Word Sense Disambiguation for 158 Languages using Word Embeddings Only

Varvara Logacheva[1], Denis Teslenko[2], Artem Shelmanov[1], Steffen Remus[3], Dmitry Ustalov[4⋆], Andrey Kutuzov[5], Ekaterina Artemova[6], Chris Biemann[3], Simone Paolo Ponzetto[4], Alexander Panchenko[1]

[1]Skolkovo Institute of Science and Technology, Moscow, Russia
v.logacheva@skoltech.ru
[2]Ural Federal University, Yekaterinburg, Russia
[3]Universität Hamburg, Hamburg, Germany
[4]Universität Mannheim, Mannheim, Germany
[5]University of Oslo, Oslo, Norway
[6]Higher School of Economics, Moscow, Russia

Abstract

Disambiguation of word senses in context is easy for humans, but is a major challenge for automatic approaches. Sophisticated supervised and knowledge-based models were developed to solve this task. However, (i) the inherent Zipfian distribution of supervised training instances for a given word and/or (ii) the quality of linguistic knowledge representations motivate the development of completely unsupervised and knowledge-free approaches to word sense disambiguation (WSD). They are particularly useful for under-resourced languages which do not have any resources for building either supervised and/or knowledge-based models. In this paper, we present a method that takes as input a standard pre-trained word embedding model and induces a fully-fledged word sense inventory, which can be used for disambiguation in context. We use this method to induce a collection of sense inventories for 158 languages on the basis of the original pre-trained fastText word embeddings by Grave et al. (2018), enabling WSD in these languages. Models and system are available online.

Keywords: word sense induction, word sense disambiguation, word embeddings, sense embeddings, graph clustering

1. Introduction

There are many polysemous words in virtually any language. If not treated as such, they can hamper the performance of all semantic NLP tasks (Resnik, 2006). Therefore, the task of resolving the polysemy and choosing the most appropriate meaning of a word in context has been an important NLP task for a long time. It is usually referred to as Word Sense Disambiguation (WSD) and aims at assigning meaning to a word in context.

The majority of approaches to WSD are based on the use of knowledge bases, taxonomies, and other external manually built resources (Moro et al., 2014; Upadhyay et al., 2018). However, different senses of a polysemous word occur in very diverse contexts and can potentially be discriminated with their help. The fact that semantically related words occur in similar contexts, and diverse words do not share common contexts, is known as distributional hypothesis and underlies the technique of constructing word embeddings from unlabelled texts. The same intuition can be used to discriminate between different senses of individual words. There exist methods of training word embeddings that can detect polysemous words and assign them different vectors depending on their contexts (Athiwaratkun et al., 2018; Jain et al., 2019). Unfortunately, many widespread word embedding models, such as GloVe (Pennington et al., 2014), word2vec (Mikolov et al., 2013), fastText (Bojanowski et al., 2017), do not handle polysemous words. Words in these models are represented with single vectors, which were constructed from diverse sets of contexts corresponding to different senses. In such cases, their disambiguation needs knowledge-rich approaches.

We tackle this problem by suggesting a method of post-hoc unsupervised WSD. It does not require any external knowledge and can separate different senses of a polysemous word using only the information encoded in pre-trained word embeddings. We construct a semantic similarity graph for words and partition it into densely connected subgraphs. This partition allows for separating different senses of polysemous words. Thus, the only language resource we need is a large unlabelled text corpus used to train embeddings. This makes our method applicable to under-resourced languages. Moreover, while other methods of unsupervised WSD need to train embeddings from scratch, we perform retrofitting of sense vectors based on existing word embeddings.

We create a massively multilingual application for on-the-fly word sense disambiguation. When receiving a text, the system identifies its language and performs disambiguation of all the polysemous words in it based on pre-extracted word sense inventories. The system works for 158 languages, for which pre-trained fastText embeddings available (Grave et al., 2018).[1] The created inventories are based on these embeddings. To the best of our knowledge, our system is the only WSD system for the majority of the presented languages. Although it does not match the state of the art for resource-rich languages, it is fully unsupervised and can be used for virtually any language.

The contributions of our work are the following:

⋆ Currently at Yandex.

[1]The full list languages is available at fasttext.cc and includes English and 157 other languages for which embeddings were trained on a combination of Wikipedia and CommonCrawl texts.

- We release word sense inventories associated with fastText embeddings for 158 languages.
- We release a system that allows on-the-fly word sense disambiguation for 158 languages.
- We present `egvi` (Ego-Graph Vector Induction), a new algorithm of unsupervised word sense induction, which creates sense inventories based on pre-trained word vectors.

2. Related Work

There are two main scenarios for WSD: the supervised approach that leverages training corpora explicitly labelled for word sense, and the knowledge-based approach that derives sense representation from lexical resources, such as WordNet (Miller, 1995). In the supervised case WSD can be treated as a classification problem. Knowledge-based approaches construct sense embeddings, i.e. embeddings that separate various word senses.

SupWSD (Papandrea et al., 2017) is a state-of-the-art system for **supervised WSD**. It makes use of linear classifiers and a number of features such as POS tags, surrounding words, local collocations, word embeddings, and syntactic relations. GlossBERT model (Huang et al., 2019), which is another implementation of supervised WSD, achieves a significant improvement by leveraging gloss information. This model benefits from sentence-pair classification approach, introduced by Devlin et al. (2019) in their BERT contextualized embedding model. The input to the model consists of a context (a sentence which contains an ambiguous word) and a gloss (sense definition) from WordNet. The context-gloss pair is concatenated through a special token (`[SEP]`) and classified as positive or negative.

On the other hand, **sense embeddings** are an alternative to traditional word vector models such as word2vec, fastText or GloVe, which represent monosemous words well but fail for ambiguous words. Sense embeddings represent individual senses of polysemous words as separate vectors. They can be linked to an explicit inventory (Iacobacci et al., 2015) or induce a sense inventory from unlabelled data (Iacobacci and Navigli, 2019). LSTMEmbed (Iacobacci and Navigli, 2019) aims at learning sense embeddings linked to BabelNet (Navigli and Ponzetto, 2012), at the same time handling word ordering, and using pre-trained embeddings as an objective. Although it was tested only on English, the approach can be easily adapted to other languages present in BabelNet. However, manually labelled datasets as well as knowledge bases exist only for a small number of well-resourced languages. Thus, to disambiguate polysemous words in other languages one has to resort to fully unsupervised techniques.

The task of **Word Sense Induction** (WSI) can be seen as an unsupervised version of WSD. WSI aims at clustering word senses and does not require to map each cluster to a predefined sense. Instead of that, word sense inventories are induced automatically from the clusters, treating each cluster as a single sense of a word. WSI approaches fall into three main groups: context clustering, word ego-network clustering and synonyms (or substitute) clustering.

Context clustering approaches consist in creating vectors which characterise words' contexts and clustering these vectors. Here, the definition of context may vary from window-based context to latent topic-alike context. Afterwards, the resulting clusters are either used as senses directly (Kutuzov, 2018), or employed further to learn sense embeddings via Chinese Restaurant Process algorithm (Li and Jurafsky, 2015), AdaGram, a Bayesian extension of the Skip-Gram model (Bartunov et al., 2016), AutoSense, an extension of the LDA topic model (Amplayo et al., 2019), and other techniques.

Word ego-network clustering is applied to semantic graphs. The nodes of a semantic graph are words, and edges between them denote semantic relatedness which is usually evaluated with cosine similarity of the corresponding embeddings (Pelevina et al., 2016) or by PMI-like measures (Hope and Keller, 2013b). Word senses are induced via graph clustering algorithms, such as Chinese Whispers (Biemann, 2006) or MaxMax (Hope and Keller, 2013a). The technique suggested in our work belongs to this class of methods and is an extension of the method presented by Pelevina et al. (2016).

Synonyms and substitute clustering approaches create vectors which represent synonyms or substitutes of polysemous words. Such vectors are created using synonymy dictionaries (Ustalov et al., 2019) or context-dependent substitutes obtained from a language model (Amrami and Goldberg, 2018). Analogously to previously described techniques, word senses are induced by clustering these vectors.

.Ruby, CRuby, CoffeeScript, Ember, Faye, Garnet, Gem, Groovy, Haskell, Hazel, JRuby, Jade, Jasmine, Josie, Jruby, Lottie, Millie, Oniguruma, Opal, Python, RUBY, Ruby., Ruby-like, Rabbitfoot, RubyMotion, Rails, Rubinius, Ruby-, Ruby-based, Ruby2, RubyGem, RubyGems, RubyInstaller, RubyOnRails, RubyRuby, RubySpec, Rubygems, Rubyist, Rubyists, Rubys, Sadie, Sapphire, Sypro, Violet, jRuby, ruby, rubyists

Table 1: Top nearest neighbours of the fastText vector of the word *Ruby* are clustered according to various senses of this word: programming language, gem, first name, color, but also its spelling variations (typeset in black color).

3. Algorithm for Word Sense Induction

The majority of word vector models do not discriminate between multiple senses of individual words. However, a polysemous word can be identified via manual analysis of its nearest neighbours—they reflect different senses of the word. Table 1 shows manually sense-labelled most similar terms to the word *Ruby* according to the pre-trained fastText model (Grave et al., 2018). As it was suggested early by Widdows and Dorow (2002), the distributional properties of a word can be used to construct a graph of words that are semantically related to it, and if a word is polysemous, such graph can easily be partitioned into a number of densely connected subgraphs corresponding to different senses of this word. Our algorithm is based on the same principle.

3.1. SenseGram: A Baseline Graph-based Word Sense Induction Algorithm

SenseGram is the method proposed by Pelevina et al. (2016) that separates nearest neighbours to induce word senses and constructs sense embeddings for each sense. It starts by constructing an *ego-graph* (semantic graph centred at a particular word) of the word and its nearest neighbours. The edges between the words denote their semantic relatedness, e.g. the two nodes are joined with an edge if cosine similarity of the corresponding embeddings is higher than a pre-defined threshold. The resulting graph can be clustered into subgraphs which correspond to senses of the word.
The sense vectors are then constructed by averaging embeddings of words in each resulting cluster. In order to use these sense vectors for word sense disambiguation in text, the authors compute the probabilities of sense vectors of a word given its context or the similarity of the sense vectors to the context.

3.2. `egvi` (Ego-Graph Vector Induction): A Novel Word Sense Induction Algorithm

Induction of Sense Inventories One of the downsides of the described above algorithm is noise in the generated graph, namely, unrelated words and wrong connections. They hamper the separation of the graph. Another weak point is the imbalance in the nearest neighbour list, when a large part of it is attributed to the most frequent sense, not sufficiently representing the other senses. This can lead to construction of incorrect sense vectors.
We suggest a more advanced procedure of graph construction that uses the interpretability of vector addition and subtraction operations in word embedding space (Mikolov et al., 2013) while the previous algorithm only relies on the list of nearest neighbours in word embedding space. The key innovation of our algorithm is the use of vector subtraction to find pairs of most dissimilar graph nodes and construct the graph only from the nodes included in such "anti-edges". Thus, our algorithm is based on *graph-based* word sense induction, but it also relies on *vector-based* operations between word embeddings to perform filtering of graph nodes. Analogously to the work of Pelevina et al. (2016), we construct a semantic relatedness graph from a list of nearest neighbours, but we filter this list using the following procedure:

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

2. Compute a list Δ = {δ_1, δ_2, ..., δ_N} for each w_i in $\mathcal{N}$, where $\delta_i = w - w_i$. The vectors in δ contain the components of sense of w which are not related to the corresponding nearest neighbours from $\mathcal{N}$.

3. Compute a list $\overline{\mathcal{N}}$ = {$\overline{w_1}$, $\overline{w_2}$, ..., $\overline{w_N}$}, such that $\overline{w_i}$ is in the top nearest neighbours of δ_i in the embedding space. In other words, $\overline{w_i}$ is a word which is the most similar to the target (ego) word w and least similar to its neighbour w_i. We refer to $\overline{w_i}$ as an *anti-pair* of w_i. The set of N nearest neighbours and their anti-pairs form a set of *anti-edges* i.e. pairs of most dissimilar nodes – those which should not be connected: $\overline{E} = \{(w_1, \overline{w_1}), (w_2, \overline{w_2}), ..., (w_N, \overline{w_N})\}$.

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

4. Construct V, the set of vertices of our semantic graph $G = (V, E)$ from the list of anti-edges $\overline{E}$, with the following recurrent procedure: $V = V \cup \{w_i, \overline{w_i} : w_i \in \mathcal{N}, \overline{w_i} \in \mathcal{N}\}$, i.e. we add a word from the list of nearest neighbours *and* its anti-pair only if both of them are nearest neighbours of the original word w. We do not add w's nearest neighbours if their anti-pairs do not belong to $\mathcal{N}$. Thus, we add only words which can help discriminating between different senses of w.

5. Construct the set of edges E as follows. For each $w_i \in \mathcal{N}$ we extract a set of its K nearest neighbours $\mathcal{N}'_i = \{u_1, u_2, ..., u_K\}$ and define $E = \{(w_i, u_j) : w_i \in V, u_j \in V, u_j \in \mathcal{N}'_i, u_j \neq \overline{w_i}\}$. In other words, we remove edges between a word w_i and its nearest neighbour u_j if u_j is also its anti-pair. According to our hypothesis, w_i and $\overline{w_i}$ belong to different senses of w, so they should not be connected (i.e. we never add anti-edges into E). Therefore, we consider any connection between them as noise and remove it.

Note that N (the number of nearest neighbours for the target word w) and K (the number of nearest neighbours of w_{ci}) do not have to match. The difference between these parameters is the following. N defines how many words will be considered for the construction of ego-graph. On the other hand, K defines the degree of relatedness between words in the ego-graph — if $K = 50$, then we will connect vertices w and u with an edge only if u is in the list of 50 nearest neighbours of w. Increasing K increases the graph connectivity and leads to lower granularity of senses.
According to our hypothesis, nearest neighbours of w are grouped into clusters in the vector space, and each of the clusters corresponds to a sense of w. The described vertices selection procedure allows picking the most representative members of these clusters which are better at discriminating between the clusters. In addition to that, it helps dealing with the cases when one of the clusters is over-represented in the nearest neighbour list. In this case, many elements of such a cluster are not added to V because their anti-pairs fall outside the nearest neighbour list. This also improves the quality of clustering.
After the graph construction, the clustering is performed using the Chinese Whispers algorithm (Biemann, 2006). This is a bottom-up clustering procedure that does not require to pre-define the number of clusters, so it can correctly process polysemous words with varying numbers of senses as well as unambiguous words.
Figure 1 shows an example of the resulting pruned graph of for the word *Ruby* for $N = 50$ nearest neighbours in terms of the fastText cosine similarity. In contrast to the baseline method by (Pelevina et al., 2016) where all 50 terms are

clustered, in the method presented in this section we sparsify the graph by removing 13 nodes which were not in the set of the "anti-edges" i.e. pairs of most dissimilar terms out of these 50 neighbours. Examples of anti-edges i.e. pairs of most dissimilar terms for this graph include: (*Haskell*, *Sapphire*), (*Garnet*, *Rails*), (*Opal*, *Rubyist*), (*Hazel*, *RubyOnRails*), and (*Coffeescript*, *Opal*).

Labelling of Induced Senses We label each word cluster representing a sense to make them and the WSD results interpretable by humans. Prior systems used hypernyms to label the clusters (Ruppert et al., 2015; Panchenko et al., 2017), e.g. "animal" in the "python (animal)". However, neither hypernyms nor rules for their automatic extraction are available for all 158 languages. Therefore, we use a simpler method to select a keyword which would help to interpret each cluster. For each graph node $v \in V$ we count the number of anti-edges it belongs to: $count(v) = |\{(w_i, \overline{w_i}) : (w_i, \overline{w_i}) \in \overline{E} \wedge (v = w_i \vee v = \overline{w_i})\}|$. A graph clustering yields a partition of V into n clusters: $V = \{V_1, V_2, ..., V_n\}$. For each cluster V_i we define a *keyword* w_i^{key} as the word with the largest number of anti-edges $count(\cdot)$ among words in this cluster.

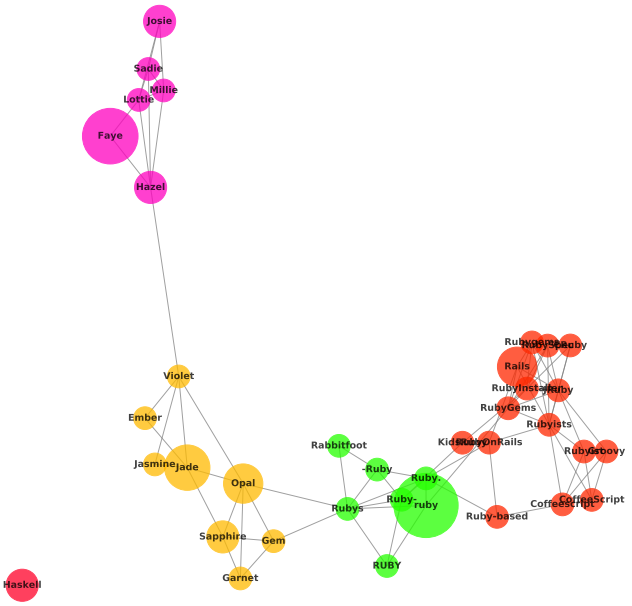

Figure 1: The graph of nearest neighbours of the word *Ruby* can be separated according several senses: programming languages, female names, gems, as well as a cluster of different spellings of the word *Ruby*.

Word Sense Disambiguation We use keywords defined above to obtain vector representations of senses. In particular, we simply use word embedding of the keyword w_i^{key} as a sense representation $\mathbf{s}_i$ of the target word w to avoid explicit computation of sense embeddings like in (Pelevina et al., 2016). Given a sentence $\{w_1, w_2, ..., w_j, w, w_{j+1}, ..., w_n\}$ represented as a matrix of word vectors, we define the context of the target word w as $\mathbf{c}_w = \frac{\sum_{j=1}^{n} w_j}{n}$. Then, we define the most appropriate sense $\hat{s}$ as the sense with the highest cosine similarity to the embedding of the word's context:

$$\hat{s} = \arg\max_{s_i} \frac{\mathbf{c}_w \cdot \mathbf{s}_i}{||\mathbf{c}_w|| \cdot ||\mathbf{s}_i||}.$$

4. System Design

We release a system for on-the-fly WSD for 158 languages. Given textual input, it identifies polysemous words and retrieves senses that are the most appropriate in the context.

4.1. Construction of Sense Inventories

To build word sense inventories (sense vectors) for 158 languages, we utilised GPU-accelerated routines for search of similar vectors implemented in Faiss library (Johnson et al., 2019). The search of nearest neighbours takes substantial time, therefore, acceleration with GPUs helps to significantly reduce the word sense construction time. To further speed up the process, we keep all intermediate results in memory, which results in substantial RAM consumption of up to 200 Gb.

The construction of word senses for all of the 158 languages takes a lot of computational resources and imposes high requirements to the hardware. For calculations, we use in parallel 10–20 nodes of the Zhores cluster (Zacharov et al., 2019) empowered with Nvidia Tesla V100 graphic cards. For each of the languages, we construct inventories based on 50, 100, and 200 neighbours for 100,000 most frequent words. The vocabulary was limited in order to make the computation time feasible. The construction of inventories for one language takes up to 10 hours, with 6.5 hours on average. Building the inventories for all languages took more than 1,000 hours of GPU-accelerated computations.

We release the constructed sense inventories for all the available languages. They contain all the necessary information for using them in the proposed WSD system or in other downstream tasks.

4.2. Word Sense Disambiguation System

The first text pre-processing step is language identification, for which we use the fastText language identification models by Bojanowski et al. (2017). Then the input is tokenised. For languages which use Latin, Cyrillic, Hebrew, or Greek scripts, we employ the Europarl tokeniser.[2] For Chinese, we use the Stanford Word Segmenter (Tseng et al., 2005). For Japanese, we use Mecab (Kudo, 2006). We tokenise Vietnamese with UETsegmenter (Nguyen and Le, 2016). All other languages are processed with the ICU tokeniser, as implemented in the PyICU project.[3] After the tokenisation, the system analyses all the input words with pre-extracted sense inventories and defines the most appropriate sense for polysemous words.

Figure 2 shows the interface of the system. It has a textual input form. The automatically identified language of text is shown above. A click on any of the words displays a prompt (shown in black) with the most appropriate sense of a word in the specified context and the confidence score. In the given example, the word *Jaguar* is correctly identified as a car brand. This system is based on the system

[2] https://www.statmt.org/europarl
[3] https://pypi.org/project/PyICU

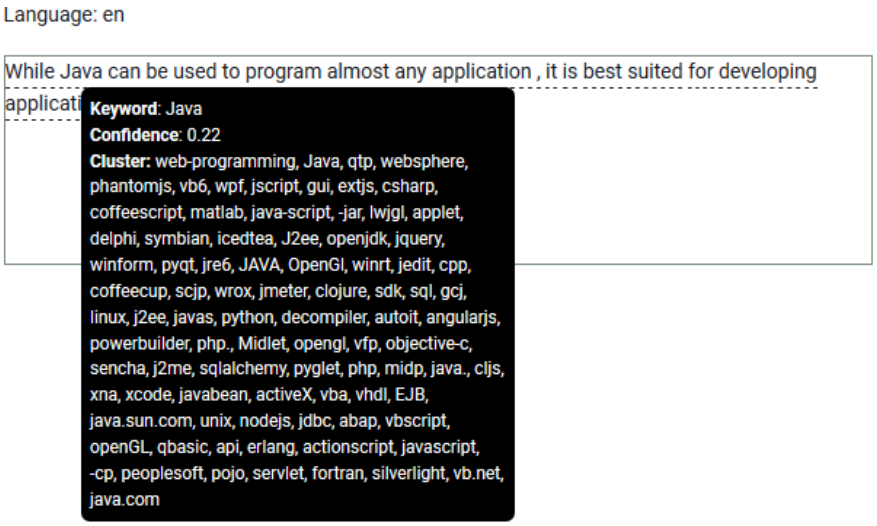

Figure 2: Interface of our WSD module with examples for the English language. Given a sentence, it identifies polysemous words and retrieves the most appropriate sense (labelled by the centroid word of a corresponding cluster).

by Ustalov et al. (2018), extending it with a back-end for multiple languages, language detection, and sense browsing capabilities.

5. Evaluation

We first evaluate our converted embedding models on multi-language lexical similarity and relatedness tasks, as a sanity check, to make sure the word sense induction process did not hurt the general performance of the embeddings. Then, we test the sense embeddings on WSD task.

5.1. Lexical Similarity and Relatedness

Experimental Setup We use the SemR-11 datasets[4] (Barzegar et al., 2018), which contain word pairs with manually assigned similarity scores from 0 (words are not related) to 10 (words are fully interchangeable) for 12 languages: English (en), Arabic (ar), German (de), Spanish (es), Farsi (fa), French (fr), Italian (it), Dutch (nl), Portuguese (pt), Russian (ru), Swedish (sv), Chinese (zh). The task is to assign relatedness scores to these pairs so that the ranking of the pairs by this score is close to the ranking defined by the oracle score. The performance is measured with Pearson correlation of the rankings. Since one word can have several different senses in our setup, we follow Remus and Biemann (2018) and define the relatedness score for a pair of words as the **maximum cosine similarity** between any of their sense vectors.

We extract the sense inventories from fastText embedding vectors. We set $N = K$ for all our experiments, i.e. the number of vertices in the graph and the maximum number of vertices' nearest neighbours match. We conduct experiments with $N = K$ set to 50, 100, and 200. For each cluster V_i we create a sense vector s_i by averaging vectors that belong to this cluster. We rely on the methodology of (Remus and Biemann, 2018) shifting the generated sense vector to the direction of the original word vector: $s_i = \lambda\, w + (1 - \lambda)\, \frac{1}{n} \sum_{u \in V_i} cos(w, u) \cdot u$, where, $\lambda \in [0, 1]$, w is the embedding of the original word, $cos(w, u)$ is the cosine similarity between w and u, and $n = |V_i|$. By introducing the linear combination of w and $u \in V_i$ we enforce the similarity of sense vectors to the original word important for this task. In addition to that, we weight u by their similarity to the original word, so that more similar neighbours contribute more to the sense vector. The shifting parameter λ is set to 0.5, following Remus and Biemann (2018).

A fastText model is able to generate a vector for each word even if it is not represented in the vocabulary, due to the use of subword information. However, our system cannot assemble sense vectors for out-of-vocabulary words, for such words it returns their original fastText vector. Still, the coverage of the benchmark datasets by our vocabulary is at least 85% and approaches 100% for some languages, so we do not have to resort to this back-off strategy very often.

We use the original fastText vectors as a **baseline**. In this case, we compute the relatedness scores of the two words as a cosine similarity of their vectors.

Discussion of Results We compute the relatedness scores for all benchmark datasets using our sense vectors and compare them to cosine similarity scores of original fastText vectors. The results vary for different languages. Figure 3 shows the change in Pearson correlation score when switching from the baseline fastText embeddings to our sense vectors. The new vectors significantly improve the relatedness detection for German, Farsi, Russian, and Chinese, whereas for Italian, Dutch, and Swedish the score slightly falls behind the baseline. For other languages, the performance of sense vectors is on par with regular fastText.

5.2. Word Sense Disambiguation

The purpose of our sense vectors is disambiguation of polysemous words. Therefore, we test the inventories constructed with `egvi` on the Task 13 of SemEval-2013 — Word Sense Induction (Jurgens and Klapaftis, 2013). The task is to identify the different senses of a target word in context in a fully unsupervised manner.

[4] https://github.com/Lambda-3/Gold-Standards/tree/master/SemR-11

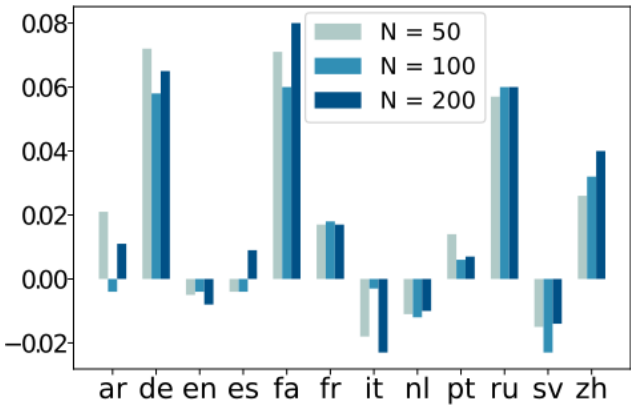

Figure 3: Absolute improvement of Pearson correlation scores of our embeddings compared to fastText. This is the averaged difference of the scores for all word similarity benchmarks.

Experimental Setup The dataset consists of a set of polysemous words: 20 nouns, 20 verbs, and 10 adjectives and specifies 20 to 100 contexts per word, with the total of 4,664 contexts, drawn from the Open American National Corpus. Given a set of contexts of a polysemous word, the participants of the competition had to divide them into clusters by sense of the word. The contexts are manually labelled with WordNet senses of the target words, the gold standard clustering is generated from this labelling.

The task allows two setups: *graded* WSI where participants can submit multiple senses per word and provide the probability of each sense in a particular context, and *non-graded* WSI where a model determines a single sense for a word in context. In our experiments we performed *non-graded* WSI. We considered the most suitable sense as the one with the highest cosine similarity with embeddings of the context, as described in Section 3.2.

The performance of WSI models is measured with three metrics that require mapping of sense inventories (Jaccard Index, Kendall's τ, and WNDCG) and two cluster comparison metrics (Fuzzy NMI and Fuzzy B-Cubed).

Discussion of Results We compare our model with the models that participated in the task, the baseline ego-graph clustering model by Pelevina et al. (2016), and AdaGram (Bartunov et al., 2016), a method that learns sense embeddings based on a Bayesian extension of the Skip-gram model. Besides that, we provide the scores of the simple **baselines** originally used in the task: assigning one sense to all words, assigning the most frequent sense to all words, and considering each context as expressing a different sense. The evaluation of our model was performed using the open source `context-eval` tool.[5]

Table 2 shows the performance of these models on the SemEval dataset. Due to space constraints, we only report the scores of the best-performing SemEval participants, please refer to Jurgens and Klapaftis (2013) for the full results. The performance of AdaGram and SenseGram models is reported according to Pelevina et al. (2016).

The table shows that the performance of `egvi` is similar to state-of-the-art word sense disambiguation and word sense induction models. In particular, we can see that it outperforms SenseGram on the majority of metrics. We should note that this comparison is not fully rigorous, because SenseGram induces sense inventories from word2vec as opposed to fastText vectors used in our work.

[5] `https://github.com/uhh-lt/context-eval`

5.3. Analysis

In order to see how the separation of word contexts that we perform corresponds to actual senses of polysemous words, we visualise ego-graphs produced by our method. Figure 1 shows the nearest neighbours clustering for the word *Ruby*, which divides the graph into five senses: *Ruby-related programming tools*, e.g. RubyOnRails (orange cluster), *female names*, e.g. Josie (magenta cluster), *gems*, e.g. Sapphire (yellow cluster), *programming languages in general*, e.g. Haskell (red cluster). Besides, this is typical for fastText embeddings featuring sub-string similarity, one can observe a cluster of different spelling of the word *Ruby* in green.

Analogously, the word *python* (see Figure 4) is divided into the senses of *animals*, e.g. crocodile (yellow cluster), *programming languages*, e.g. perl5 (magenta cluster), and *conference*, e.g. pycon (red cluster).

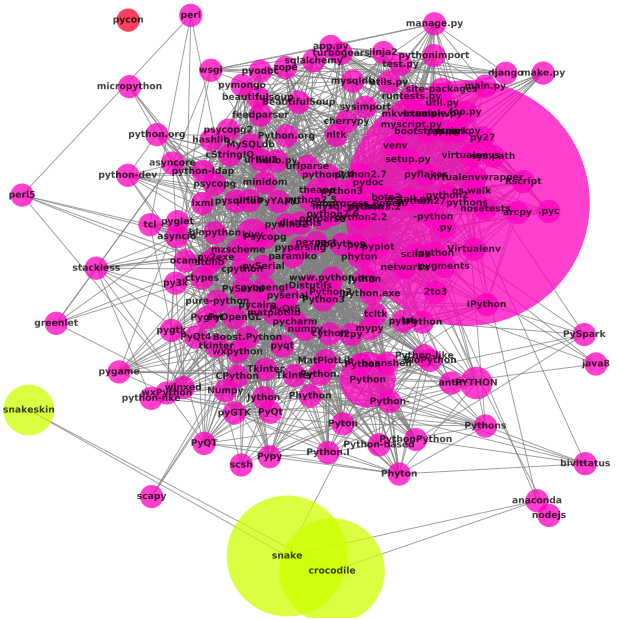

Figure 4: Ego-graph for a polysemous word *python* which is clustered into senses *snake* (yellow), *programming language* (magenta), and *conference* (red). Node size denotes word importance with the largest node in the cluster being used as a keyword to interpret an induced word sense.

In addition, we show a qualitative analysis of senses of *mouse* and *apple*. Table 4 shows nearest neighbours of the original words separated into clusters (labels for clusters were assigned manually). These inventories demonstrate clear separation of different senses, although it can be too fine-grained. For example, the first and the second cluster for *mouse* both refer to computer mouse, but the first one addresses the different types of computer mice, and the second one is used in the context of mouse actions. Similarly, we see that *iphone* and *macbook* are sep-

Model	Supervised Evaluation			Clustering Evaluation	
	Jacc. Ind.	**Kendall's τ**	**WNDCG**	**F.NMI**	**F.B-Cubed**
Baselines					
One sense for all	0.192	0.609	0.288	0.000	0.631
One sense per instance	0.000	0.000	0.000	0.071	0.000
Most Frequent Sense	0.455	0.465	0.339	–	–
SemEval-2013 participants					
AI-KU (base)	0.197	0.620	0.387	0.065	0.390
AI-KU (remove5-add1000)	0.244	0.642	0.332	0.039	0.451
Unimelb (50k)	0.213	0.620	0.371	0.060	0.483
Sense embeddings					
AdaGram, α = 0.05, 100 dim. vectors	0.274	0.644	0.318	0.058	0.470
SenseGram (word2vec)	0.197	0.615	0.291	0.011	0.615
egvi (fastText, K=200)	0.229	0.625	0.300	0.035	0.541

Table 2: WSD performance on the SemEval-2013 Task 13 dataset for the English language.

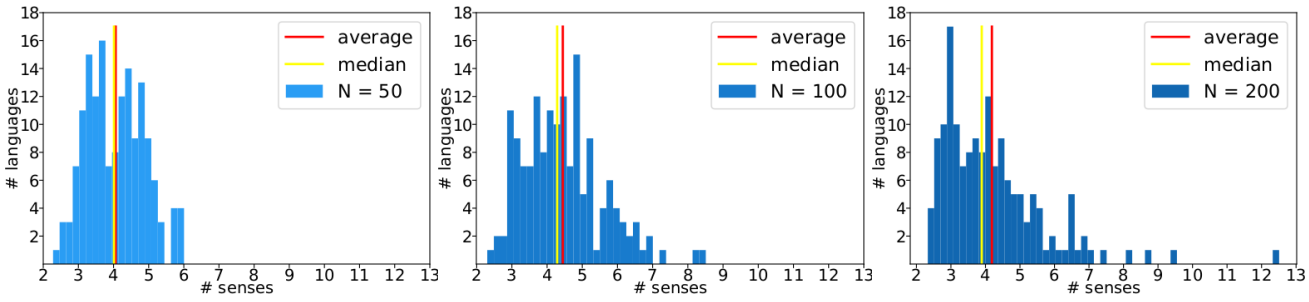

Figure 5: Distribution of the number of senses per word in the generated inventories for all 158 languages for the number of neighbours set to: $N \in \{50, 100, 200\}$, $K \in \{50, 100, 200\}$ with $N = K$.

arated into two clusters. Interestingly, fastText handles typos, code-switching, and emojis by correctly associating all non-standard variants to the word they refer, and our method is able to cluster them appropriately.

Both inventories were produced with $K = 200$, which ensures stronger connectivity of graph. However, we see that this setting still produces too many clusters. We computed the average numbers of clusters produced by our model with $K = 200$ for words from the word relatedness datasets and compared these numbers with the number of senses in WordNet for English and RuWordNet (Loukachevitch and Dobrov, 2014) for Russian (see Table 3). We can see that the number of senses extracted by our method is consistently higher than the real number of senses.

We also compute the average number of senses per word for all the languages and different values of K (see Figure 5). The average across languages does not change much as we increase K. However, for larger K the average exceed the median value, indicating that more languages have lower number of senses per word. At the same time, while at smaller K the maximum average number of senses per word does not exceed 6, larger values of K produce outliers, e.g. English with 12.5 senses.

Notably, there are no languages with an average number of senses less than 2, while numbers on English and Russian WordNets are considerably lower. This confirms that our method systematically over-generates senses. The presence of outliers shows that this effect cannot be eliminated by further increasing K, because the i-th nearest neighbour of a word for $i > 200$ can be only remotely related to this word, even if the word is rare. Thus, our sense clustering algorithm needs a method of merging spurious senses.

	mc	rg	simlex	ws353	total
	English				
inventory	9.8	9.8	12.6	11.3	12.5
WordNet	3.6	3.7	6.5	5.5	1.23 (nouns) 2.16 (verbs)
	Russian				
inventory	1.8	2.0	–	2.2	2.97
RuWordNet	1.4	1.4	–	1.8	1.12 (nouns) 1.33 (verbs)

Table 3: Average number of senses for words from SemR-11 dataset in our inventory and in WordNet for English and ruWordNet for Russian. The rightmost column gives the average number of senses in the inventories and WordNets.

6. Conclusions and Future Work

We present `egvi`, a new algorithm for word sense induction based on graph clustering that is fully unsupervised and relies on graph operations between word vectors. We

Label	Nearest neighbours
	MOUSE
computer mouse types	touch-pad, logitec, scrollwheel, **mouses**, mouse.It, mouse.The, joystick, trackpads, nano-receiver, 800DPI, nony, track-pad, M325, keyboard-controlled, Intellipoint, MouseI, intellimouse, Swiftpoint, Evoluent, 800dpi, moused, game-pad, steelseries, ErgoMotion, IntelliMouse, <...>
computer mouse actions	Ctrl-Alt, right-mouse, cursor, left-clicks, spacebar, rnUse, mouseclick, **click**, mousepointer, keystroke, cusor, mousewheel, MouseMove, mousebutton, leftclick, click-dragging, mouse-button, cursor., arrow-key, double-clicks, mouse-down, ungrab, mouseX, arrow-keys, right-button, <...>
rodent	Rodent, rodent, mousehole, rats, **mice**, mice-, hamster, SOD1G93A, meeses, mice.The, PDAPP, hedgehog, Maukie, rTg4510, mousey, meeces, rodents, cat, White-footed, rat, Mice, <...>
keyboard	keyborad, keybard, keybord, keyboardThe, keyboad, keyboar, **Keyboard**, keboard, keyboardI, keyb, keyboard.This, keybaord, keyboard
medical	SENCAR, mTERT, **mouse-specific**
Latin	Apodemus, **Micormys**
Latin	**Akodon**
	APPLE
iphone	mobileme, idevice, carplay, iphones, icloud, iwatch, ios5, ipod, **iphone**, android, ifans, iphone.I, iphone4, iphone5s, idevices, ipad, ios, ipad., iphone5, iphone., ios7
fruit	**apples**, apple-producing, Honeycrisp, apple-y, Macouns, apple-growing, pear, apple-pear, Gravensteins, apple-like, Apples, Honeycrisps, apple-related, Borkh, Braeburns, Starkrimson, Apples-, SweeTango, Elstar
macbook	macbook, macbookpro, macbookair, imac, ibooks, **tuaw**, osx, macintosh, imacs, apple.com, applestore, Tagsapple, stevejobs, applecare
fruit, typos pinklady	blackerry, blackberry, blueberry, aplle, cidar, **apple.The**, apple.I, aple, appple, calvados, pie.It,
tokenisation issues, typos	Apple.This, AMApple, it.Apple, too.Apple, AppleApple, **up.Apple**, AppleA, Aplle, Apple.Apple
Apple criticism	anti-apple, Aple, Crapple, isheep, iDiots, crapple, Appple, iCrap, **non-apple**
Bible	**Adam**
cooking	**caramel-dipped**
iphone	**earpod**
Russian	яблоко [Russian: “apple”]
emoji	**[apple emoji]**

Table 4: Clustering of senses for words *mouse* and *apple* produced by our method. Cluster labels in this table were assigned manually for illustrative purposes. For on-the-fly disambiguation we use centroid words in clusters as sense labels (shown here **in bold**).

apply this algorithm to a large collection of pre-trained fastText word embeddings, releasing sense inventories for 158 languages.[6] These inventories contain all the necessary information for constructing sense vectors and using them in downstream tasks. The sense vectors for polysemous words can be directly retrofitted with the pre-trained word embeddings and do not need any external resources. As one application of these multilingual sense inventories, we present a multilingual word sense disambiguation system that performs unsupervised and knowledge-free WSD for 158 languages without the use of any dictionary or sense-labelled corpus.

The evaluation of quality of the produced sense inventories is performed on multilingual word similarity benchmarks, showing that our sense vectors improve the scores compared to non-disambiguated word embeddings. Therefore, our system in its present state can improve WSD and downstream tasks for languages where knowledge bases, taxonomies, and annotated corpora are not available and supervised WSD models cannot be trained.

A promising direction for future work is combining distributional information from the induced sense inventories with lexical knowledge bases to improve WSD performance. Besides, we encourage the use of the produced word sense inventories in other downstream tasks.

[6]Links to the produced datasets, online demo, and source codes are available at: `http://uhh-lt.github.io/158`.

7. Acknowledgements

We acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) foundation under the “JOIN-T 2” and “ACQuA” projects. Ekaterina Artemova was supported by the framework of the HSE University Basic Research Program and Russian Academic Excellence Project “5-100”.

8. Bibliographical References

Amplayo, R. K., Hwang, S.-w., and Song, M. (2019). AutoSense Model for Word Sense Induction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6212–6219, Honolulu, HI, USA. Association for the Advancement of Artificial Intelligence (AAAI).

Amrami, A. and Goldberg, Y. (2018). Word Sense Induction with Neural biLM and Symmetric Patterns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2018, pages 4860–4867, Brussels, Belgium. Association for Computational Linguistics.

Athiwaratkun, B., Wilson, A., and Anandkumar, A. (2018). Probabilistic FastText for Multi-Sense Word Embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2018, pages 1–11, Melbourne, VIC, Australia. Association for Computational Linguistics.

Bartunov, S., Kondrashkin, D., Osokin, A., and Vetrov, D. P. (2016). Breaking Sticks and Ambiguities with Adaptive Skip-gram. *Journal of Machine Learning Research*, 51:130–138.

Barzegar, S., Davis, B., Zarrouk, M., Handschuh, S., and Freitas, A. (2018). SemR-11: A Multi-Lingual Gold-Standard for Semantic Similarity and Relatedness for Eleven Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 3912–3916, Miyazaki, Japan. European Language Resources Association (ELRA).

Biemann, C. (2006). Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 73–80, New York, NY, USA. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). 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)*, NAACL-HLT 2019, pages 4171–4186, Minneapolis, MN, USA. Association for Computational Linguistics.

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 3483–3487, Miyazaki, Japan. European Language Resources Association (ELRA).

Hope, D. and Keller, B. (2013a). MaxMax: A Graph-Based Soft Clustering Algorithm Applied to Word Sense Induction. In *Computational Linguistics and Intelligent Text Processing, 14th International Conference, CICLing 2013, Samos, Greece, March 24-30, 2013, Proceedings, Part I*, volume 7816 of *Lecture Notes in Computer Science*, pages 368–381. Springer Berlin Heidelberg, Berlin and Heidelberg, Germany.

Hope, D. and Keller, B. (2013b). UoS: A Graph-Based System for Graded Word Sense Induction. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 689–694, Atlanta, GA, USA. Association for Computational Linguistics.

Huang, L., Sun, C., Qiu, X., and Huang, X. (2019). GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514, Hong Kong. Association for Computational Linguistics.

Iacobacci, I. and Navigli, R. (2019). LSTMEmbed: Learning Word and Sense Representations from a Large Semantically Annotated Corpus with Long Short-Term Memories. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL 2019, pages 1685–1695, Florence, Italy. Association for Computational Linguistics.

Iacobacci, I., Pilehvar, M. T., and Navigli, R. (2015). Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, ACL-IJCNLP 2015, pages 95–105, Beijing, China. Association for Computational Linguistics.

Jain, S., Bodapati, S. B., Nallapati, R., and Anandkumar, A. (2019). Multi Sense Embeddings from Topic Models. In *Proceedings of the 3rd International Conference on Natural Language and Speech Processing*, ICNLSP 2019, pages 34–41, Trento, Italy. Association for Computational Linguistics.

Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*.

Jurgens, D. and Klapaftis, I. (2013). SemEval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 290–299, Atlanta, GA, USA, June. Association for Computational Linguistics.

Kudo, T. (2006). MeCab: Yet Another Part-of-Speech and Morphological Analyzer. `http://mecab.sourceforge.jp/`.

Kutuzov, A. (2018). Russian word sense induction by clustering averaged word embeddings. In *Komp'yuternaya Lingvistika i Intellektual'nye Tekhnologii: Dialog conference*, pages 391–403, Moscow, Russia.

Li, J. and Jurafsky, D. (2015). Do Multi-Sense Embed-

dings Improve Natural Language Understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2015, pages 1722–1732, Lisbon, Portugal. Association for Computational Linguistics.

Loukachevitch, N. and Dobrov, B. (2014). RuThes Linguistic Ontology vs. Russian Wordnets. In *Proceedings of the Seventh Global Wordnet Conference*, GWC 2014, pages 154–162, Tartu, Estonia. University of Tartu Press.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, NIPS 2013, pages 3111–3119. Curran Associates, Inc., Harrahs and Harveys, NV, USA.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Moro, A., Cecconi, F., and Navigli, R. (2014). Multilingual Word Sense Disambiguation and Entity Linking for Everybody. In *Proceedings of the ISWC 2014 Posters & Demonstrations Track*, pages 25–28, Riva del Garda, Italy.

Navigli, R. and Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Nguyen, T.-P. and Le, A.-C. (2016). A hybrid approach to vietnamese word segmentation. In *2016 IEEE RIVF International Conference on Computing Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, pages 114–119, Hanoi, Vietnam. IEEE.

Panchenko, A., Marten, F., Ruppert, E., Faralli, S., Ustalov, D., Ponzetto, S. P., and Biemann, C. (2017). Unsupervised, knowledge-free, and interpretable word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 91–96, Copenhagen, Denmark, September. Association for Computational Linguistics.

Papandrea, S., Raganato, A., and Delli Bovi, C. (2017). SupWSD: A Flexible Toolkit for Supervised Word Sense Disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, EMNLP 2017, pages 103–108, Copenhagen, Denmark. Association for Computational Linguistics.

Pelevina, M., Arefiev, N., Biemann, C., and Panchenko, A. (2016). Making Sense of Word Embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, RepL4NLP, pages 174–183, Berlin, Germany. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2014, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Remus, S. and Biemann, C. (2018). Retrofitting Word Representations for Unsupervised Sense Aware Word Similarities. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 1035–1041, Miyazaki, Japan. European Language Resources Association (ELRA).

Resnik, P. (2006). WSD in NLP Applications. In *Word Sense Disambiguation*, pages 299–337. Springer.

Ruppert, E., Kaufmann, M., Riedl, M., and Biemann, C. (2015). JoBimViz: A web-based visualization for graph-based distributional semantic models. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 103–108, Beijing, China, July. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.

Tseng, H., Chang, P., Andrew, G., Jurafsky, D., and Manning, C. (2005). A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168–171.

Upadhyay, S., Gupta, N., and Roth, D. (2018). Joint Multilingual Supervision for Cross-lingual Entity Linking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2018, pages 2486–2495, Brussels, Belgium. Association for Computational Linguistics.

Ustalov, D., Teslenko, D., Panchenko, A., Chersnoskutov, M., Biemann, C., and Ponzetto, S. P. (2018). An Unsupervised Word Sense Disambiguation System for Under-Resourced Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 1018–1022, Miyazaki, Japan. European Language Resources Association (ELRA).

Ustalov, D., Panchenko, A., Biemann, C., and Ponzetto, S. P. (2019). WATSET: Local-Global Graph Clustering with Applications in Sense and Frame Induction. *Computational Linguistics*, 45(3):423–479.

Widdows, D. and Dorow, B. (2002). A Graph Model for Unsupervised Lexical Acquisition. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Taipei, Taiwan. Association for Computational Linguistics.

Zacharov, I., Arslanov, R., Gunin, M., Stefonishin, D., Bykov, A., Pavlov, S., Panarin, O., Maliutin, A., Rykovanov, S., and Fedorov, M. (2019). "Zhores" — Petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology. *Open Engineering*, 9(1):512–520.