Making Sense of Word Embeddings

Maria Pelevina[1], **Nikolay Arefyev**[2], **Chris Biemann**[1] **and Alexander Panchenko**[1]

[1]Technische Universität Darmstadt, LT Group, Computer Science Department, Germany
[2]Moscow State University, Faculty of Computational Mathematics and Cybernetics, Russia
panchenko@lt.informatik.tu-darmstadt.de

Abstract

We present a simple yet effective approach for learning word sense embeddings. In contrast to existing techniques, which either directly learn sense representations from corpora or rely on sense inventories from lexical resources, our approach can induce a sense inventory from existing word embeddings via clustering of ego-networks of related words. An integrated WSD mechanism enables labeling of words in context with learned sense vectors, which gives rise to downstream applications. Experiments show that the performance of our method is comparable to state-of-the-art unsupervised WSD systems.

1 Introduction

Term representations in the form of dense vectors are useful for many natural language processing applications. First of all, they enable the computation of semantically related words. Besides, they can be used to represent other linguistic units, such as phrases and short texts, reducing the inherent sparsity of traditional vector-space representations (Salton et al., 1975).

One limitation of most word vector models, including sparse (Baroni and Lenci, 2010) and dense (Mikolov et al., 2013) representations, is that they conflate all senses of a word into a single vector. Several architectures for learning multi-prototype embeddings were proposed that try to address this shortcoming (Huang et al., 2012; Tian et al., 2014; Neelakantan et al., 2014; Nieto Piña and Johansson, 2015; Bartunov et al., 2016). Li and Jurafsky (2015) provide indications that such sense vectors improve the performance of text processing applications, such as part-of-speech tagging and semantic relation identification.

The contribution of this paper is a novel method for learning word sense vectors. In contrast to previously proposed methods, our approach relies on existing single-prototype word embeddings, transforming them to sense vectors via ego-network clustering. An ego network consists of a single node (ego) together with the nodes they are connected to (alters) and all the edges among those alters. Our method is fitted with a word sense disambiguation (WSD) mechanism, and thus words in context can be mapped to these sense representations. An advantage of our method is that one can use existing word embeddings and/or existing word sense inventories to build sense embeddings. Experiments show that our approach performs comparably to state-of-the-art unsupervised WSD systems.

2 Related Work

Our method learns multi-prototype word embeddings and applies them to WSD. Below we briefly review both strains of research.

2.1 Multi-Prototype Word Vector Spaces

In his pioneering work, Schütze (1998) induced sparse sense vectors by clustering context vectors using the EM algorithm. This approach is fitted with a similarity-based WSD mechanism. Later, Reisinger and Mooney (2010) presented a multi-prototype vector space. Sparse TF-IDF vectors are clustered using a parametric method fixing the same number of senses for all words. Sense vectors are centroids of the clusters.

While most dense word vector models represent a word with a single vector and thus conflate senses (Mikolov et al., 2013; Pennington et al., 2014), there are several approaches that produce word sense embeddings. Huang et al. (2012) learn

Proceedings of the 1st Workshop on Representation Learning for NLP, pages 174–183,
Berlin, Germany, August 11th, 2016.

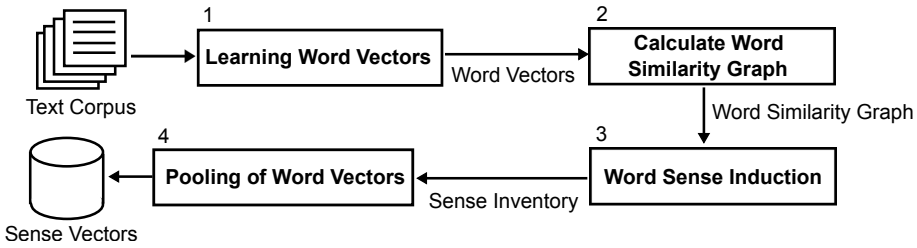

Figure 1: Schema of the word sense embeddings learning method.

dense vector spaces with neural networks. First, contexts are represented with word embeddings and clustered. Second, word occurrences are relabeled in the corpus according to the cluster they belong to. Finally, embeddings are re-trained on this sense-labeled terms. Tian et al. (2014) introduced a probabilistic extension of the Skip-gram model (Mikolov et al., 2013) that learns multiple sense-aware prototypes weighted by their prior probability. These models use parametric clustering algorithms that produce a fixed number of senses per word.

Neelakantan et al. (2014) proposed a multi-sense extension of the Skip-gram model that was the first one to learn the number of senses by itself. During training, a new sense vector is allocated if the current context's similarity to existing senses is below some threshold. Li and Jurafsky (2015) use a similar idea by integrating the Chinese Restaurant Process into the Skip-gram model. All mentioned above sense embeddings were evaluated on the contextual word similarity task, each one improving upon previous models.

Nieto and Johansson (2015) presented another multi-prototype modification of the Skip-gram model. Their approach outperforms that of Neelakantan et al. (2014), but requires as an input the number of senses for each word.

Li and Jurafsky (2015) show that sense embeddings can significantly improve the performance of part-of-speech tagging, semantic relation identification and semantic relatedness tasks, but yield no improvement for named entity recognition and sentiment analysis.

Bartunov et al. (2016) introduced AdaGram, a non-parametric method for learning sense embeddings based on a Bayesian extension of the Skip-gram model. The granularity of learned sense embeddings is controlled by the parameter α. Comparisons of their approach to (Neelakantan et al., 2014) on three SemEval word sense induction and disambiguation datasets show the advantage of their method. For this reason, we use AdaGram as a representative of the state-of-the-art methods in our experiments.

Several approaches rely on a knowledge base (KB) to provide sense information. Bordes et al. (2011) propose a general method to represent entities of any KB as a dense vector. Such representation helps to integrate KBs into NLP systems. Another approach that uses sense inventories of knowledge bases was presented by Camacho-Collados et al. (2015). Rothe and Schütze (2015) combined word embeddings on the basis of WordNet synsets to obtain sense embeddings. The approach is evaluated on lexical sample tasks by adding synset embeddings as features to an existing WSD system. They used a weighted pooling similar to the one we use, but their method is not able to find new senses in a corpus. Finally, Nieto Piña and Johansson (2016) used random walks on the Swedish Wordnet to generate training data for the Skip-gram model.

2.2 Word Sense Disambiguation (WSD)

Many different designs of WSD systems were proposed, see (Agirre and Edmonds, 2007; Navigli, 2009). Supervised approaches use an explicitly sense-labeled training corpus to construct a model, usually building one model per target word (Lee and Ng, 2002; Klein et al., 2002). These approaches demonstrate top performance in competitions, but require considerable amounts of sense-labeled examples.

Knowledge-based approaches do not learn a model per target, but rather derive sense representation from information available in a lexical resource, such as WordNet. Examples of such system include (Lesk, 1986; Banerjee and Pedersen, 2002; Pedersen et al., 2005; Moro et al., 2014)

Unsupervised WSD approaches rely neither on hand-annotated sense-labeled corpora, nor on

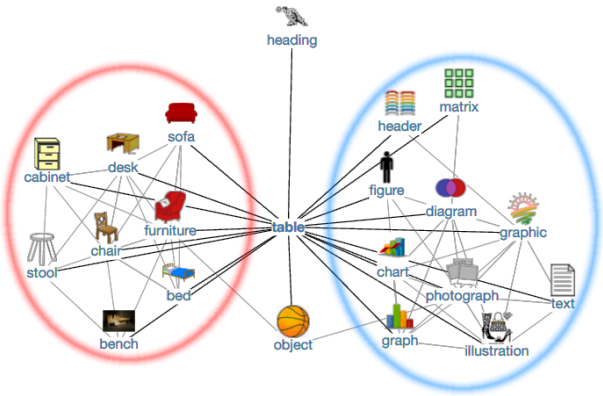

Figure 2: Visualization of the ego-network of "table" with furniture and data sense clusters. Note that the target "table" is excluded from clustering.

handcrafted lexical resources. Instead, they automatically induce a sense inventory from raw corpora. Such unsupervised sense induction methods fall into two categories: context clustering, such as (Pedersen and Bruce, 1997; Schütze, 1998; Reisinger and Mooney, 2010; Neelakantan et al., 2014; Bartunov et al., 2016) and word (ego-network) clustering, such as (Lin, 1998; Pantel and Lin, 2002; Widdows and Dorow, 2002; Biemann, 2006; Hope and Keller, 2013). Unsupervised methods use disambiguation clues from the induced sense inventory for word disambiguation. Usually, the WSD procedure is determined by the design of sense inventory. It might be the highest overlap between the instance's context words and the words of the sense cluster, as in (Hope and Keller, 2013) or the smallest distance between context words and sense hubs in graph sense representation, as in (Véronis, 2004).

3 Learning Word Sense Embeddings

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

Our method can use existing word embeddings, sense inventories and word similarity graphs. To demonstrate such use-cases and to study the performance of the method in different settings, as variants of the complete pipeline presented in Figure 1, we experiment with two additional setups. First, we use an alternative approach to compute the word similarity graph, which relies on dependency features and is expected to provide more accurate similarities (therefore, the stage (2) is changed). Second, we use a sense inventory constructed using crowdsourcing (thus, stages (2) and (3) are skipped). Below we describe each of the stages of our method in detail.

3.1 Learning Word Vectors

To learn word vectors, we use the *word2vec* toolkit (Mikolov et al., 2013), namely we train CBOW word embeddings with 100 or 300 dimensions, context window size of 3 and minimum word frequency of 5. We selected these parameters according to prior evaluations, e.g. (Baroni et al., 2014), and tested them on the development dataset (see Section 5.1). Initial experiments showed that this configuration is superior to others, e.g. the Skip-gram model, with respect to WSD performance.

For training, we modified the standard implementation of *word2vec*[1] so that it also saves context vectors needed for one of our WSD approaches. For experiments, we use two commonly used corpora for training distributional models: Wikipedia[2] and ukWaC (Ferraresi et al., 2008).

3.2 Calculating Word Similarity Graph

At this step, we build a graph of word similarities, such as (table, desk, 0.78). For each word we retrieve its 200 nearest neighbours. This number is motivated by prior studies (Biemann and Riedl, 2013; Panchenko, 2013): as observed, only few words have more strongly semantically related words. This graph is computed either based on word embeddings learned during the previous step or using semantic similarities provided by the *JoBimText* framework (Biemann and Riedl, 2013).

Similarities using word2vec (w2v). In this case, nearest neighbours of a term are terms with the highest cosine similarity of their respective vectors. For scalability reasons, we perform similarity computations via block matrix multiplications, using blocks of 1000 vectors.

Similarities using JoBimText (JBT). In this unsupervised approach, every word is represented

[1] https://code.google.com/p/word2vec

[2] We used an English Wikipedia dump of October 2015: http://panchenko.me/data/joint/corpora/en59g/wikipedia.txt.gz

as a bag of sparse dependency-based features extracted using the Malt parser and collapsed using an approach similar to (Ruppert et al., 2015). Features are normalized using the LMI score (Church and Hanks, 1990) and further pruned down according to the recommended defaults: we keep 1000 features per word and 1000 words per feature. Similarity of two words is equal to the number of common features.

Multiple alternatives exist for computation of semantic relatedness (Zhang et al., 2013). JBT has two advantages in our case: (1) accurate estimation of word similarities based on dependency features; (2) efficient computation of nearest neighbours for all words in a corpus. Besides, we observed that nearest neighbours of word embeddings often tend to belong to the dominant sense, even if minor senses have significant support in the training corpus. We wanted to test if the same problem remains for a principally different method for similarity computation.

Algorithm 1: Word sense induction.

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

3.3 Word Sense Induction

We induce a sense inventory using a method similarly to (Pantel and Lin, 2002) and (Biemann, 2006). A word sense is represented by a word cluster. For instance the cluster "chair, bed, bench, stool, sofa, desk, cabinet" can represent the sense "table (furniture)". To induce senses, first we construct an ego-network G of a word t and then perform graph clustering of this network. The identified clusters are interpreted as senses (see Table 2). Words referring to the same sense tend to be tightly connected, while having fewer connections to words referring to different senses.

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-left0, 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 1: 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.

The sense induction presented in Algorithm 1 processes one word t of the word similarity graph T per iteration. First, we retrieve nodes V of the ego-network G: these are the N most similar words of t according to T. The target word t itself is not part of the ego-network. Second, we connect the nodes in G to their n most similar words from T. Finally, the ego-network is clustered with the Chinese Whispers algorithm (Biemann, 2006). This method is parameter free, thus we make no assumptions about the number of word senses.

The sense induction algorithm has three meta-parameters: the ego-network size (N) of the target ego word t; the ego-network connectivity (n) is the maximum number of connections the neighbour v is allowed to have within the ego-network; the minimum size of the cluster k. The n parameter regulates the granularity of the inventory. In our experiments, we set the N to 200, n to 50, 100 or 200 and k to 5 or 15 to obtain different granulates, cf. (Biemann, 2010).

Each word in a sense cluster has a weight which is equal to the similarity score between this word and the ambiguous word t.

3.4 Pooling of Word Vectors

At this stage, we calculate sense embeddings for each sense in the induced inventory. We assume that a word sense is a composition of words that represent the sense. We define a sense vector as a function of word vectors representing cluster items. Let W be a set of all words in the training corpus and let $S_i = \{w_1, \ldots, w_n\} \subseteq W$ be

	TWSI	JBT	w2v
table (furniture)	counter, console, bench, dinner table, dining table, desk, surface, bar, board	chair, room, desk, pulpit, couch, furniture, fireplace, bench, door, box, railing, tray	tray, bottom, bucket, basket, cup, pile, bracket, pot, cue, plate, jar, platter, ladder
table (data)	chart, list, index, graph, columned list, tabulation, standings, diagram, ranking	procedure, system, datum, process, mechanism, tool, method, database, calculation, scheme	diagram, brackets, stack, list, parenthesis, playfield, dropdown, cube, hash, results, tab
table (negotiations)	surface, counter, console, bargaining table, platform, negotiable, machine plate, level	—	—
table (geo)	level, plateau, plain, flatland, saturation level, water table, geographical level, water level	—	—

Table 2: Word sense clusters from inventories derived from the Wikipedia corpus via crowdsourcing (TWSI), JoBimText (JBT) and word embeddings (w2v). The sense labels are introduced for readability.

a sense cluster obtained during the previous step. Consider a function $vec_w : W \rightarrow \mathbb{R}^m$ that maps words to their vectors and a function $\gamma_i : W \rightarrow \mathbb{R}$ that maps cluster words to their weight in the cluster S_i. We experimented with two ways to calculate sense vectors: unweighted average of word vectors:

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

and weighted average of word vectors:

$$\mathbf{s}_i = \frac{\sum_{k=1}^{n} \gamma_i(w_k) vec_w(w_k)}{\sum_{k=1}^{n} \gamma_i(w_k)}.$$

Table 1 provides an example of weighted pooling results. While the original neighbours of the word "table" contain words related to both furniture and data, the neighbours of the sense vectors are either related to furniture or data, but not to both at the same time. Besides, each neighbour of a sense vector has a sense identifier as we calculate cosine between sense vectors, not word vectors.

4 Word Sense Disambiguation

This section describes how sense vectors are used to disambiguate a word in a context.

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

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

where $\bar{\mathbf{c}}_c$ is the mean of context embeddings: $k^{-1} \sum_{i=1}^{k} vec_c(c_i)$ and functions $vec_c : W \rightarrow \mathbb{R}^m$ map context words to context embeddings. Using the mean of context embeddings to calculate sense probability is natural with the CBOW because this model optimises exactly the same mean to have high scalar product with word embeddings for words occurred in context and low scalar product for random words (Mikolov et al., 2013).

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

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

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

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

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

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

	Full TWSI		Balanced TWSI	
	w2v	**JBT**	**w2v**	**JBT**
no filter	0.676	0.669	0.383	0.397
filter, $p = 5$	0.679	0.674	0.386	0.403
filter, $p = 3$	0.681	0.676	0.387	0.409
filter, $p = 2$	**0.683**	**0.678**	**0.389**	**0.410**
filter, $p = 1$	**0.683**	0.676	**0.390**	**0.410**

Table 4: Influence of context filtering on disambiguation in terms of F-score. The models were trained on Wikipedia corpus; the w2v is based on weighted pooling and similarity-based disambiguation. All differences between filtered and unfiltered models are significant ($p < 0.05$).

5 Experiments

We evaluate our method on two complementary datasets: (1) a crowdsourced collection of sense-labeled contexts; and (2) a commonly used SemEval dataset.

5.1 Evaluation on TWSI

The goal of this evaluation is to test different configurations of our approach on a large-scale dataset, i.e. it is used for development purposes.

Dataset. This test collection is based on a large-scale crowdsourced resource by Biemann (2012) that comprises 1,012 frequent nouns with average polysemy of 2.26 senses per word. For these nouns the dataset provides 145,140 annotated sentences sampled from Wikipedia. Besides, it is accompanied by an explicit sense inventory, where each sense is represented with a list of words that can substitute target noun in a given sentence.

The sense distribution across sentences in the dataset is skewed, resulting in 79% of contexts assigned to the most frequent senses. Therefore, in addition to the full TWSI dataset, we also use a balanced subset that has no bias towards the Most Frequent Sense (MFS). This dataset features 6,165 contexts with five contexts per sense excluding monosemous words.

Evaluation metrics. To compute WSD performance, we create an explicit mapping between the system-provided sense inventory and the TWSI senses: senses are represented as bag of words vectors, which are compared using cosine similarity. Every induced sense gets assigned to at most one TWSI sense. Once the mapping is completed, we can calculate precision and recall of sense prediction with respect to the original TWSI labeling.

Performance of a disambiguation model depends on quality of the sense mapping. These baselines facilitate interpretation of results:

- **Upper bound of the induced inventory** selects the correct sense for the context, but only if the mapping exist for this sense.
- **MFS of the TWSI inventory** assigns the most frequent sense in the TWSI dataset.
- **MFS of the induced inventory** assigns the identifier of the largest sense cluster.
- **Random sense baseline** of the TWSI and induced sense inventories.

Discussion of results. Table 2 presents examples of the senses induced via clustering of nearest neighbours generated by word embeddings (w2v) and JBT as compared to the inventory produced via crowdsourcing (TWSI). The TWSI contains more senses (2.26 on average), while induced ones have less senses (1.56 and 1.64, respectively). The senses in the table are arranged in the way they are mapped to TWSI during evaluation.

Table 3 illustrates how the granularity of the inventory influences WSD performance. The more granular the sense inventory, the better the match between the TWSI and the induced inventory can be established (mind that we map every induced sense to at most one TWSI sense). Therefore, the upper bound of WSD performance is maximal for the most fine-grained inventories.

However, the relation of actual WSD performance to granularity is inverse: the lower the number of senses, the higher the WSD performance (in the limit, we converge to the strong MFS baseline). We select a coarse-grained inventory for our further experiments (n=200, k = 15).

Table 4 illustrates the fact that using context filtering positively impacts disambiguation performance, reaching optimal characteristics when two context words are used.

Finally, Figure 3 presents results of our experiments on the full and sense-balanced TWSI datasets. First of all, our models significantly outperform random sense baseline of both TWSI and induced inventories. Secondly, we observe that pooling vectors using similarity scores as weights is better than unweighted pooling. Indeed, some clusters may contain irrelevant words and thus their contribution should be discounted. Third, we observe that using similarity-based disambiguation mechanism yields better results as compared

Inventory	#Senses	Upper-bound of Inventory			Probability-based WSD		
		Prec.	Recall	F-score	Prec.	Recall	F-score
TWSI	2.26	1.000	1.000	1.000	0.484	0.483	0.484
w2v wiki, $k = 15$	1.56	1.000	0.479	0.648	0.367	0.366	0.366
JBT wiki, $n = 200$, $k = 15$	1.64	1.000	0.488	0.656	**0.383**	**0.383**	**0.383**
JBT ukWaC, $n = 200$, $k = 15$	1.89	1.000	0.526	0.690	0.360	0.360	0.360
JBT wiki, $n = 200$, $k = 5$	2.55	1.000	0.598	0.748	0.338	0.338	0.338
JBT wiki, $n = 100$, $k = 5$	3.59	1.000	0.671	0.803	0.305	0.305	0.305
JBT wiki, $n = 50$, $k = 5$	5.13	**1.000**	**0.724**	**0.840**	0.275	0.275	0.275

Table 3: Upper-bound and actual value of the WSD performance on the sense-balanced TWSI dataset, function of sense inventory used for unweighted pooling of word vectors.

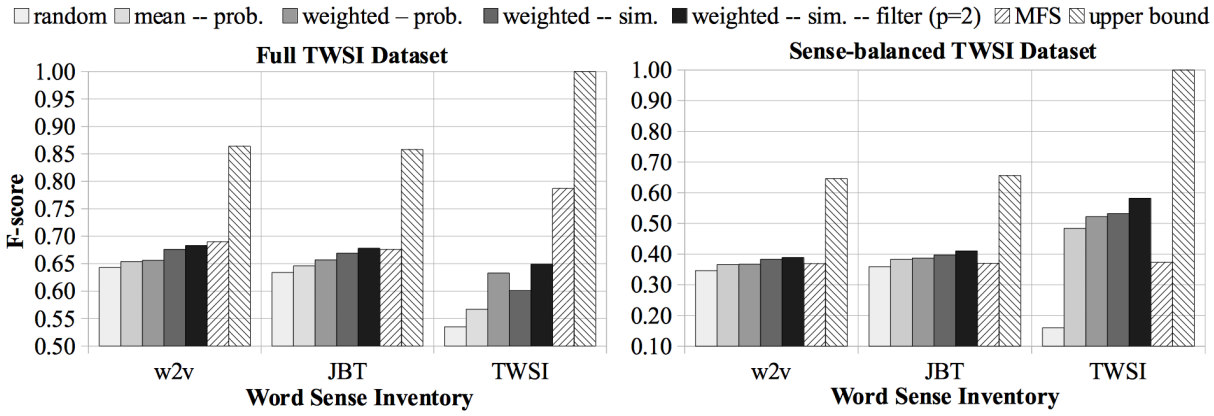

Figure 3: WSD performance of our method trained on the Wikipedia corpus on the full (on the left) and on the sense-balanced (on the right) TWSI dataset. The w2v models are based on the CBOW with 300 dimensions and context window size 3. The JBT models are computed using the Malt parser.

to the mechanism based on probabilities. Indeed, cosine similarity between embeddings proved to be useful for semantic relatedness, yielding state-of-the-art results (Baroni et al., 2014), while there is less evidence about successful use-cases of the CBOW as a language model.

Fourth, we confirm our observation that filtering context words positively impacts WSD performance. Finally, we note that models based on JBT- and w2v-induced sense inventories yield comparable results. However, the JBT inventory shows higher performance (0.410 vs 0.390) on the balanced TWSI, indicating the importance of a precise sense inventory. Finally, using the "gold" TWSI inventory significantly improves the performance on the balanced dataset outperforming models based on induced inventories.

5.2 Evaluation on SemEval-2013 Task 13

The goal of this evaluation is to compare the performance of our method to state-of-the-art unsupervised WSD systems.

Dataset. The SemEval-2013 task 13 "Word Sense Induction for Graded and Non-Graded Senses" (Jurgens and Klapaftis, 2013) provides 20 nouns, 20 verbs and 10 adjectives in WordNet-sense-tagged contexts. It contains 20-100 contexts per word, and 4,664 contexts in total, which were drawn from the Open American National Corpus. Participants were asked to cluster these 4,664 instances into groups, with each group corresponding to a distinct word sense.

Evaluation metrics. Performance is measured with three measures that require a mapping of sense inventories (Jaccard Index, Tau and WNDCG) and two cluster comparison measures (Fuzzy NMI and Fuzzy B-Cubed).

Discussion of results. We compare our approach to SemEval participants and the AdaGram sense embeddings. The *AI-KU* system (Baskaya et al., 2013) directly clusters test contexts using the k-means algorithm based on lexical substitution features. The *Unimelb* system (Lau et al., 2013) uses a hierarchical topic model to induce and dis-

Model		Supervised Evaluation			Clustering Evaluation	
		Jacc. Ind.	**Tau**	**WNDCG**	**F.NMI**	**F.B-Cubed**
Baselines	One sense for all	0.171	0.627	0.302	0.000	0.631
	One sense per instance	0.000	0.953	0.000	0.072	0.000
	Most Frequent Sense (MFS)	0.579	0.583	0.431	–	–
SemEval	AI-KU (add1000)	0.176	0.609	0.205	0.033	0.317
	AI-KU	0.176	0.619	0.393	0.066	0.382
	AI-KU (remove5-add1000)	0.228	0.654	0.330	0.040	0.463
	Unimelb (5p)	0.198	0.623	0.374	0.056	0.475
	Unimelb (50k)	0.198	0.633	0.384	0.060	0.494
	UoS (#WN senses)	0.171	0.600	0.298	0.046	0.186
	UoS (top-3)	0.220	0.637	0.370	0.044	0.451
	La Sapienza (1)	0.131	0.544	0.332	–	–
	La Sapienza (2)	0.131	0.535	0.394	–	–
Sense emb.	AdaGram, α = 0.05, 100 dim. vectors	0.274	0.644	0.318	0.058	0.470
Our models	w2v – weighted – sim. – filter ($p = 2$)	0.197	0.615	0.291	0.011	0.615
	w2v – weighted – sim. – filter ($p = 2$): nouns	0.179	0.626	0.304	0.011	0.623
	JBT – weighted – sim. – filter ($p = 2$)	0.205	0.624	0.291	0.017	0.598
	JBT – weighted – sim. – filter ($p = 2$): nouns	0.198	0.643	0.310	0.031	0.595
	TWSI – weighted – sim. – filter ($p = 2$): nouns	0.215	0.651	0.318	0.030	0.573

Table 5: The best configurations of our method selected on the TWSI dataset on the SemEval 2013 Task 13 dataset. The w2v-based methods rely on the CBOW model with 100 dimensions and context window size 3. The JBT similarities were computed using the Malt parser. All systems were trained on the ukWaC corpus.

ambiguate word senses. The *UoS* system (Hope and Keller, 2013) is most similar to our approach: to induce senses it builds an ego-network of a word using dependency relations, which is subsequently clustered using a simple graph clustering algorithm. The *La Sapienza* system (Agirre and Soroa, 2009), relies on WordNet to get word senses and perform disambiguation.

Table 5 shows a comparative evaluation of our method on the SemEval dataset. Like above, dependency-based (JBT) word similarities yield slightly better results than word embedding similarity (w2v) for inventory induction. In addition to these two configurations, we also built a model based on the TWSI sense inventory (only for nouns as the TWSI contains nouns only). This model significantly outperforms both JBT- and w2v-based models, thus precise sense inventories greatly influence WSD performance.

As one may observe, performance of the best configurations of our method is comparable to the top-ranked SemEval participants, but is not systematically exceeding their results. AdaGram sometimes outperforms our method, sometimes it is on par, depending on the metric. We interpret these results as an indication of comparability of our method to state-of-the-art approaches.

Finally, note that none of the unsupervised WSD methods discussed in this paper, including the top-ranked SemEval submissions and AdaGram, were able to beat the most frequent sense baselines of the respective datasets (with the exception of the balanced version of TWSI). Similar results are observed for other unsupervised WSD methods (Nieto Piña and Johansson, 2016).

6 Conclusion

We presented a novel approach for learning of multi-prototype word embeddings. In contrast to existing approaches that learn sense embeddings directly from the corpus, our approach can operate on existing word embeddings. It can either induce or reuse a word sense inventory. Experiments on two datasets, including a SemEval challenge on word sense induction and disambiguation, show that our approach performs comparably to the state of the art.

An implementation of our method with several pre-trained models is available online.[3]

Acknowledgments

We acknowledge the support of the Deutsche Forschungsgemeinschaft (DFG) foundation under the project "JOIN-T: Joining Ontologies and Semantics Induced from Text".

[3] https://github.com/tudarmstadt-lt/sensegram

References

Eneko Agirre and Philip Edmonds. 2007. *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media.

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 33–41, Athens, Greece.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145, Mexico City, Mexico.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 238–247, Baltimore, Maryland. Association for Computational Linguistics.

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. AI-KU: Using Substitute Vectors and Co-Occurrence Modeling for Word Sense Induction and Disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM): SemEval 2013)*, volume 2, pages 300–306, Atlanta, Georgia, USA.

Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95.

Chris Biemann. 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*, pages 73–80, New York City, USA.

Chris Biemann. 2010. Co-occurrence cluster features for lexical substitutions in context. In *Proceedings of the 5th Workshop on TextGraphs in conjunction with ACL 2010*, Uppsala, Sweden.

Chris Biemann. 2012. Turk Bootstrap Word Sense Inventory 2.0: A Large-Scale Resource for Lexical Substitution. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 4038–4042, Istanbul, Turkey.

Antoine Bordes, Weston Jason, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proc. AAAI*, San Francisco, CA, USA.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A unified multilingual semantic representation of concepts. In *Proceedings of the Association for Computational Linguistics*, pages 741–751, Beijing, China.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

David Hope and Bill Keller. 2013. MaxMax: A Graph-based Soft Clustering Algorithm Applied to Word Sense Induction. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, pages 368–381, Samos, Greece. Springer-Verlag.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the ACL*, pages 873–882, Jeju Island, Korea.

David Jurgens and Ioannis Klapaftis. 2013. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 290–299, Atlanta, Georgia, USA.

Dan Klein, Kristina Toutanova, H. Tolga Ilhan, Sepandar D. Kamvar, and Christopher D. Manning. 2002. Combining Heterogeneous Classifiers for Word-Sense Disambiguation. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, volume 8, pages 74–80, Philadelphia, PA.

Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013. unimelb: Topic Modelling-based Word Sense Induction. In *Second Joint Conference on Lexical and Computational Semantics (*SEM): SemEval 2013)*, volume 2, pages 307–311, Atlanta, Georgia, USA.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the Empirical Methods in Natural Language Processing*, volume 10, pages 41–48, Philadelphia, PA.

Michael Lesk. 1986. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *Proceedings of the 5th International Conference on Systems Documentation*, pages 24–26, Toronto, ON, Canada. ACM.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Conference on Empirical Methods in Natural Language Processing, EMNLP'2015*, pages 1722–1732, Lisboa, Portugal.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of ICML*, volume 98, pages 296–304, Madison, WI, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop at International Conference on Learning Representations (ICLR).*, pages 1310–1318, Scottsdale, AZ, USA.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar.

Luis Nieto Piña and Richard Johansson. 2015. A simple and efficient method to generate word sense representations. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, Hissar, Bulgaria, September.

Luis Nieto Piña and Richard Johansson. 2016. Embedding senses for efficient graph-based word sense disambiguation. In *Proceedings of TextGraphs-10, Proceedings of the Human Language Technology Conference of the NAACL*, San Diego, United States.

Alexander Panchenko. 2013. *Similarity measures for semantic relation extraction*. Ph.D. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619. ACM.

Ted Pedersen and Rebecca Bruce. 1997. Distinguishing word senses in untagged text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 197–207, Providence, RI.

Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan. 2005. Maximizing semantic relatedness to perform word sense disambiguation. *University of Minnesota supercomputing institute research report UMSI*, 25:2005.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117, Los Angeles, California.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the Association for Computational Linguistics*, pages 1793–1803, Beijing, China. Association for Computational Linguistics.

Eugen Ruppert, Jonas Klesy, Martin Riedl, and Chris Biemann. 2015. Rule-based dependency parse collapsing and propagation for german and english. In *Proceedings of the GSCL 2015*, pages 58–66, Duisburg, Germany.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational linguistics*, 24(1):97–123.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*, pages 151–160, Dublin, Ireland.

Jean Véronis. 2004. HyperLex: Lexical cartography for information retrieval. *Computer Speech and Language*, 18:223–252.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Taipei, Taiwan.

Ziqi Zhang, Anna Lisa Gentile, and Fabio Ciravegna. 2013. Recent advances in methods of lexical semantic relatedness–a survey. *Natural Language Engineering*, 19(04):411–479.