Linear Algebraic Structure of Word Senses, with Applications to Polysemy

Sanjeev Arora Yuanzhi Li Yingyu Liang Tengyu Ma Andrej Risteski [*]

Abstract

Word embeddings are ubiquitous in NLP and information retrieval, but it's unclear what they represent when the word is *polysemous*, i.e., has multiple senses. Here it is shown that multiple word senses reside in linear superposition *within* the word embedding and can be recovered by simple sparse coding.

The success of the method —which applies to several embedding methods including **word2vec**— is mathematically explained using the *random walk on discourses* model (Arora et al., 2016). A novel aspect of our technique is that each word sense is also accompanied by one of about 2000 "discourse atoms" that give a succinct description of which other words co-occur with that word sense. Discourse atoms seem of independent interest, and make the method potentially more useful than the traditional clustering-based approaches to polysemy.

1 Introduction

Word embeddings represent the "meaning" of a word as a real-valued vector. Their construction typically uses Firth's hypothesis that a word's sense is captured by the distribution of other words around it (Firth, 1957). Classical *vector space models* (see the survey (Turney et al., 2010)) use simple linear algebra on the matrix of word-word co-occurrence counts, whereas recent neural network and energy-based models such as **word2vec** use nonconvex optimization (Mikolov et al., 2013a;b). Word embeddings are useful in many NLP tasks, and seem involved in understanding neural encoding of semantics (Mitchell et al., 2008).

However, it has been unclear what embeddings represent when the word is polysemous, i.e., has multiple word senses. The monolithic approach of representing a word by a single word vector, with its inner information extracted only via inner product, is felt to fail in capturing this finer structure (Griffiths et al., 2007; Reisinger and Mooney, 2010).

The current paper goes beyond this monolithic view, by describing how multiple senses of a word actually reside *within* the word embedding in linear superposition, and can be recovered by simple sparse coding. The linear structure is revealed in Section 2 via a surprising thought experiment.

Our work is inspired by the discovery that word analogies can be solved by linear algebraic methods (Mikolov et al., 2013b). However, the mathematical explanation of the efficacy of our method (see Section 3 and Section 5) uses a recent *random walk on discourses* model of Arora et al. (2016). Our experiments use 300-dimensional embeddings created using a Wikipedia corpus of 3 billion tokens (Wikimedia, 2012) using their embedding, but word senses can also be recovered – with some loss in performance – for other recent embeddings such as **word2vec** and **GloVe** as well as the older vector space methods such as PMI (Church and Hanks, 1990). These methods are known to be interrelated; see (Levy and Goldberg, 2014; Arora et al., 2016).

The experimental demonstration of the performance of our method for extracting word senses appears in Section 4, together with a comparison to older graph clustering-based approaches.

*Princeton University, Computer Science Department. {arora,yuanzhil,yingyul,tengyu,risteski}@cs.princeton.edu.
This work was supported in part by NSF grants CCF-0832797, CCF-1117309, CCF-1302518, DMS-1317308, Simons Investigator Award, and Simons Collaboration Grant. Tengyu Ma was also supported by Simons Award for Graduate Students in Theoretical Computer Science.

1.1 Comparison with related work

Automatic learning of word senses (Word Sense Induction) usually is done via variants of the *exemplar* approach of Yarowsky (1995) (see also (Schutze, 1998; Reisinger and Mooney, 2010; Di Marco and Navigli, 2013)), which identifies multiple word senses by clustering neighboring words. Firth's hypothesis justifies this approach, since the senses are different if and only if their contexts involve different word distributions.[1]

The following quote from the abstract of (Iacobacci et al., 2015) is representative of the general feeling about word embeddings: "*word embeddings are by their very nature unable to capture polysemy, as different meanings of a word are conflated into a single representation.*" One line of work tries to capture word senses by augmenting embeddings into more complicated representations than a single vector (e.g., (Murphy et al., 2012; Huang et al., 2012)). However, to relate these different word senses into a sense inventory seems to require an external source such as WordNet (see also (Faruqui et al., 2015a)), which is problematic in languages that lack WordNets. By contrast, our approach is purely unsupervised and exhibits that the word senses reside within existing embeddings, including the classical PMI-based ones.

An important feature of our approach is that the corpus is used only to create a 300-dimensional embedding for each word. By contrast, the graph-theoretic clustering approaches work directly with the corpus and are orders of magnitude slower. Section 3.1 explains how to interpret sparse coding on word embeddings as a linear algebraic surrogate for graph clustering.

Another difference from past clustering approaches is that the senses of different words recovered by our method are interconnected via the notion of *atoms of discourse*, a new notion introduced in Section 3. To give an example, the "article of clothing" sense of *tie* is automatically related to words like *trousers, blouse* etc. (Table 4).

Word sense disambiguation (WSD) is the supervised problem of identifying the sense of a word used in each occurrence during a document, and is left for future work. But our techniques may help create resources useful for WSD such as annotated corpora or WordNet (Fellbaum, 1998) that are lacking in many languages.

Sparse coding applied to word embeddings has been used before as a way of getting representations that are more useful in other NLP tasks (Faruqui et al., 2015b), but not in connection with polysemy.

2 Linear Structure of Word Senses

In this section we propose a thought experiment that uncovers, qualitatively and quantitatively, the linear algebraic structure of a polysemous word with respect to its senses in the embedding space.

Consider a polysemous word, say *tie*, which can refer to an article of clothing, or a drawn match, or a physical act. Let's take the viewpoint —simplistic yet instructive— that it is a single lexical token that represents unrelated words *tie1, tie2, ...*. (Of course, the "article of clothing" sense is metaphorically very related to "physical act", but the two appear very different in terms of distributions of neighboring words.) Now we describe a surprising experiment that suggests that the embedding for "tie" should be approximately a weighted sum of the (hypothethical) embeddings of *tie1, tie2, ...*.

The experiment consists of creating an artificial polysemous word w_{new} by combining two random (and hence presumably unrelated) words w_1, w_2, where w_1 is more frequent than w_2. Every occurrence of w_1 or w_2 now counts as an occurrence of w_{new}. Next, an embedding is computed for w_{new} using the same embedding method while deleting embeddings for w_1, w_2 but preserving the embeddings for all other words. This experiment was repeated with a wide range of values for the ratio between the frequencies of w_1 and w_2. It was always found that the new vector $v_{w_{new}}$ lies close to the subspace spanned by v_{w_1} and v_{w_2}: the cosine of its angle with the subspace is on average 0.97 with standard deviation 0.02. Thus $v_{w_{new}} \approx \alpha v_{w_1} + \beta v_{w_2}$. The question is how do α, β depend upon the ratio r of frequencies of w_1, w_2. Figure 1 shows that $\alpha \approx 1$ whereas

$$\beta \approx 1 - c \lg r$$

for some constant $c \approx 0.5$. (Note this formula is meaningful for sufficiently small r, i.e. when $r < 10^{1/c} \approx 100$.

[1]This raises well-known controversies about word senses, e.g., whether *paint* is used in different senses in *paint the wall* versus *paint a mural.* It is, if we equate word sense with distribution of neighboring words.

Atom 1978	825	231	616	1638	149	330
drowning	instagram	stakes	membrane	slapping	orchestra	conferences
suicides	twitter	thoroughbred	mitochondria	pulling	philharmonic	meetings
overdose	facebook	guineas	cytosol	plucking	philharmonia	seminars
murder	tumblr	preakness	cytoplasm	squeezing	conductor	workshops
poisoning	vimeo	filly	membranes	twisting	symphony	exhibitions
commits	linkedin	fillies	organelles	bowing	orchestras	organizes
stabbing	reddit	epsom	endoplasmic	slamming	toscanini	concerts
strangulation	myspace	racecourse	proteins	tossing	concertgebouw	lectures
gunshot	tweets	sired	vesicles	grabbing	solti	presentations

Table 1: Some discourse atoms and their nearest 9 words. By Eqn. (2) words most likely to appear in a discourse are those nearest to it.

Moreover, the Pearson correlation coefficient between β and $-\lg r$ is 0.67.) This logarithmic behavior is actually a good feature, because it allows the less dominant/frequent sense to have a *superproportionate* contribution to $v_{w_{new}}$, thus making it detectable *in principle* despite noise/error.

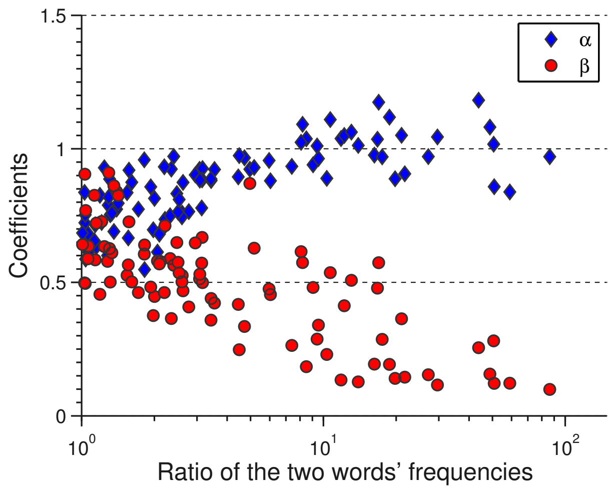


Figure 1: Thought experiment of creating an artificial polysemous word w_{new} by merging two words w_1 and w_2. The fitted word vector $v_{w_{new}}$ is found to be very close to $\alpha v_{w_1} + \beta v_{w_2}$. The plot shows the coefficients of α and β for 100 artificial polysemous words versus the the ratio r of the frequencies of w_1 and w_2 (plotted on a log scale).

A mathematical explanation for this experiment using the model by Arora et al. (2016) appears in Section 5 but it is not needed in the other sections.

3 Extracting Senses via Atoms of Discourse

The above experiment suggests that

$$v_{tie} \approx \alpha_1 \cdot v_{tie1} + \alpha_2 \cdot v_{tie2} + \alpha_3 \cdot v_{tie3} + \cdots \tag{1}$$

but this is insufficient to mathematically pin down the senses, since v_{tie} can be expressed in infinitely many ways as such a combination. To pin down the senses we will interrelate the senses of different words —for example, relate the "article of clothing" sense *tie1* with *shoe, jacket* etc. This relies on a recent paper that connects senses to directions in the embedding space.

Random walk on discourses model. Arora et al. (2016) introduce the *random walk on discourses* model which suggests that directions in embedding space correspond to local topic structure in the corpus, which they term *discourse*. (Aside: their main goal was to give theoretical justification for earlier methods like PMI and word2vec.)

The model builds upon an earlier *loglinear topic model* of (Mnih and Hinton, 2007), which allows in principle a continuum of topics: each word w is represented by a vector $v_w \in \Re^d$ and a topic is just any unit vector c in $\Re^d$. Topic c defines a distribution over words as follows:

$$\Pr[w \mid c] \propto \exp(c \cdot v_w). \tag{2}$$

The Random-Walks model treats c as a *discourse* ("what is being talked about at this place in the corpus"). The corpus is assumed to be generated by allowing c to make a slow geometric random walk over the unit sphere in $\Re^d$: when the walk is at c, a few words are emitted using iid samples from the word distribution (2), which due to its loglinear form, strongly favors words that are close to c (in cosine similarity).

When this model is fitted to real corpora such as Wikipedia, it is found that the distribution (2), though it gives nonzero probability to every word, is fairly concentrated on a small number of words, around 100 or so. So a vector c defines a fairly distinct but narrow "discourse," whose "meaning" can be discerned by examining its closest word-vectors (cosine similarity). Discourses defined by two c, c' look similar to humans if their inner product ≥ 0.85, and quite different if the inner product ≤ 0.5.

Extracting word senses. Recall that senses *tie1, tie2,...* correspond to different word distributions occuring around *tie*, which we interpret as distinct discourses (= directions). One imagines that there is a "clothing" discourse that has high probability of outputting the *tie1* sense, and also of outputting related words such as *shoe, jacket* etc. By (2) the probability of being output by a discourse is determined by the inner product, so one expects that the vector for "clothing" discourse has high inner product with all of *shoe, jacket, tie1* etc., and thus can stand as surrogate for v_{tie1} in (1)! This motivates the following global optimization.

Given word vectors in $\Re^d$, totaling about $60,000$ in this case, a sparsity parameter k, and an upper bound m, find a set of unit vectors $A_1, A_2, \ldots, A_m$ such that

$$v_w = \sum_{j=1}^{m} \alpha_{w,j} A_j + \eta_w \tag{3}$$

where at most k of the coefficients $\alpha_{w,1}, \ldots, \alpha_{w,m}$ are nonzero (so-called hard sparsity constraint*), and η_w is a noise vector.* Both A_j's and $\alpha_{w,j}$'s are unknowns in this optimization, so the problem is non-convex. This is nothing but *sparse coding*, useful in neuroscience (Olshausen and Field, 1997) and also in image processing, computer vision, etc. We solve it using standard k-SVD algorithm (Damnjanovic et al., 2010), using m about 2000. (Details are below.)

This optimization is a surrogate for the desired expansion of v_{tie} as in (3) because one can hope that $A_1, \ldots A_m$ will represent important discourses in the corpus, which we refer to as *atoms of discourse*, and consequently will contain atoms corresponding to *clothing*, *sports matches* etc. that will have high inner product (close to 1) with *tie1*, *tie2* respectively. Furthermore, restricting m to be much smaller than the number of words ensures that the same discourse needs to be used for multiple words.

Representative atoms appear in Table 1; each represents some clear, narrow "topic." Each word is represented using five atoms, which usually capture distinct senses (with some noise/mistakes). See Table 4 for the senses recovered for *tie*.

3.1 Relationship to clustering

Sparse coding as described in (3) is usually solved –for any choice of m, k—by using alternating minimization to find the A_i's that minimize the ℓ_2-reconstruction error

$$\sum_w |v_w - \sum_{j=1}^{m} \alpha_{w,j} A_j|_2^2.$$

tie					spring				
trousers	season	scoreline	wires	operatic	beginning	dampers	flower	creek	humid
blouse	teams	goalless	cables	soprano	until	brakes	flowers	brook	winters
waistcoat	winning	equaliser	wiring	mezzo	months	suspension	flowering	river	summers
skirt	league	clinching	electrical	contralto	earlier	absorbers	fragrant	fork	ppen
sleeved	finished	scoreless	wire	baritone	year	wheels	lilies	piney	warm
pants	championship	replay	cable	coloratura	last	damper	flowered	elk	temperatures

Table 2: Five discourse atoms linked to the words "tie" and "spring". Each atom is represented by its nearest 6 words. The algorithm often makes a mistake in the last atom (or two), as happened here.

This objective function reveals sparse coding to be a linear algebraic analogue of overlapping clustering, whereby the A_i's act as *cluster centers* and each v_w is assigned in a soft way (using the coefficients $\alpha_{w,j}$, of which only k are nonzero) to a linear combination of at most k of them. In fact this clustering viewpoint is also the basis of the alternating minimization algorithm. (Notice, if $k = 1$ then each v_w has to be assigned to a single cluster, which is familiar geometric clustering with squared ℓ_2 distance.)

In practice, sparse coding optimization produces coefficients α_{ij}'s almost all positive even though they're *unconstrained*. (Probably because the appearances of a word are best explained by what discourse *is* being used to generate it, rather than what discourses are *not* being used.) Furthermore, as is usual in sparse coding, the nonzero coefficients for *tie* usually correspond to basis vectors with low pairwise inner product, meaning they are pairwise quite different. Effectively, the sparse coding ends up writing v_i's as weighted sums of five fairly different A_j's with which it has positive inner product.

There is an intuitive connection to classical WSI approaches, which would cluster the words appearing around *tie* in the corpus via clustering on a weighted graph whose edge weights are similarity scores (based upon distributional similarity) for word pairs. In our approach the word embedding for *tie* contains a summary of how often it appears in all the discourses of the corpus, which implicitly captures its similarity with all other words. The k cluster centers A_j's that *tie* is softly assigned to represent interesting discourses that it has good probability of occurring in (i.e., has good inner product with). The fact that A_j is an interesting or significant discourse is guaranteed by the fact that sparse coding ensures that each basis element is (softly) chosen by many other words in addition to *tie*.

3.2 Relationship to topic models

Atoms of discourse may be reminiscent of results from other automated methods for obtaining a thematic understanding of text, such as topic modeling, described in the survey (Blei, 2012). The meta atoms are highly reminiscent of past results from *hierarchical topic models* (Griffiths and Tenenbaum, 2004). Indeed, the model (2) used to compute the word embeddings is related to a log-linear topic model from (Mnih and Hinton, 2007). However, the discourses here are computed via sparse coding on word embeddings, which is very distinct from topic modeling. The atoms are also reminiscent of coherent "word clusters" detected in the past using *Brown clustering*, or even sparse coding (Murphy et al., 2012). The novelty in the current paper is a clear interpretation of the sparse coding results –as atoms of discourse— as well as its use to capture different word senses.

3.3 Experiments with atoms of discourse

Experimentation showed that the best sparsity parameter[2] in sparse coding—i.e., the maximum number of allowed senses per word— is 5.

The best basis size —i.e., the number of atoms of discourses— was found to be around 2000. This was estimated by re-running the sparse coding (k-SVD) algorithm multiple times with different random

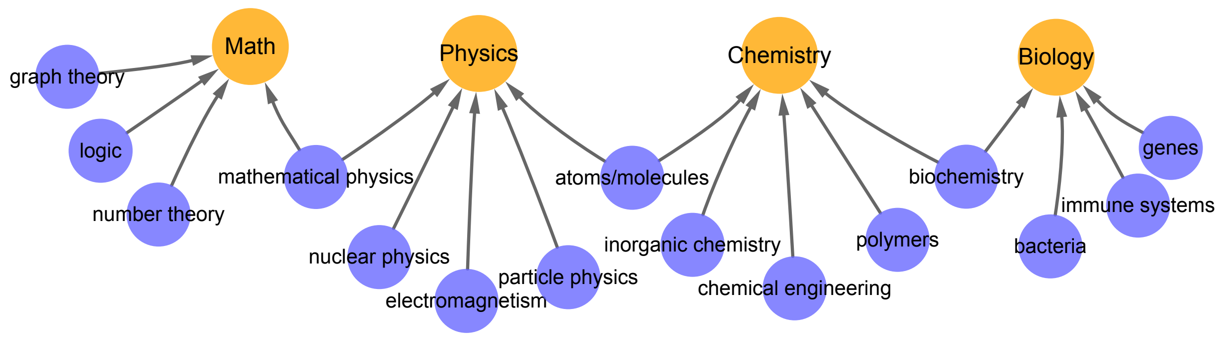
[2]Some plausible alternatives —based upon the intuition that the number of senses a word has is correlated with its overall frequency— were explored but seemed to be worse or no better.

initializations, whereupon substantial overlap was found between the two bases: a large fraction of vectors in one basis were found to have a very close vector in the other. Thus combining the bases while merging duplicates yielded a basis of about the same size. Around 100 atoms are used by a large number of words or have no close by words. They appear semantically meaningless and thus filtered.

As mentioned, we refer to the significant discourses represented by the basis vectors as *atoms of discourse*. The "content" of a discourse atom can be discerned by looking at the set of nearby words (by cosine). Table 1 contains some examples of the discourse atoms, and Table 4 shows the 5 discourse atoms linked to the words "tie" and "spring."

Hierarchy of discourse atoms. The atoms are fairly fine-grained, but it is possible to extract more coarse-grained set of discourses. For instance, the discourse atoms found for *jazz, rock, classical* and *country* are more related to each other than to atoms about, say, *mathematics*. Again, model (2) suggests that similar discourse atoms should have higher inner product to each other, and thus sparse coding should be able to identify these similarities and create meta-discourse vectors such as *music*.

By some experimentation, best results involve sparse coding on discourse atoms using hard sparsity 2 and allowing a basis of size 200, which turn out to be *meta-discourse* vectors. Figure 2 shows such an example using atoms for scientific fields; more examples can be found in the full version. A discourse about an interdisciplinary science like *biochemistry* turns out to be approximately linear combinations of two meta-discourses of *biology* and *chemistry*.



Atom	28	2016	468	1318	411
	logic deductive propositional semantics	graph subgraph bipartite vertex	boson massless particle higgs	polyester polypropylene resins epoxy	acids amino biosynthesis peptide
tag	*logic*	*graph theory*	*particle physics*	*polymer*	*biochemistry*

Figure 2: Some atoms of discourse (small circles) related to scientific fields and their meta-atoms (large circles) found by the second level sparse coding. The connecting line corresponds to discourse atoms that use the meta atoms in the coding. Thus the atom for biochemistry is found to be expressed using a linear combination of the meta discourse vectors of chemistry and biology (plus noise).

Outputting relevant sentences To use this method to construct a dictionary (like WordNet) in a completely automated way, one would want, for each polysemous word some representative sentences illustrating its various senses. This can be done as follows. Noting that sentences in general correspond to more than one discourse atom (simply because *number of possible things one could talk about* exceeds 2000, the number of atoms), let us define the *semantic representation* of a sentence to be the best rank-3 approximation (via

	sentence
1	The spectrum of any commutative ring with the Zariski topology (that is, the set of all prime ideals) is compact.
2	The inner 15-point ring is guarded with 8 small bumpers or posts.
3	Allowing a Dect phone to ring and answer calls on behalf of a nearby mobile phone.
4	The inner plastid-dividing ring is located in the inner side of the chloroplast's inner.
5	Goya (wrestler), ring name of Mexican professional wrestler Gloria Alvarado Nava.
6	The Chalk Emerald ring, containing a top-quality 37-carat emerald, in the U.S. National Museum of Natural History.
7	Typically, elf circles were fairy rings consisting of a ring of small mushrooms.

Table 3: Relevant fragments from top 7 sentences identified by the algorithm for the word "ring." The math sense in the first sentence was missing in WordNet. More examples in the full version.

Principal Component Analysis) to the subspace spanned by the word embeddings of its words. For a given polysemous word we take its five atoms as well as atoms for its inflectional forms (e.g., past tense, plural etc., generated by (Manning et al., 2014)). This yields between 10 to 20 atoms for the word, whereupon each sentence is scored with respect to each atom by an appropriate cosine similarity between the atom and the semantic representation of the sentence. Finally output the top few sentences with highest scores. Table 3 presents the sentences found for the word "ring". More examples can be found in the full version.

4 A Quantitative Test

While the main purpose of the paper is to show the linear algebraic structure of word senses within existing embeddings, it is desirable to have some quantification of the efficacy of our approach. This runs into well-known difficulties, as reflected in the changing metrics for SENSEVAL tests over the years (Navigli and Vannella, 2013). Some metrics involve a custom similarity score based upon WordNet that is hard to interpret and may not be available for other languages. Here a new simple test —–inspired by word-intrusion tests for topic coherence (Chang et al., 2009)—is proposed, which has the advantages of being easy to understand, and can also be administered to humans.

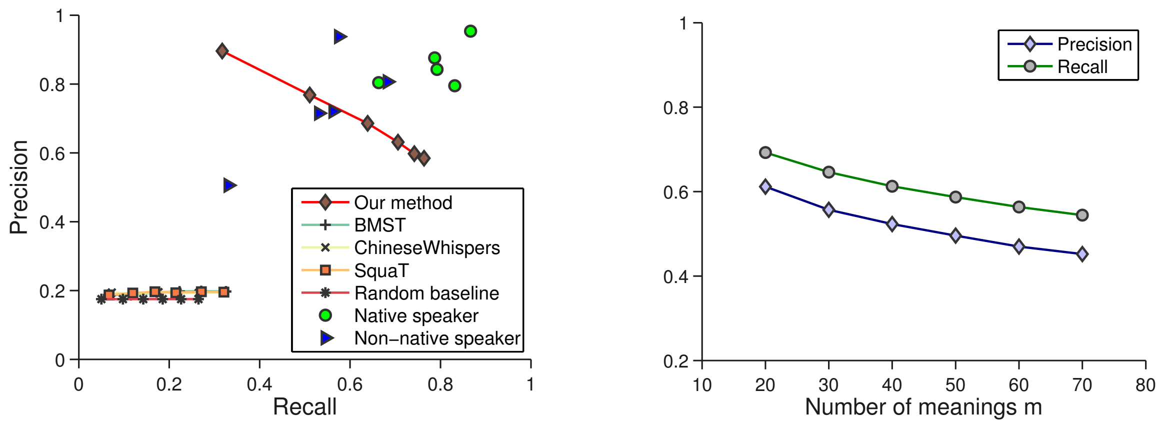
The testbed uses 200 polysemous words and their 704 senses according to WordNet. Each "sense" is represented by a set of 8 related words; these were collected from WordNet and online dictionaries by college students who were told to identify *most relevant* other words occurring in the online definitions of this word sense as well as in the accompanying illustrative sentences. These are considered as *ground truth* representation of the word sense. These 8 words are typically not synonyms; e.g., for the *tool/weapon* sense of "axe" they were: "handle, harvest, cutting, split, tool, wood, battle, chop."

The quantitative test is called *police line up*: the algorithm is given a random one of these 200 polysemous words and a set of m senses which contain the true sense for the word as well as some *distractors* (randomly picked senses from other words). The algorithm (or human) has to identify the word's true senses in this set. Our method is described in Algorithm 1.

The precision (fraction of senses that were correct) and recall (fraction of ground truth senses that were recovered) for different m and k are presented in Figure 3. For $m = 20$ and $k = 4$, our algorithm succeeds with precision 63% and recall 70%, and performance remains reasonable for $m = 50$. Giving the same test to humans[3] for $m = 20$ (see the left figure) suggests that our method performs similarly to non-native speakers.

Comparison to WSI using graph clustering. Tests were performed using WSI code released in the recent papers (Di Marco and Navigli, 2013; Vannella et al., 2014). Given a target polysemous word, these methods first construct a co-occurrence graph on words related to the target word, and then use some

[3]Human subjects are graduate students from science or engineering majors at major U.S. universities. Non-native speakers have 7 to 10 years of English language use/learning.



A **B**

Figure 3: Precision and recall in the polysemy test. (**A**) For each polysemous word, a set of $m = 20$ senses containing the ground truth senses of the word are presented. Human subjects are told that on average each word has 3.5 senses and were asked to choose the senses they thought were true. The algorithms select k senses for $k = 1, 2, \dots, 6$. For each k, each algorithm was run 5 times (standard deviations over the runs are < 0.02 and thus not plotted). (**B**) The performance of our method for $k = 4$ and $m = 20, 30, \dots, 70$.

Algorithm 1 Our method for the police line up test

```
1: Find the inflectional forms of the target word
2: Find 5 atoms for the target and each inflectional form. Let $C_a$ denote the union of the atoms.
3: For any word or atom $x$ and a set $Y$ of words, define similarity $\mathrm{s}(x, Y) = \sqrt{\sum_{y \in Y} \langle v_x, v_y \rangle^2}$
   and define the average similarities
   $$s_1^Y = \frac{\sum_{a' \in A} \mathrm{s}(v_{a'}, Y)}{|A|}, s_2^Y = \frac{\sum_{w' \in V} \mathrm{s}(v_{w'}, Y)}{|V|}$$
   where $A$ are all the atoms, $V$ are all the words
4: Initialize the set of candidate senses $C_s \leftarrow \emptyset$
5: for each atom $a \in C_a$ do
6:     Rank senses $L \in S$ by
         $\mathrm{score}(a, L) = \mathrm{s}(a, L) - s_1^L + \mathrm{s}(w, L) - s_2^L$
7:     Add the two senses with highest scores to $C_s$
8: end for
Ensure: The top $k$ senses with highest scores in $C_s$
```

clustering algorithm on the graph to produce several clusters of words, which are regarded as the senses and were used for grouping web search results. We tested three clustering algorithms (ChineseWhispers, BMST, and SquaT) provided.

Given the lineup of m senses (each a list of 8 words) and the identified clusters, we used the provided matching algorithm to match each sense with a cluster, and chose the top k senses according to the metric used for this matching. The hyperparameters are enumerated among 4^α times the default values in the software for $\alpha \in \{-2, -1, 0, 1, 2\}$ and the matching algorithm is enumerated among {WORD_OVERLAP, DEGREE_OVERLAP} provided in (Di Marco and Navigli, 2013). The best results are also plotted in Figure 3(A).

These results are much worse than ours, not much better than chance. Some human study suggested the following reason. The clusters returned by these methods are topically coherent but only a few are related to a recognizable sense for the word (i.e. precision is low). A typical example: *dog, dogs, breed* was a cluster for *abstract*. This was not an issue for the primary goal in those papers —clustering web search results—since a web search for *abstract* would not return pages related to *dogs, breed* etc. But on our test this will greatly hurt the performance. We did not investigate how to improve the clustering algorithms, or the implicit algorithm to match a given set of words to a cluster.

Other word embeddings. These were constructed using many past methods. They were used to generate atoms of discourse and given the police lineup test. They all do OK but with slightly lower performance. For $m = 20$ and $k = 4$, the precision/recall are lower by the following amounts: **GloVe** 0.3%/0.76%, **word2vec**(CBOW) 2.3%/4.9%, NNSE ((Murphy et al., 2012), matrix factorization on PMI to rank 300) 23%/23%.

5 The mathematical explanation

Section 2 concerned the experiment of an artificial polysemous word by merging two unrelated words w_1, w_2 with frequency ratio $r = \frac{\Pr[w_1]}{\Pr[w_2]} \geq 1$ into a new word w, whose embedding turns out to be close to $\alpha v_1 + \beta v_2$, where v_1 and v_2 are the embeddings of words w_1, w_2 (Figure 1). We now give a theoretical explanation. It exposes parameters that affect β and allows us to explain both the general behaviour of β and the spread in the actual values observed.

Main Claim: *Denoting* $\kappa_1 = \log(1 + 1/r)$ *and* $\kappa_2 = \log(1 + r)$*, we have*

$$v_w \approx (1 - \kappa_1 c_1)v_1 + (1 - \kappa_2 c_2)v_2$$

where c_1 *and* c_2 *are two small constants respectively.*

A claim like the above is of course impossible to prove unless we make assumptions about the word-occurrences patterns $\text{PMI}(\chi, w)$ and the word vectors v_χ. We now review the model and make some simplifying assumptions, which we justify intuitively.

First, we assume the original word vectors are trained from fitting the PMI model in Arora et al. (2016). It is based on the method-of-moments performed on (2), yielding an optimization reminiscent of the classical PMI method:

$$\sum_{w_1, w_2} \Pr(w_1, w_2) \cdot (\text{PMI}(w_1, w_2) - \langle v_{w_1}, v_{w_2} \rangle)^2 \tag{4}$$

where $\Pr(w_1, w_2)$ is the empirical probability that words w_1, w_2 occur within distance 5 (say) of each other in the corpus, and $\text{PMI}(w_1, w_2) = \log(\Pr[w_1, w_2] / \Pr[w_1] \Pr[w_2])$. (A similar but more calculationally involved explanation can be done using the SN model in Arora et al. (2016), which is empirically better.)

The first simplifying assumption is that the original word vectors perfectly fit the model:

Simplification 1: *The original word vectors perfectly fit the PMI model* (4), *i.e. every word* $\chi \neq w_1, w_2$ *satisfies* $\text{PMI}(\chi, w_1) = \langle v_\chi, v_1 \rangle$, $\text{PMI}(\chi, w_2) = \langle v_\chi, v_2 \rangle$.[4]

Next, we review how the new artificial word w relates to the PMI. By construction, the new artificial word w has co-occurrence probability $\Pr(w, \chi) = \Pr(w_1, \chi) + \Pr(w_2, \chi)$ for every other word χ, and marginal distribution $\Pr(w) = \Pr(w_1) + \Pr(w_2) = (1 + 1/r) \Pr(w_1) = (1 + r) \Pr(w_2)$. The new vector v_w for it is obtained by optimizing (4) using the new co-occurrence probabilities. All the other word vectors remained unchanged, so the optimization reduces to

$$\begin{aligned} v_w &= \text{argmin}_z f(z) \\ &= \sum_\chi \Pr(\chi, w) \cdot (\text{PMI}(\chi, w) - \langle v_\chi, z \rangle)^2 \end{aligned}$$

[4]In general the proof can be carried through if the fitting has errors which are small and random.

where $\text{PMI}(\chi, w)$ is defined according to the co-occurrence $\Pr(\chi, w)$ and $\Pr(w)$.

Furthermore, since w_1 and w_2 are unrelated, and the co-occurence matrix is sparse, the set of words χ that appear with *both* w_1, w_2 in the corpus is far smaller than the set of words that appear with *exactly one* of them. (Indeed, if nonzero entries constitute a p fraction of some matrix, then in two randomly picked rows, one expects only p^2 fraction of the entries to be nonzero in both.) Accordingly we make the following simplifcation.

Simplification 2: *In solving the optimization, restrict attention to words χ that co-occur with only one of w_1, w_2 in the corpus, not both.*

As a consequence, if T_1 is the set of words such that $\Pr(\chi, w_1) \neq 0$, and T_2 the set of words such that $\Pr(\chi, w_2) \neq 0$, $T_1 \cap T_2$ is expected to be of negligible size compared to T_1 and T_2. Thus, for $\chi \in T_1$, we have that $\Pr(\chi, w) = \Pr(\chi, w_1)$, and $\text{PMI}(\chi, w) = \text{PMI}(\chi, w_1) - \log(1 + 1/r) = \text{PMI}(\chi, w_1) - \kappa_1$. Similarly for $\chi \in T_2$, we have $\text{PMI}(\chi, w) = \text{PMI}(\chi, w_2) - \kappa_2$.

Towards establishing the main claim we first simplify $f(z)$ using the assumptions that no word belongs to T_1 and T_2 to

$$f(z) = \underbrace{\sum_{\chi \in T_1} \Pr(\chi, w_1) \cdot (\text{PMI}(\chi, w_1) - \langle \chi, z \rangle - \kappa_1)^2}_{f_1(z)}$$
$$+ \underbrace{\sum_{\chi \in T_2} \Pr(\chi, w_2) \cdot (\text{PMI}(\chi, w_2) - \langle \chi, z \rangle - \kappa_2)^2}_{f_2(z)}.$$

Simplification 1 implies $\text{PMI}(\chi, w_1) = \langle v_\chi, w_1 \rangle$ and $\text{PMI}(\chi, w_2) = \langle v_\chi, w_2 \rangle$. Thus if shifts κ_1 and κ_2 were zero, f_1 and f_2 would be minimized at $z = v_1$ and $z = v_2$ respectively by definition. Our argument consists of two parts. First we show that even with the non-zero constant shift κ_1 and κ_2:

$$\begin{aligned} \text{argmin } f_1(z) &\approx (1 - \kappa_1 c_1) v_1 \\ \text{argmin } f_2(z) &\approx (1 - \kappa_2 c_2) v_2 \end{aligned} \tag{5}$$

where c_1 and c_2 are small constants (to be estimated later). That is, the minimizers of f_1 and f_2 are only changed by a scaling due to the presence of the constant shift κ_1 and κ_2. Second, we will show

$$\text{argmin } f(z) \approx \text{argmin } f_1(z) + \text{argmin } f_2(z) \tag{6}$$

From (5) and (6), MAIN CLAIM follows.

To prove (5) we analyze the structure of $f_1(z)$ and $f_2(z)$. Replacing $\text{PMI}(\chi, w_1)$ by $\langle v_\chi, v_1 \rangle$ we obtain:

$$\begin{aligned} f_1(z) &= \sum_{\chi \in T_1} \Pr(\chi, w_1)(\langle v_\chi, v_1 - z \rangle - \kappa_1)^2 \\ &= (v_1 - z)^\top \Sigma_1 (v_1 - z) + \sum_{\chi \in T_1} \Pr(\chi, w_1) v_\chi^\top (v_1 - z) + C \end{aligned}$$

where C is a constant that doesn't depend on z, and Σ_1 is the matrix $\sum_{\chi \in T_1} \Pr(\chi, w_1) v_\chi v_\chi^T$, the weighted covariance matrix of the word vectors in T_1. Another simplification lets us gets a handle on it.

Simplification 3: *Σ_1, the matrix defined above, has the form $\gamma_1 v_1 v_1^T + \tau_1 I$. Furthermore, $\sum_{\chi \in T_1} \Pr(\chi, w) R v_\chi \approx b_1 e_1$ for a scalar b_1.*

To justify this simplification, note that Arora et al. (2016) shows evidence (contrary to prior intuition) that word vectors are distributed fairly uniformly in space, and in the bulk have many properties similar to random Gaussian vectors. Since words that cooccur with w_1 have (by the PMI model) nontrivial inner product with v_1, the word vectors in T_1 can be expected to behave like $\rho v_1 + \xi$ *where ρ is a scalar and ξ is a spherical Gaussian vector with comparable norm to that of ρv_1*. Indeed, Simplification 3 follows if all the v_χ in T_1 are distributed as $\rho v_1 + \xi$ with scalar ρ and spherical Gaussian random variable ξ such that ξ has expected norm on the same order as ρv_1, with τ_1 roughly γ_1/d. Moreover, we can compute the value of γ_1 (which will be useful later) : we have $v_1^\top \left(\sum_{\chi\in T_1} \Pr(\chi, w_1) v_\chi v_\chi^T\right) v_1 = v_1^\top \Sigma_1 v_1 = \gamma_1 \|v_1\|^4 + \tau_1 \|v_1\|^2$. This (together with $\tau_1 \ll \gamma_1$ and $\|v_1\|$ being roughly a constant) leads to

$$\gamma_1 = \sum_{\chi\in T_1} \Pr(\chi, w_1)\langle \chi, v_1\rangle^2 / \|v_1\|^4 \tag{7}$$

Here the math is a bit cumbersome since $\|v_1\|$ has not on the order of 1 but is not necessarily a unit vector. For simplicity we suggest the reader to pretend $\|v_1\| = 1$ for the rest of the proof since this case captures the essences.

Having justified Simplification 3 let us return to the proof. The quadratic form Σ_1 specifies can be better understood after picking an orthogonal basis in which it is diagonalized. Let R be the matrix that satisfies $Rv_1 = e_1$ and $Rv_2 = e_2$.[5] Letting $\tilde{z} = Rz$,

$$f_1(z) = (e_1 - \tilde{z})^T \Lambda (e_1 - \tilde{z}) + 2\kappa_1 t_1^T (e_1 - \tilde{z}) + C$$

with C, C_1 constants, $t_1 = \sum_{\chi\in T_1} \Pr(\chi, w) R v_\chi \approx b_1 e_1$ with b_1 being a scalar because of Simplification 3 and $\Lambda = R^{-T}\Sigma_1 R^{-1}$ is a diagonal matrix. The diagonal entry of Λ can be computed by evaluating $e_j^\top \Lambda e_j$, and it turns out that $\Lambda = \mathrm{diag}(\gamma_1\|v_1\|^4, \dots)$ where the entries except the first one are bounded by τ_1. By a slightly abuse of notation, we use τ_1 for the other diagonal entries of Λ since using an upper bound don't change the proof. Therefore, explicitly, after diagonalization, the objective becomes

$$f_1(z) = g_1(\tilde{z}) = \gamma_1(1 - \tilde{z}_1)^2 + 2\kappa_1 b_1 \tilde{z}_1 + \tau_1 \cdot \sum_{j\neq 1} (\tilde{z}_j - t_{1,j})^2.$$

Note that t_1 is very close to e_1 and therefore $t_{1,j} \approx 0$ for $j \neq 1$ and $t_1' \approx t_1\kappa_1/\gamma_1 = \kappa_1 b_1/\gamma_1 \cdot e_1$. Thus the minimizer $\tilde{z}^\star$ of $g_1(\tilde{z})$ should satisfy $\tilde{z}_j^\star = t_{1,j} \approx 0$ for $j \neq 1$, and the value of $\tilde{z}_1^\star$ is the minimizer of the univariate quadratic function $\gamma_1(1 - \tilde{z}_1)^2 + 2\kappa_1 b_1 \tilde{z}_1$. By elementary calculation we get that $\tilde{z}_1^\star = 1 - c_1\kappa_1$ with $c_1 = b_1/\gamma_1$. That is, $\tilde{z}^\star \approx (1 - c_1\kappa_1)e_1$.

Since $\tilde{z} = Rz$, the minimizer of $f_1(z)$ is equal to $R^{-1}\tilde{z}^\star$. Thus, we have

$$\mathrm{argmin}\ f_1(z) = R^{-1}\tilde{z}^\star \approx (1 - c_1\kappa_1)v_1.$$

To estimate c_1, we first compute b_1. By definition, $t_1 \approx b_1 e_1$, and therefore,

$$b_1 \approx \langle t_1, e_1\rangle = \sum_{\chi\in T_1} \Pr(\chi, w)\langle R v_\chi, e_1\rangle = \sum_{\chi\in T_1} \Pr(\chi, w)\langle v_1, v_\chi\rangle.$$

Plugging in the estimate for γ_1 in (7) we obtain $c_1 \approx \frac{\sum_{\chi\in T_1} \Pr(\chi, v_1)\langle \chi, v_1\rangle}{\sum_{\chi\in T_1} \Pr(\chi, v_1)\langle \chi, v_1\rangle^2}$. Therefore c_1 is expected to be on the order of a constant since $\|v_1\|$ and $\langle \chi, v_1\rangle$ are all on the order of 1. Thus we have established the first equation in (5). The second one can proved analogously.

What remains is to show (6). The additive structure of (6) is obviously false in general, but holds in this case because f_1 and f_2 interact with each other in a rather limited way. On a high level, $f_1(z)$ turns out to be only sensitive to the component of z on v_1, and f_2 only to the component of z on v_2. Hence the minimizer of $f_1 + f_2$ is achieved when z has the right projection to v_1 and v_2. Formally, $f(z)$ can be simplified as

$$\begin{aligned} &f(z) = f_1(z) + f_2(z) = g_1(\tilde{z}) + g_2(\tilde{z}) = \\ &\gamma_1(1 - \kappa_1 c_1 - \tilde{z}_1)^2 + \gamma_2(1 - \kappa_2 c_2 - \tilde{z}_2)^2 + \tau_1(t'_{2,j} - \tilde{z}_2) \\ &+ \tau_2(t'_{2,j} - \tilde{z}_1) + \tau_1 \cdot \mathrm{rest}_1 + \tau_2 \cdot \mathrm{rest}_2 \end{aligned} \tag{8}$$

[5]Effectively, R is a change-of-basis matrix and has columns that are orthogonal, but which doesn't necessarily have norm 1, since v_1 and v_2 are not necessarily of norm 1.

where $\text{rest}_\ell = \sum_{j\neq 1,2}(\tilde{z}_j - t'_{\ell,j})$, for $\ell = 1, 2$. Therefore we see that the presence of rest_ℓ in the minimization objective forces $\tilde{z}_3, \dots, \tilde{z}_d$ to be close to either $t'_{1,j}$ or $t'_{2,j}$, both of which are small.[6] For coordinate $\tilde{z}_1$ and $\tilde{z}_2$, when $\tau_2 \ll \gamma_1$ and $\tau_1 \ll \gamma_2$, the minimizer of (8) is approximately $\tilde{z}_1 = 1 - \kappa_1 c_1$ and $\tilde{z}_2 = 1 - \kappa_2 c_2$. Therefore, we obtain that the minimizer of $g_1(\tilde{z}) + g_2(\tilde{z})$ is approximately $\tilde{z} = (1 - \kappa_1 c_1)e_1 + (1 - \kappa_2 c_2)e_2$, and consequently,

$$\text{argmin}_z f(z) \approx (1 - \kappa_1 c_1)v_1 + (1 - \kappa_2 c_2)v_2$$

Finally, we argue that τ_1 is indeed expected to be (much) smaller than γ_2. As discussed above, τ_1 is approximately γ_1/d and therefore when $\gamma_1/\gamma_2 < d$, which further translates to roughly $\Pr(w_1)/\Pr(w_2) < d$, we will have $\tau_1 < \gamma_2$. Moreover, in the case when τ_1 is comparable to γ_2 or much larger, we can still analytically solve the minimizer of (8), and obtain that $\tilde{z}_1 = \frac{\gamma_1(1-\kappa_1 c_1)+\tau_2 t'_{2,j}}{\gamma_1+\tau_2}$ and $\tilde{z}_2 = \frac{\gamma_2(1-\kappa_2 c_2)+\tau_1 t'_{1,j}}{\gamma_2+\tau_1}$, and consequently, $v \approx \tilde{z}_1 v_1 + \tilde{z}_2 v_2$.

6 Conclusions

Word embeddings obtained from (Arora et al., 2016)—and also several earlier methods—have been shown to contain information about the different senses of the word that is extractable via simple sparse coding. Currently it seems to do better with nouns than other parts of speech, and improving this is left for future work. One novel aspect of our approach is that the word senses are interrelated using one of about 2000 discourse vectors that give a succinct description of which other words appear in the neighborhood with that sense. This makes the method potentially more useful for other tasks in Natural Language Processing, as well as automated creation of WordNets in other languages. The method can be more useful —and accurate—in a semi-automated mode, since human helpers are better at *recognizing* a presented word sense than at coming up with a complete list. Thus instead of listing only 5 senses per word as given by the sparse coding, the algorithm can ask a human helper to examine the list of 20 closest discourse atoms (and sample sentences as in Section 3.3) which usually gives high recall for senses missing in the top 5. This semi-supervised version seems promising as a fast way to create WordNets for other languages.

As mentioned in Section 2, the use of logarithm in these new embedding methods as well as the old PMI method seems key to the success of our approach, because the logarithm allows less frequent senses to have a superproportionate weight in the linear superposition. This may have relevance of neuroscience, where word embeddings have been used in fMRI studies to map what the subject *is thinking about* (Mitchell et al., 2008). The words in that study were largely monosemous, and the more nuanced word embeddings and discourse vectors introduced in this work may be useful in further explorations, especially since sparse coding as well as logarithms are thought to be neurally plausible.

References

Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *Transaction of Association for Computational Linguistics*, 2016.

David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, Reykjavk, Icelandik, Iceland, May 2014.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, pages 288–296, 2009.

[6] $[t_{1,3}, \dots, t_{1,d}]$ was proved to be negligible in ℓ_2 norm.

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

Ivan Damnjanovic, Matthew EP Davies, and Mark D Plumbley. Smallbox-an evaluation framework for sparse representations and dictionary learning algorithms. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 418–425. Springer, 2010.

Antonio Di Marco and Roberto Navigli. Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics*, 39(3):709–754, 2013.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 144197010X, 9781441970107.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, 2015a.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. Sparse overcomplete word vector representations. In *Proceedings of ACL*, 2015b.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

John Rupert Firth. A synopsis of linguistic theory. In *Studies in Linguistic Analysis*. Philological Society, Oxford, 1957.

DMBTL Griffiths and MIJJB Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in Neural Information Processing Systems*, 2004.

Thomas L Griffiths, Mark Steyvers, and Joshua B Tenenbaum. Topics in semantic representation. *Psychological review*, 114(2):211, 2007.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 873–882, 2012.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. Sensembed: learning sense embeddings for word and relational similarity. In *Proceedings of Association for Computational Linguistics*, pages 95–105, 2015.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

Matt Mahoney. Wikipedia text preprocess script. http://mattmahoney.net/dc/textdata.html, 2008. Accessed Mar-2015.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014. URL `http://www.aclweb.org/anthology/P/P14/P14-5010`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013a.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013b.

Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. Predicting human brain activity associated with the meanings of nouns. *Science*, 320(5880):1191–1195, 2008.

Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine learning*, pages 641–648. ACM, 2007.

Brian Murphy, Partha Pratim Talukdar, and Tom M. Mitchell. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of the 24th International Conference on Computational Linguistics*, 2012.

Roberto Navigli and Daniele Vannella. Semeval-2013 task 11: Word sense induction and disambiguation within an end-user application. In *Second Joint Conference on Lexical and Computational Semantics (SEM)*, volume 2, pages 193–201, 2013.

Bruno Olshausen and David Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):33113325, 1997.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing*, 12, 2014.

Joseph Reisinger and Raymond Mooney. Multi-prototype vector-space models of word meaning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2010.

Hinrich Schutze. Automatic word sense discrimination. *Computational Linguistics*, 21(1):97123, 1998.

Peter D Turney, Patrick Pantel, et al. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010.

Daniele Vannella, Tiziano Flati, and Roberto Navigli. Wosit: A word sense induction toolkit for search result clustering and diversification. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 67–72, 2014.

Wikimedia. English Wikipedia dump. http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2, 2012. Accessed Mar-2015.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, pages 189–196, 1995.

A Word Vector Training Method

The data set used was the English Wikipedia (Wikimedia, 2012). The data was preprocessed by a standard approach (Mahoney, 2008), including removing non-textual elements, sentence splitting, and tokenization, yielding a corpus with about 3 billion tokens. Words that appeared less than 1000 times in the corpus were ignored, leaving a vocabulary of size 68430. The co-occurrences were then computed within a window size of 10. The 300 dimensional word vectors were trained by optimizing the objective function **SN** derived in (Arora et al., 2016):

$$\min_{\{v_w\},C} \sum_{w,w'} X_{w,w'} \left(\log(X_{w,w'}) - \|v_w + v_{w'}\|_2^2 - C\right)^2 \tag{9}$$

where v_w is the vector for the word w and $X_{w,w'}$ is the co-occurrence of the words w, w'. The optimization method was AdaGrad (Duchi et al., 2011).

This objective is closely related to the old PMI model of Church and Hanks (1990).

B Sparse Coding Method

We used the well-known k-SVD algorithm (Aharon et al., 2006). The basis were initialized with randomly picked word vectors, and then the representation and the basis were alternatingly updated; see (Elad, 2010; Damnjanovic et al., 2010). The basis size was set to be 2000, and the sparsity was set to 5, which means that each word vector is approximated by a linear combination of at most 5 atoms.

The above algorithm was run 5 times with different random initializations. The underlying theory suggests that a set of meaningful atoms should be stable across different runs. This is indeed empirically observed: around 2/3 of the atoms in one basis had atoms in the other bases, which had inner product larger than 0.85. An inner product larger than 0.85 means that the two atoms are near-duplicates: for example, the 10 nearest words will tend to be the same. Therefore, a basis with stable atoms were obtained by merging the bases obtained in different runs, i.e. removing the duplicates and removing the unstable atoms (more precisely, those atoms that did not have a neighbor with inner product larger than 0.2 in other bases). The resulting basis had 2376 atoms.

About 10% of atoms were found to be semantically meaningless. Some corresponded to systematic bias in the word vectors or the corpus; a telltale sign is that they were used in the representations of a large number of words. The 25 most popular atoms (in terms of how many words pick them in their representations) were removed. Some other atoms contained user names of many Wikipedia editors or last names of people whose first name is “Albert”. Words near such atoms are unlikely to have large inner products with each other. By checking such a condition, such atoms were identified and removed. The details are similar to those under the heading “outputting coherent word groups.”At the end, 2095 atoms were left.

Each word vector is then represented as a linear combination of at most five atoms, where the representation is computed by the same k-SVD algorithm. As an example, Table 4 shows the atoms linked to the words “chair” and “bank”.

C Experiments with Atoms of Discourse

C.1 Hierarchy of discourse atoms

Here are two more examples for meta-atoms. One example consists of meta-atoms for transportation, racing, and sports, and is shown in Figure 4 . Note that bike racing and general racing both belong to racing and sports. The other example consists of the food meta-atom, and is shown in Figure 5.

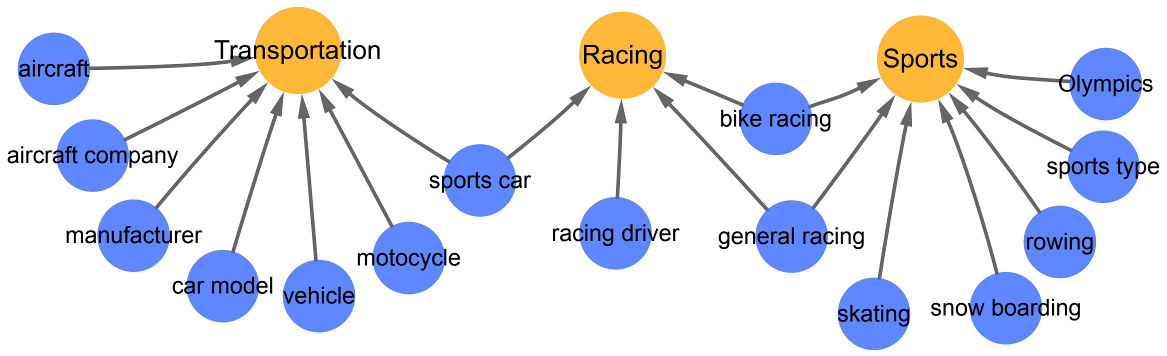
C.2 Outputting relevant sentences

First, for each sentence in the corpus, define its semantic representation to be the best rank-3 approximation to the subspace spanned by the word embeddings of its component words (filtering out frequent words like

Atom 1187	739	590	1322	1457
sitting	committee	graduating	urology	protesters
seated	chaired	studied	neurology	demonstrators
standing	chairing	graduated	obstetrics	crowds
sits	consultative	doctorate	gynecology	protestors
beside	committees	graduate	ophthalmology	gathered
sit	convened	studying	neurosurgery	crowd

Atom 599	1308	209	1653	1050
bank	believe	river	masjid	hermetic
hsbc	own	tributaries	chowk	occult
banks	why	tributary	bazar	freemasonry
citibank	actual	rivers	moti	aleister
banking	understand	watershed	bagh	thelema
banco	working	flows	mahal	esoteric

Table 4: Five discourse atoms linked to the words "chair" and "bank" by our method. Each atom is represented by and its nearest 6 words. The algorithm often makes a mistake in the last atom (or two), as happened here.

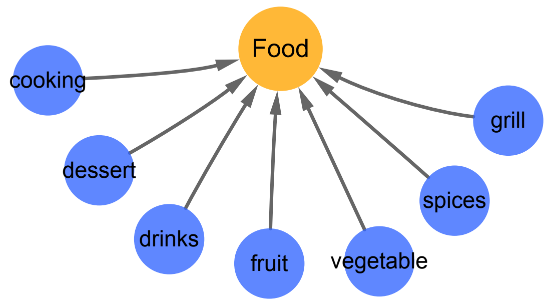


Atom	722	407	1164	1623	114	1932	633
	helicopter	sedan	sportscar	race	giro	skating	cheerleading
	helicopters	hatchback	drivers	podiums	vuelta	skater	softball
	sikorsky	sedans	motorsport	races	uci	skaters	volleyball
	aircraft	hardtop	racing	laps	ttt	isu	frisbee
tag	*aircraft*	*car model*	*sports car*	*general racing*	*bike racing*	*skating*	*sports type*

Figure 4: Meta-atoms for transportation, racing, and sports. The figure shows the atoms that uses them, where atoms/meta-atoms are tagged manually. The table shows the top 4 nearest words and tags for some atoms.

is, the that carry little meaning), where the best rank-3 approximation can be computed by PCA (Principal Component Analysis). The similarity between a sentence and an atom is then defined as the cosine similarity, that is, the length of the projection of the atom onto the semantic representation of the sentence. Since there may be bias in the vectors of the words in the sentence, the relevance between an atom a and a sentence s should be normalized:

$$\text{rel}(a, s) = \|Sa\|_2 - \sum_{a' \in A} \|Sa'\|_2 / |A|,$$



Atom	6	1273	128	1549	959	1109	1280
	cooked fried dish boiled	cakes dessert cake pastries	drinks drink beverage beverages	banana coconut beans sugarcane	onions cabbage garlic spinach	garlic spices coriander sauce	grilled beef pork cooked
tag	*cooking*	*dessert*	*drinks*	*fruit*	*vegetable*	*spices*	*grill*

Figure 5: Meta-atom for food. The figure shows the atoms that uses them, where atoms/meta-atoms are tagged manually. The table shows the top 4 nearest words and tags for some atoms.

where S is the semantic representation of the sentence s presented as a matrix of 3 rows, and A is the set of all atoms.

To find relevant sentences for a polysemous word w, first compute the atoms for the word, and also the atoms for its inflectional forms of w, such as the past tense of a verb. These inflectional forms are found by the Stanford NLP tool (Manning et al., 2014). For each atom, compute its relevance with each sentence that contains w and contains more than 10 words, and find the sentence with the highest relevance. If two atoms are close to each other, only one sentence needs to be output for them. So the atoms are clustered as follows: build a graph on the atoms by adding an edge between any two atoms with inner product larger than 0.6; let each connected component of the graph be an cluster. For each cluster, sort the sentences associated with the atoms in the cluster, and keep only the sentence with highest relevance. The top 7 sentences are output; if there are less than 7, output all the sentences.

D A Quantitative Test

The testbed was constructed by graduate students, who were instructed to pick typical polysemous words and their meanings, spot uses of each word meaning in sentences in WordNet and other online dictionaries, and select 8 related words from those sentences that can represent a "discourse" capturing the word meaning. The final testbed consists of 200 polysemous words and their 704 meanings. The number of meanings for one word ranges from 2 to 14, and on average is about 3.5.

The test is then as follows: given a polysemous word w and a set of m meanings which contain the true ones for the word as well as some distractors (randomly picked meanings), identify the word's true meanings.

The algorithm for the test begins with finding the atoms for the given polysemous word w and removing atoms with too small coefficients. Since more frequent words tend to have more meanings, atoms with coefficients below $0.1 + 0.2(\text{rank}(w)/N)$ are removed, where $\text{rank}(w)$ is the frequency rank of w in the vocabulary, and N is the size of the vocabulary. So the threshold is about 0.1 for the most frequent word and 0.3 for the least frequent. The resulting set of atoms are further augmented with the top atom of each inflectional forms of the word, such as the past tense of a verb. These inflectional forms are found by Stanford

	sentence
1	1986's *No. 10, Upping St.* reunited Jones for one album with former Clash band-mate Joe Strummer, who was a co-producer of the album and co-writer of a numberof its songs.
2	Band (radio), a range of frequencies or wavelengths used in radio transmission and radar.
3	...... acclaimed for their first three albums. The band has also been important for the advocacy of medical and recreational use of cannabis in the United States.
4	Three equally sized horizontal bands of blue, red, and green, with a white crescent and an eight-pointed star centered in the red band.
5	Gel electrophoresis of the plasmids would normally show the negatively supercoiled form as the main band, while nicked DNA (open circular form) and the relaxed closed circular form appears as minor bands.
6	ZigBee - low power lightweight wireless protocol in the ISM band.
7	At the height of their popularity, the band's success spurred a host of cult-like activities.

Table 5: Relevant fragments from top 7 sentences identified for the word "band."

	sentence
1	For a sine wave modulation, the modulation index is seen to be the ratio of the amplitude of the modulating sine wave to the amplitude of the carrier wave.
2	They experienced the emergence of music videos, new wave music, electronic music, synthpop, glam rock, heavy metal and the spin-off glam metal, punk rock and the spin-off pop punk, alternative rock, grunge, and hip hop.
3	Earthquakes, along with severe storms, volcanic activity, coastal wave attack, and wildfires, can produce slope instability leading to landslides, a major geological hazard.
4	sinusoidal plane-wave solutions of the electromagnetic wave equation
5	Slide cards under chips (in handheld games); wave hand horizontally (in games dealt face up).
6	The Australian flag shall triumphantly wave in the sunshine of its own blue and peerless sky, over thousands of Australia's adopted sons.
7	Kristin Hersh, American singer-songwriter and guitarist(Throwing Muses and 50 Foot Wave)

Table 6: Relevant fragments from top 7 sentences identified for the word "wave."

bank					chair		
bank1	*bank2*	*bank3*	*bank4*	*bank5*	*chair1*	*chair2*	*chair3*
water	institution	arrangement	flight	faith	one	professor	officer
land	deposits	similar	aircraft	confidence	person	award	address
sloping	money	objects	tip	support	support	honor	meetings
river	lending	tiers	turning	help	back	recognition	organization
hill	custody	group	flying	rely	table	titled	lead
ravine	loan	series	tilt	consider	sit	endowed	first
canyon	money	together	sidewise	believe	room	university	primary
acclivity	funds	arrange	rounding	safe	wooden	college	charge

Table 7: Two examples from the polysemy testbed.

	native speaker					non-native speaker				
precision	0.88	0.80	0.84	0.80	0.95	0.51	0.72	0.94	0.72	0.81
recall	0.79	0.66	0.79	0.83	0.87	0.33	0.56	0.58	0.53	0.68

Table 8: Human performance on the polysemy testbed.

NLP tool (Manning et al., 2014).

For each atom a, the algorithm scores each meaning (represented by a list L of words) by the similarity between the meaning and the atom. First, the meaning should be matched to the polysemous word under the discourse represented by the atom, so the word vector should also be considered. Second, since there may be bias in the vectors for L, the score should be normalized. So the score is defined as

$$\text{score}(a, L) = \|Ua\|_2 - \sum_{a' \in A} \|Ua'\|_2/|A| + \|Uu\|_2 - \sum_{w' \in V} \|Uv_{w'}\|_2/|V|,$$

where U is a matrix whose rows are vectors for words in L, u is the vector for the word associated with the atom a (the word w or its inflectional forms), A is the set of all atoms, and V is the vocabulary. The top two meanings with highest scores for each atom are found. Take the union of all the meanings found for all the atoms, and if a meaning is found by multiple atoms, accumulate its scores. Finally, the top k ones in the union with highest scores are returned; if the union has less than k meanings, return all the meanings.

A formal description of the algorithm is presented in Algorithm 1.

D.1 Comparison with human performance

We also asked 10 graduate students in science or engineering fields[7] to evaluate the testbed. The testbed was randomly partitioned into 10 sub-testbeds, each consisting of 20 polysemous words. Each student was given one sub-testbed, and was asked to identify for each polysemous word the true meanings among 20 ones that contains the true ones and some random distractors. They were told that the number of true meanings for one word is between 2 and 14, and is 3.3 on average, but the numbers of true meanings for individual words were unrevealed.

The results are shown in Table 8. Native speakers achieved on average 85% precision and 79% recall, while non-native speakers achieved 74% precision and 54% recall. Note that there are large variances, especially for non-native speakers. Our algorithm with $k = 2$ achieved 76% precision and 50% recall, which is similar to the average performance of non-native speakers. There results are also plotted in the main text.

D.2 Comparison to graph clustering based WSI algorithms

Tests were performed using the WSI algorithms in the recent paper (Di Marco and Navigli, 2013). The code (Vannella et al., 2014) we used was released by the same set of authors. In their test, given a target polysemous word, these methods first construct a co-occurrence graph on words related to the target word, and then use some clustering algorithm on the graph to produce several clusters of words, which are regarded as the senses and were used for grouping web search results returned when searching for the target word.

The code was adapted to our testbed as follows. First, we constructed the co-occurrence graph for the target word, using the algorithm detailed in (Di Marco and Navigli, 2013). Since the corpus we used (English Wikipedia) is different from the corpus in their paper, we also did hyperparameter tuning; see the paragraph below. Second, we used the graph clustering based WSI algorithms provided in (Vannella et al., 2014) to get several clusters of words. Three clustering algorithms (ChineseWhispers, BMST, and SquaT) were tested. Finally, these clusters are used to pass the police lineup test. Given the lineup of m senses (each a list of 8 words), we matched each sense with each WSI cluster, set the score of a sense to be its best matching metric

[7]These students were different from those who constructed the testbed. Non-native speakers have 7 to 10 years of English language use/learning.

COMMON	Atom 2	365	402	754	1140	1766
	ferrari	regulations	preaching	variational	importers	warmachine
	lamborghini	stricter	christ	deconvolution	exporter	warhammer
	supercar	stringent	gospel	nonlinear	wholesaler	khador
	roadster	enforce	bible	regularization	exporters	everblight
GLOVE	Atom 362	479	707	933	937	1246
	yale	advocating	species	instagram	hyperbola	timezone
	harvard	activism	endemic	twitter	parabola	dst
	swarthmore	advocates	genus	facebook	tangent	cest
	princeton	advocacy	subspecies	tumblr	parallelogram	cet
CBOW	Atom 3	269	542	1638	2802	3083
	protestantism	cheeks	probabilistic	knife	wheel	hegemony
	catholicism	forehead	heuristic	blade	lever	colonialism
	other_religions	lips	empirical	spear	spindle	imperialism
	christianity	buttocks	theory	claw	spinning	republics
NNSE	Atom 67	147	194	970	1464	1739
	chemicals	exporters	graphics	toe	amounts	notify
	substances	businessmen	landscapes	hem	amount	inform
	pesticides	merchants	visuals	heel	chunk	assist
	solvents	artisans	portraits	cuff	quantities	instruct

Table 9: Examples of discourse atoms and their nearest 4 words for different semantic vectors.

across all clusters, and then return the top k senses with the highest scores. We used two matching metrics {WORD_OVERLAP, DEGREE_OVERLAP} provided in (Di Marco and Navigli, 2013).

The hyperparameters of constructing the co-occurrence graphs are enumerated among 4^{α} times the default values in the code for $\alpha \in \{-2, -1, 0, 1, 2\}$. For each of the three clustering algorithms, the best results among all sets of hyperparameters and the two matching metrics are reported in Figure 3.

D.3 Performance of some other types of semantic vectors

Our approach can also be applied to other types of semantic vectors, including those computed on other data sets, and those computed by other methods. Here we provide results for four other types of semantic vectors:

- COMMON: computed using the objective function **SN** on a subset of a large data set called common crawl (Buck et al., 2014). This subset contains about 50 billion tokens.
- GLOVE: computed using the approach in (Pennington et al., 2014) on the English Wikipedia data.
- CBOW: computed using the approach in (Mikolov et al., 2013a) on the English Wikipedia data.
- NNSE: precomputed vectors by (Murphy et al., 2012).

These also yield meaningful atoms, some examples of which are shown in Table S6. Of course, it should be not surprising that the technique applies to these methods, since recent work (Levy and Goldberg, 2014; Arora et al., 2016) suggests they are quite related.

The word embeddings from these methods their corresponding atoms are also evaluated on the polysemy testbed, and the performances are plotted in Figure 6. GLOVE vectors achieve similar results to ours. COMMON and CBOW have slight worse performance, and NNSE has the worst.

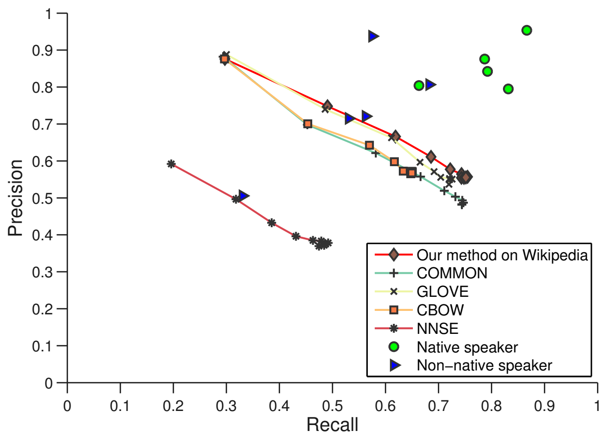


Figure 6: Precision and recall in the polysemy test. For each polysemous word, a set of m meanings are presented, and the algorithm returns k meanings. The figure plots results of different methods for $m = 20$, $k = 1, 2, \dots, 10$.

chair	seat sitting stand
	advisory chairman committee subcommittee
	faculty graduated lectures phd professor
	committee consultant convene
	bench curtain desk drawer fireplace furniture room shelves
colon	apostrophe comma hyphen punctuation quotation semicolon
	bladder intestinal kidney liver lung pancreas spleen stomach
	frac ldots mapsto mathbb mathbf mathrm rightarrow varphi
	apostrophe comma dash hyphen semicolon
coordinate	angle angular axis directed perpendicular rotation
	cartography geodesy geodetic geospatial map survey
	attempt efforts help integrated organization try
	efforts initially project spearheaded undertaken
	affairs executive governmental oversee response
	latd latm latns lats long longd longew longm

Table 10: Examples of coherent word groups.

E Outputting Coherent Word Groups

A salient feature of WordNet is computing *synsets* for each word sense: list of synonyms. Here we mention a technique to output semantically coherent word groups. These are not synsets per se, but in a semisupervised setting (with a human in the loop) they might be useful to construct synsets.

The idea is to produce a list of words $L_{(a,w)}$ that forms a "dense" cluster located "near" the atom a for atom-word pairs (a, w). Doing this for all atoms in the representation of a word and its inflectional forms can be viewed as producing relevant words for the meanings of the word.

"Dense" and "near" are quantified as follows: the list of words $L_{(a,w)}$ is τ-dense if for all words w', the 30-th percentile of the normalized inner products of w' with the rest of the words in $L_{(a,w)}$ is at least τ. $L_{(a,w)}$ is d-near a, if all the words in $L_{(a,w)}$ have normalized inner product at least d. Given fixed τ and d, the algorithm greedily builds the cluster of words $L_{(a,w)}$ by iteratively adding the vertex which maximizes a weighted sum of the 30-th percentile of the inner product to the vertices already in $L_{(a,w)}$ and the the inner product with a. (Precisely, when adding the i-th word, maximize over all words w': $\tau_{w'} + 0.5 \cdot (1 + 4/i) \cdot d_{w'}$, where $\tau_{w'}$ is the 30th percentile of the inner product of w' to the vertices already in $L_{(a,w)}$ and $d_{w'}$ the inner product of w' with a.) The algorithm stops when the vertex added results in violating the density or nearness parameters. The lists are further capped to be no longer than 8 words. τ is initially set to 0.45 and d to 0.5, and gradually decreased until a word list of length at least 3 is constructed. Table 10 shows the results for 3 words (we take as pairs the word itself, along with the top 3 atoms in the representation, as well as the top atom in the representation of the inflected forms).

It is observed that many of the words found are related to existing synsets in WordNet. When they are not, they could suggest new relations that WordNet happens not to encode but perhaps should. In other words, the method may point the way to a more densely connected network among the words, which would be more valuable for NLP applications requiring word sense discrimination and disambiguation.