical relations between terms:
  - (apple, **is-a**, fruit): apple = **hyponym**, fruit = **hypernym**;
  - (jaguar, **is-a**, animal).
- **Baseline approach** [43]. A projection matrix  $\Phi^*$  is obtained:
  - given vectors  $\vec{x}$  and  $\vec{y}$  representing hyponym and hypernym
  - a square matrix  $\Phi^*$  is fit

# Prediction of Hypernym Embeddings

- Based on publications [4].

## Introduction

- **Hypernymy** represent hierarchical relations between terms:
  - (apple, **is-a**, fruit): apple = **hyponym**, fruit = **hypernym**;
  - (jaguar, **is-a**, animal).
- **Baseline approach** [43]. A projection matrix  $\Phi^*$  is obtained:
  - given vectors  $\vec{x}$  and  $\vec{y}$  representing hyponym and hypernym
  - a square matrix  $\Phi^*$  is fit
  - on the training set of positive pairs  $\mathcal{P}$ :

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

# Prediction of Hypernym Embeddings

Hypernymy extraction via **regularized** projection learning.

- Linguistic Constraints via Regularization

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

- **Asymmetric Regularization.** Enforces the asymmetry: the same transformation to the predicted hypernym should not provide a vector similar to the initial hyponym:

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

# Prediction of Hypernym Embeddings

Hypernymy extraction via **regularized** projection learning.

- **Neighbor Regularization.** Negative sampling by explicitly providing the examples of semantically related words  $\vec{z}$  of the hyponym  $\vec{x}$ : penalizes the model to produce similar vectors:

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

- We use synonyms of hyponyms as  $\mathcal{N}$
- **Regularizers without Re-Projection.** The neighbor regularizer:

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

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hypernyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Extracting of Hypernyms via Sense Graph Clustering

- Based on publication [19].

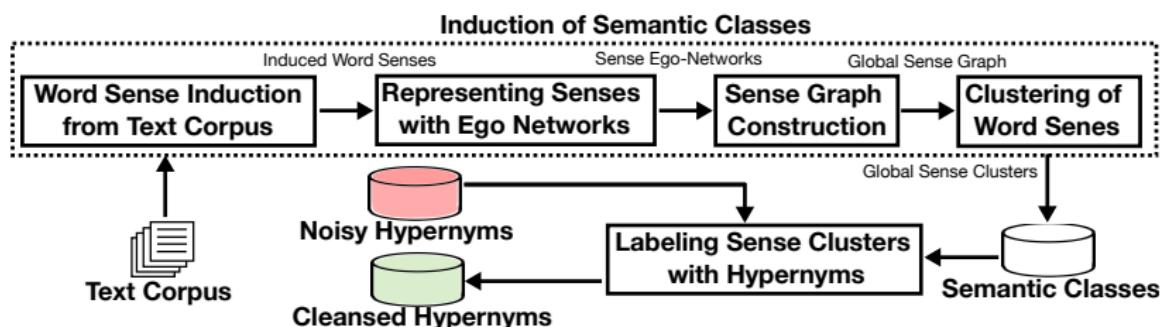


Figure: Outline of the approach: sense-aware distributional semantic classes are induced from a text corpus and then used to filter noisy hypernyms database (e.g. extracted by an external method from a text corpus).

# Extracting of Hypernyms via Sense Graph Clustering

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, grapefruit#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:** Sample of the induced sense clusters representing “fruits” and “programming language” semantic classes.

# Extracting of Hypernyms via Sense Graph Clustering

## Representing Senses with Ego Networks

- 1 Represent each induced sense  $s$  by a second-order **ego network** consisting of related senses  $\mathcal{R}(s)$  of the ego sense  $s$ :

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

# Extracting of Hypernyms via Sense Graph Clustering

## Representing Senses with Ego Networks

- 1 Represent each induced sense  $s$  by a second-order **ego network** consisting of related senses  $\mathcal{R}(s)$  of the ego sense  $s$ :

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

- 2 Edge weight  $\mathcal{W}_s(s_i, s_j)$  between two senses is equal to a distributional semantic relatedness score between  $s_i$  and  $s_j$ .

# Extracting of Hypernyms via Sense Graph Clustering

## Representing Senses with Ego Networks

- 1 Represent each induced sense  $s$  by a second-order **ego network** consisting of related senses  $\mathcal{R}(s)$  of the ego sense  $s$ :

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

- 2 Edge weight  $\mathcal{W}_s(s_i, s_j)$  between two senses is equal to a distributional semantic relatedness score between  $s_i$  and  $s_j$ .
- 3 Cluster each ego network and discard networks for which the cluster containing the target sense  $s$  contains less than 80% nodes of the respective network to ensure semantic coherence.

# Extracting of Hypernyms via Sense Graph Clustering

## Global Sense Graph Construction

- 1 Compute weights of the edges of the global graph by counting the number of co-occurrences of the edge in ego networks:

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

- 2 To filter noisy edges and re-scale weights:

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

# Extracting of Hypernyms via Sense Graph Clustering

## Clustering of Word Senses

- **Fine-grained:** 208,871 word senses  $\Rightarrow$  1,870 semantic classes,
- **Coarse-grained:** 18,028 word senses  $\Rightarrow$  734 semantic classes.

# Extracting of Hypernyms via Sense Graph Clustering

## Clustering of Word Senses

- **Fine-grained:** 208,871 word senses  $\Rightarrow$  1,870 semantic classes,
- **Coarse-grained:** 18,028 word senses  $\Rightarrow$  734 semantic classes.

## Denoising Hypernyms using the Distributional Semantic Classes

- Sense cluster is labeled with top 5 common hypernyms.

# Extracting of Hypernyms via Sense Graph Clustering

## Clustering of Word Senses

- **Fine-grained:** 208,871 word senses  $\Rightarrow$  1,870 semantic classes,
- **Coarse-grained:** 18,028 word senses  $\Rightarrow$  734 semantic classes.

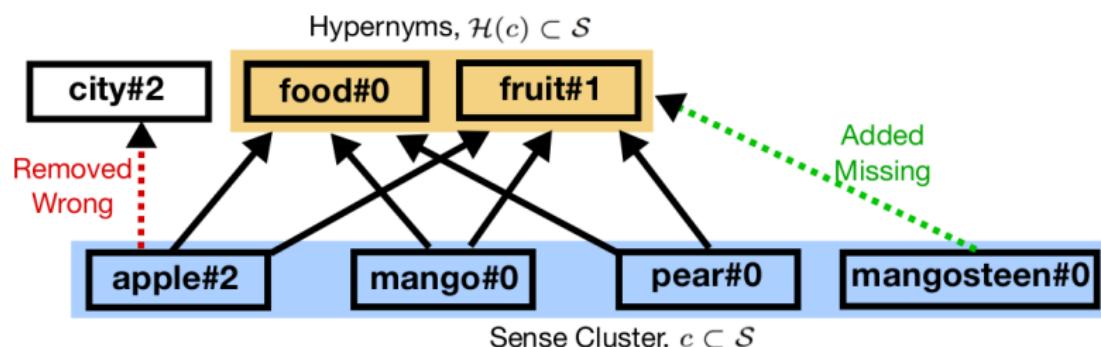
## Denoising Hypernyms using the Distributional Semantic Classes

- Sense cluster is labeled with top 5 common hypernyms.
- For labeling we used the tf-idf weighting:

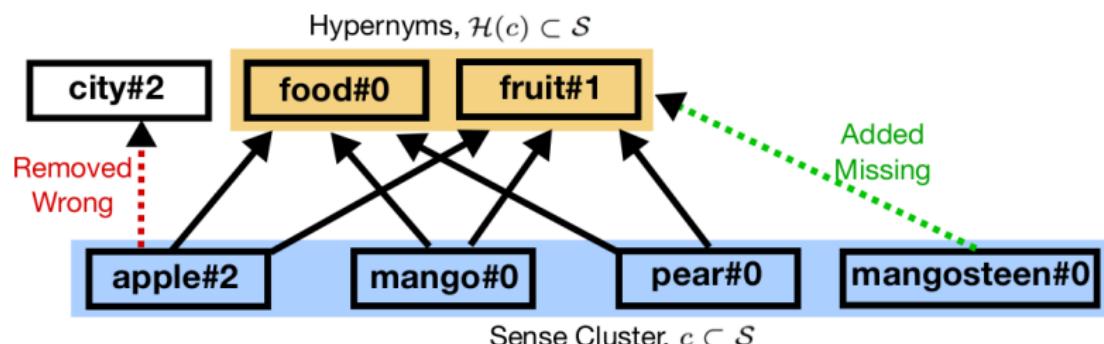
$$\text{tf-idf}(h) = \sum_{s \in c} \mathcal{H}(s) \cdot \log \frac{|\mathcal{S}|}{|h \in \mathcal{H}(s) : \forall s \in \mathcal{S}|},$$

where  $\sum_{s \in c} \mathcal{H}(s)$  is a sum of weights for all hypernyms  $s$ .

# Extracting of Hypernyms via Sense Graph Clustering

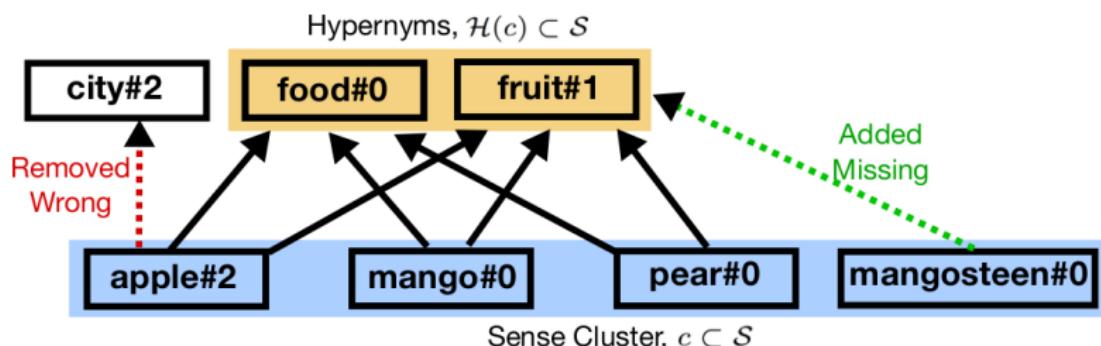


# Extracting of Hypernyms via Sense Graph Clustering



- Post-processing of hypernymy relations using distributionally induced semantic classes, represented by clusters of induced word senses labeled with noisy hypernyms.

# Extracting of Hypernyms via Sense Graph Clustering



- Post-processing of hypernymy relations using distributionally induced semantic classes, represented by clusters of induced word senses labeled with noisy hypernyms.
  - Wrong hypernyms outside the cluster labels are removed, while the missing ones not present in the noisy database of hypernyms are added.

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 **Taxonomy Enrichment using Hyperbolic Embeddings**
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Taxonomy Enrichment using Hyperbolic Embeddings

- Based on publication [5].

## Introduction

- Given a noisy taxonomic graph  $G = (V, E)$  with errors:

# Taxonomy Enrichment using Hyperbolic Embeddings

- Based on publication [5].

## Introduction

- Given a noisy taxonomic graph  $G = (V, E)$  with errors:
  - **Absent edges:**  $E_{abs} = \{(v_i, v_j) \in E : v_i \text{ is-a } v_j \wedge (v_i, v_j) \notin E\}$ .  
Special case is **orphan nodes:**  $V_{orh} = \{v_i \in V : \nexists (v_i, v_j) \in E\}$ .

# Taxonomy Enrichment using Hyperbolic Embeddings

- Based on publication [5].

## Introduction

- Given a noisy taxonomic graph  $G = (V, E)$  with errors:
  - **Absent edges:**  $E_{abs} = \{(v_i, v_j) \in E : v_i \text{ is-a } v_j \wedge (v_i, v_j) \notin E\}$ .  
Special case is **orphan nodes**:  $V_{orh} = \{v_i \in V : \nexists (v_i, v_j) \in E\}$ .
  - **Wrong edges:**  $E_{wrg} = \{(v_i, v_j) \in E : v_i \text{ not-is-a } v_j\}$ .

# Taxonomy Enrichment using Hyperbolic Embeddings

- Based on publication [5].

## Introduction

- Given a noisy taxonomic graph  $G = (V, E)$  with errors:
  - **Absent edges:**  $E_{abs} = \{(v_i, v_j) \in E : v_i \text{ is-a } v_j \wedge (v_i, v_j) \notin E\}$ .  
Special case is **orphan nodes**:  $V_{orh} = \{v_i \in V : \nexists (v_i, v_j) \in E\}$ .
  - **Wrong edges:**  $E_{wrg} = \{(v_i, v_j) \in E : v_i \text{ not-is-a } v_j\}$ .
- Build a graph  $G' = (V, E')$  correcting the two edge errors by:

# Taxonomy Enrichment using Hyperbolic Embeddings

- Based on publication [5].

## Introduction

- Given a noisy taxonomic graph  $G = (V, E)$  with errors:
  - **Absent edges:**  $E_{abs} = \{(v_i, v_j) \in E : v_i \text{ is-a } v_j \wedge (v_i, v_j) \notin E\}$ .  
Special case is **orphan nodes**:  $V_{orh} = \{v_i \in V : \nexists (v_i, v_j) \in E\}$ .
  - **Wrong edges:**  $E_{wrg} = \{(v_i, v_j) \in E : v_i \text{ not-is-a } v_j\}$ .
- Build a graph  $G' = (V, E')$  correcting the two edge errors by:
  - **Adding absent** edges:  $E = E \cup \{E_{abs}\}$  and
  - **Removing wrong** edges:  $E = E \setminus E_{wrg}$

# Taxonomy Enrichment using Hyperbolic Embeddings

- Based on publication [5].

## Introduction

- Given a noisy taxonomic graph  $G = (V, E)$  with errors:
  - **Absent edges:**  $E_{abs} = \{(v_i, v_j) \in E : v_i \text{ is-a } v_j \wedge (v_i, v_j) \notin E\}$ .  
Special case is **orphan nodes**:  $V_{orh} = \{v_i \in V : \nexists (v_i, v_j) \in E\}$ .
  - **Wrong edges:**  $E_{wrg} = \{(v_i, v_j) \in E : v_i \text{ not-is-a } v_j\}$ .
- Build a graph  $G' = (V, E')$  correcting the two edge errors by:
  - **Adding absent** edges:  $E = E \cup \{E_{abs}\}$  and
  - **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
    - **Relocation** i.e. a combination of removing wrong with adding absent edge(s) is needed.

# Taxonomy Enrichment using Hyperbolic Embeddings

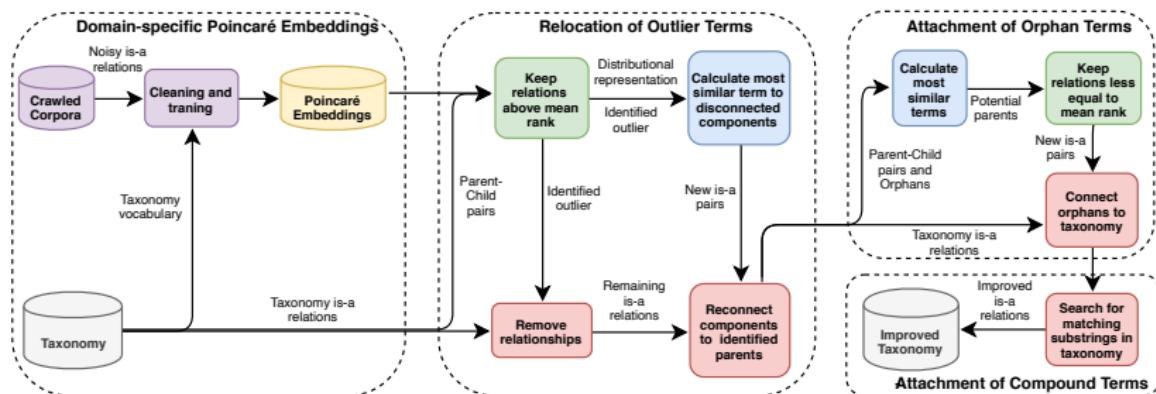


Figure: Outline of the taxonomy refinement method.

# Taxonomy Enrichment using Hyperbolic Embeddings

## Hypernym-Hyponym Distance

- Poincaré embeddings are trained on extracted from text IS-A relations or WordNet.

# Taxonomy Enrichment using Hyperbolic Embeddings

## Hypernym-Hyponym Distance

- Poincaré embeddings are trained on extracted from text IS-A relations or WordNet.
- **Modelling co-hyponyms:** Distance between two terms  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$  in Euclidean space (e.g.  $d$ -dimensional Skip-gram word embeddings):

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

# Taxonomy Enrichment using Hyperbolic Embeddings

## Hypernym-Hyponym Distance

- Poincaré embeddings are trained on extracted from text IS-A relations or WordNet.
- **Modelling co-hyponyms:** Distance between two terms  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$  in Euclidean space (e.g.  $d$ -dimensional Skip-gram word embeddings):

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

- **Modelling hypernyms:** Distance between two terms  $\mathbf{u}, \mathbf{v} \in \mathcal{B}^d$  for a  $d$ -dimensional Poincaré Ball model:

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

# Taxonomy Enrichment using Hyperbolic Embeddings

## Relocation of Outlier Terms

- 1 Poincaré embeddings are used to compute ranks between every child-parent of the existing taxonomy.

# Taxonomy Enrichment using Hyperbolic Embeddings

## Relocation of Outlier Terms

- 1 Poincaré embeddings are used to compute ranks between every child-parent of the existing taxonomy.
- 2 Hypernym-hyponym relationships with a rank larger than the mean of all ranks are removed.

# Taxonomy Enrichment using Hyperbolic Embeddings

## Relocation of Outlier Terms

- 1 Poincaré embeddings are used to compute ranks between every child-parent of the existing taxonomy.
- 2 Hypernym-hyponym relationships with a rank larger than the mean of all ranks are removed.
- 3 Disconnected components that have children are re-connected to the most similar parent in the taxonomy or to the taxonomy root.

# Taxonomy Enrichment using Hyperbolic Embeddings

## Relocation of Outlier Terms

- 1 Poincaré embeddings are used to compute ranks between every child-parent of the existing taxonomy.
- 2 Hypernym-hyponym relationships with a rank larger than the mean of all ranks are removed.
- 3 Disconnected components that have children are re-connected to the most similar parent in the taxonomy or to the taxonomy root.
- 4 Previously or now disconnected isolated nodes are subject to orphan attachment.

# Taxonomy Enrichment using Hyperbolic Embeddings

## Relocation of Outlier Terms

- 1 Poincaré embeddings are used to compute ranks between every child-parent of the existing taxonomy.
- 2 Hypernym-hyponym relationships with a rank larger than the mean of all ranks are removed.
- 3 Disconnected components that have children are re-connected to the most similar parent in the taxonomy or to the taxonomy root.
- 4 Previously or now disconnected isolated nodes are subject to orphan attachment.
- 5 Compute distance to the closest co-hyponym for every node to identify and relocate outliers.

# Taxonomy Enrichment using Hyperbolic Embeddings

## Attachment of Orphan Terms

- 1 Attach orphans by computing the rank between every orphan and the most similar node in the taxonomy.

# Taxonomy Enrichment using Hyperbolic Embeddings

## Attachment of Orphan Terms

- 1 Attach orphans by computing the rank between every orphan and the most similar node in the taxonomy.
- 2 Only hypernym-hyponym relationships with a rank lower or equal to the mean of all stored ranks are added to the taxonomy.

# Taxonomy Enrichment using Hyperbolic Embeddings

## Attachment of Orphan Terms

- 1 Attach orphans by computing the rank between every orphan and the most similar node in the taxonomy.
- 2 Only hypernym-hyponym relationships with a rank lower or equal to the mean of all stored ranks are added to the taxonomy.
- 3 For Euclidean embeddings, a link is added between the parent of the most similar co-hyponym and the orphan.

# Taxonomy Enrichment using Hyperbolic Embeddings

Word	Parent patterns	Parent refinement	after	Gold parent	Closest neighbors
second language acquisition	—	linguistics	linguistics	applied semantics, linguistics	linguistics, semantics, linguistics
botany	—	genetics	plant ecology	science,	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:** 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.

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Node Embeddings of Lexical-Semantic Graphs

- Based on publication [3].

## Introduction

- Custom **graph node similarity measures**  $sim : V \times V \rightarrow \mathbb{R}$  have been defined on pairs of nodes  $V$  of a graph  $G = (V, E)$ .

# Node Embeddings of Lexical-Semantic Graphs

- Based on publication [3].

## Introduction

- 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: travel time, community, or knowledge graph based semantic distances, etc.

# Node Embeddings of Lexical-Semantic Graphs

- Based on publication [3].

## Introduction

- 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: travel time, community, or knowledge graph based semantic distances, etc.
- 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:

# Node Embeddings of Lexical-Semantic Graphs

- Based on publication [3].

## Introduction

- 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: travel time, community, or knowledge graph based semantic distances, etc.
- 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:
  - cup → container ← vessel ← drinking\_vessel ← mug.

# Node Embeddings of Lexical-Semantic Graphs

- Based on publication [3].

## Introduction

- 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: travel time, community, or knowledge graph based semantic distances, etc.
- 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:
  - cup → container ← vessel ← drinking\_vessel ← mug.
  - Computing directly on the graph can be prohibitively computationally expensive.

# Node Embeddings of Lexical-Semantic Graphs

- path2vec is a **graph metric embeddings model** learns embeddings of the graph nodes  $\{v_i, v_j\} \in V$

# Node Embeddings of Lexical-Semantic Graphs

- 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  $s_{ij}$ .

# Node Embeddings of Lexical-Semantic Graphs

- 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  $v_j$

# Node Embeddings of Lexical-Semantic Graphs

- 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.

# Node Embeddings of Lexical-Semantic Graphs

- The path2vec model preserves both **global** and **local** relations between nodes by minimizing

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

# Node Embeddings of Lexical-Semantic Graphs

- The path2vec model preserves both **global** and **local** relations between nodes by minimizing

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

- $s_{ij} = sim(v_i, v_j)$  is the value of a ‘gold’ similarity measure between a pair of nodes  $v_i$ ;

# Node Embeddings of Lexical-Semantic Graphs

- The path2vec model preserves both **global** and **local** relations between nodes by minimizing

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

- $s_{ij} = sim(v_i, v_j)$  is the value of a ‘gold’ similarity measure between a pair of nodes  $v_i$ ;
- $v_j$ ,  $\mathbf{v}_i$  and  $\mathbf{v}_j$  are the embeddings of the first and the second node,  $B$  is a training batch;

# Node Embeddings of Lexical-Semantic Graphs

- The path2vec model preserves both **global** and **local** relations between nodes by minimizing

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

- $s_{ij} = sim(v_i, v_j)$  is the value of a ‘gold’ similarity measure between a pair of nodes  $v_i$ ;
- $v_j$ ,  $\mathbf{v}_i$  and  $\mathbf{v}_j$  are the embeddings of the first and the second node,  $B$  is a training batch;
- $\alpha$  is a regularization coefficient.

## Node Embeddings of Lexical-Semantic Graphs

- **Leacock-Chodorow:**  $sim(w_i, w_j) = \frac{\text{depth}(\text{lcs}(w_i, w_j))}{\text{depth}(w_i) + \text{depth}(w_j)}$ ,  
lcs - lowest common subsumer in taxonomy.

# Node Embeddings of Lexical-Semantic Graphs

- **Leacock-Chodorow:**  $sim(w_i, w_j) = \frac{\text{depth}(\text{lcs}(w_i, w_j))}{\text{depth}(w_i) + \text{depth}(w_j)}$ ,  
lcs - lowest common subsumer in taxonomy.
- **Wu-Palmer:**  $sim(w_i, w_j) = -\log \text{pathlen}(w_i, w_j)$ .

# Node Embeddings of Lexical-Semantic Graphs

- **Leacock-Chodorow:**  $sim(w_i, w_j) = \frac{\text{depth}(\text{lcs}(w_i, w_j))}{\text{depth}(w_i) + \text{depth}(w_j)}$ ,

lcs - lowest common subsumer in taxonomy.

- **Wu-Palmer:**  $sim(w_i, w_j) = -\log \text{pathlen}(w_i, w_j)$ .

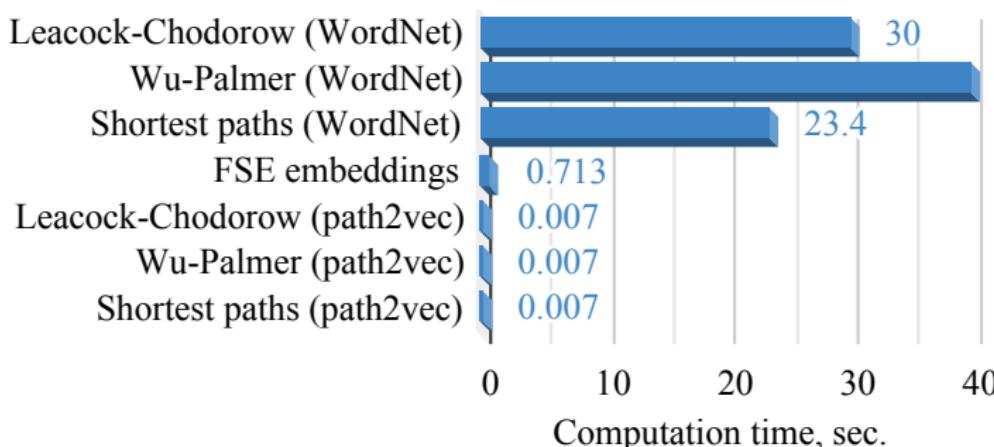


Figure: Similarity computation: graph vs vectors.

# Node Embeddings of Lexical-Semantic Graphs

## Computing Training Similarities

- Model requires computing **pairwise node similarities**  $s_{ij}$  for between pairs of nodes in the input graph  $G = (V, E)$ .

# Node Embeddings of Lexical-Semantic Graphs

## Computing Training Similarities

- Model requires computing **pairwise node similarities**  $s_{ij}$  for between pairs of nodes in the input graph  $G = (V, E)$ .
- This is **computationally expensive** for large  $|V|$ :  $O(|V|^2)$ .

# Node Embeddings of Lexical-Semantic Graphs

## Computing Training Similarities

- Model requires computing **pairwise node similarities**  $s_{ij}$  for between pairs of nodes in the input graph  $G = (V, E)$ .
- This is **computationally expensive** for large  $|V|$ :  $O(|V|^2)$ .
- It is useful to **prune the input training set** so that each node  $v_i \in V$  has only  $k \in [50; 200]$  most similar nodes.

# Node Embeddings of Lexical-Semantic Graphs

## Computing Training Similarities

- Model requires computing **pairwise node similarities**  $s_{ij}$  for between pairs of nodes in the input graph  $G = (V, E)$ .
- This is **computationally expensive** for large  $|V|$ :  $O(|V|^2)$ .
- It is 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.

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

# Lexical Substitution and Analysis of Semantic Relations

## Lexical Substitution with LMs

- 1 Build our substitute probability estimators using LMs/MLMs  $P(s|C)$ : ELMo [44], BERT [45], XLNet [46], etc.
- 2 Combine 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:

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

- 3 The final distribution is obtained by the formula:

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

# Lexical Substitution and Analysis of Semantic Relations

- Based on publication [2].

We were not able to travel in the weather , and there was no <b>phone</b> .										
GOLD	<b>telephone (5)</b>									
OOC	phone	<b>telephone</b>	phones	cellphone	fone	videophone	handset	telephones	p990i	cell-phone
XLNet	electricity	internet	<b>phone</b>	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 (4)</b>	<b>disposal (1)</b>	<b>container (1)</b>							
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

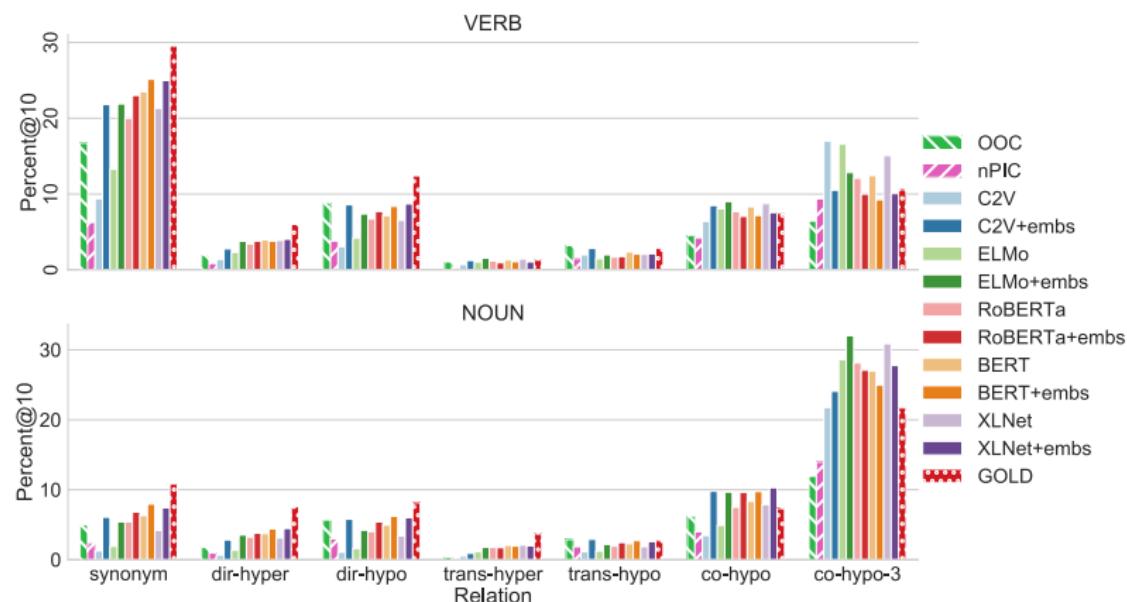
Types of semantic relations:

- synonym
- co-hyponym
- co-hyponym 3
- target
- direct hypernym
- transitive hypernym

  direct hyponym   transitive hyponym   unknown-relation   unknown-word

**Figure:** 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.

# Lexical Substitution and Analysis of Semantic Relations

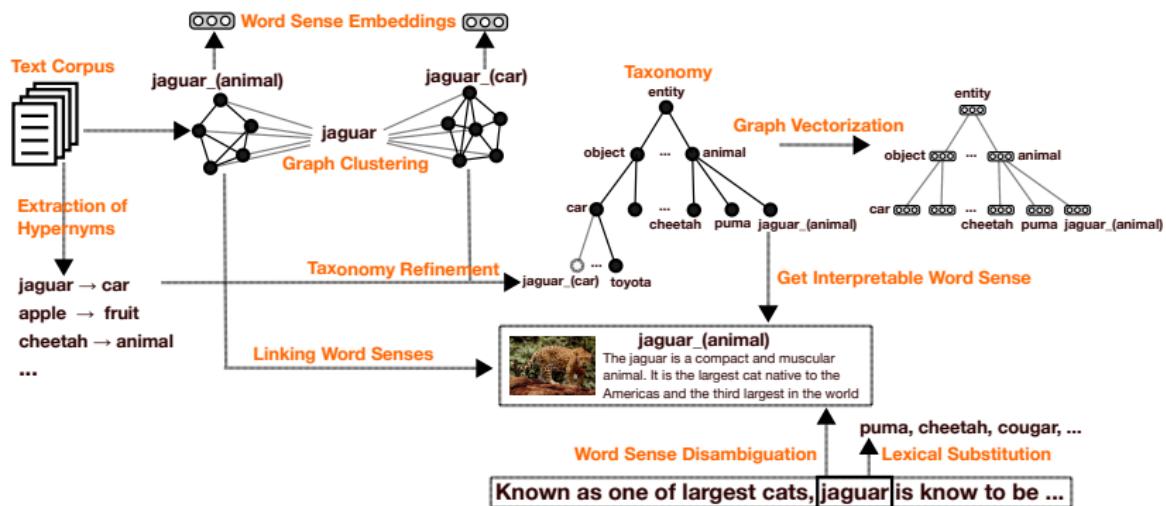


**Figure:** Proportions of substitutes related to the target by various semantic relations according to WordNet.

# Outline

- 1 Introduction
- 2 Graph Clustering for Sense and Frame Induction
- 3 Word Sense Embeddings
- 4 Unsupervised Interpretable Word Sense Disambiguation
- 5 Linking Word Sense Representations
- 6 Prediction of Hypernym Embeddings
- 7 Extracting of Hyponyms via Sense Graph Clustering
- 8 Taxonomy Enrichment using Hyperbolic Embeddings
- 9 Node Embeddings of Lexical-Semantic Graphs
- 10 Lexical Substitution and Analysis of Semantic Relations
- 11 Conclusion

## Summary



**Figure:** Overview of various methods for computational lexical semantics presented in this dissertation and their interrelations.

# Acknowledgements

- **Christian Biemann** as the main advisor and supporter.
- Key co-authors: **Dmitry Ustalov**, **Nikolay Arefyev**, **Simono Paolo Ponzetto**, **Stefano Faralli**, and **Andrey Kutuzov**.
- Student co-authors: Ramy Aly, Maria Pelevina, Shantanu Acharya, Eugen Ruppert, Boris Sheludko, Mohamman Dorgham, Oleksiy Oliynyk, Alexander Ossa, Arne Kohn, Yuri Arkhipov, Saba Anwar, and Özge Sevgili.
- Yuri Nikolaevich Philippovich and Cédrick Fairon for introducing me into CL, NLP and research in general.
- Colleagues helped and encouraged in manuscript preparation: Elena Gryazina and Elena Tutubalina.
- My family: Luidmila Borisovna, Ivan Ivanovich, Polina, Evgenii, and Konstantin.

-  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.
-  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.

-  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.
-  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.
-  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.

-  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.
-  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.

-  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.
-  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.
-  D. Ustalov, A. Panchenko, A. Kutuzov, C. Biemann, and S. P. Ponzetto, “**Unsupervised Semantic Frame Induction using Triclusustering**,” 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.*

-  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.
-  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.
-  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.

- ❑ 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.
- ❑ 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.
- ❑ 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.

- 📄 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.
- 📄 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.

-  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.
-  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.
-  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.

-  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.
-  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.

 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 (Dialogue2018). Moscow, Russia.*, pp. 192–201, RGGU, 2018.

 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 (Dialogue2018). Moscow, Russia.*, pp. 201–212, RGGU, 2018.

-  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.
-  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.
-  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.

- 📄 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.
- 📄 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.

 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.

 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.

-  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.
-  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.

- ❑ 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.
- ❑ 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.
- ❑ N. Arefyev, B. Sheludko, A. Davletov, D. Kharchev, A. Nevidomsky, and A. Panchenko, “**Neural GRANNy 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.

**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.

 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.

 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.

-  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*, (Taipei, Taiwan), pp. 39–47, Association for Computational Linguistics, Nov. 2022.
-  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.

- ❑ 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.
- ❑ 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.

-  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.
-  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.

-  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.