Word2Sense : Sparse Interpretable Word Embeddings

Abhishek Panigrahi
Microsoft Research India
t-abpani@microsoft.com

Harsha Vardhan Simhadri
Microsoft Research India
harshasi@microsoft.com

Chiranjib Bhattacharyya
Microsoft Research India, and
Indian Institute of Science
chiru@iisc.ac.in

Abstract

We present an unsupervised method to generate **Word2Sense** word embeddings that are interpretable — each dimension of the embedding space corresponds to a fine-grained *sense*, and the non-negative value of the embedding along the j-th dimension represents the relevance of the j-th sense to the word. The underlying LDA-based generative model can be extended to refine the representation of a polysemous word in a short context, allowing us to use the embeddings in contextual tasks. On computational NLP tasks, Word2Sense embeddings compare well with other word embeddings generated by unsupervised methods. Across tasks such as word similarity, entailment, sense induction, and contextual interpretation, Word2Sense is competitive with the state-of-the-art method for that task. Word2Sense embeddings are at least as sparse and fast to compute as prior art.

1 Introduction

Several unsupervised methods such as $SkipGram$ (Mikolov et al., 2013) and $Glove$ (Pennington et al., 2014) have demonstrated that co-occurrence data from large corpora can be used to compute low-dimensional representations of words (a.k.a. embeddings) that are useful in computational NLP tasks. While not as accurate as semi-supervised methods such as $BERT$ (Devlin et al., 2018) and $ELMO$ (Peters et al., 2018) that are trained on various downstream tasks, they do not require massive amounts of compute unaccessible to all but few.

Nearly all such methods produce dense representations for words whose coordinates in themselves have no meaningful interpretation. The numerical values of a word's embedding are meaningful only in relation to representations of other words. A unitary rotation can be applied to many of these embeddings retaining their utility for computational tasks, and yet completely changing the values of individual coordinates. Can we design an interpretable embedding whose coordinates have a clear meaning to humans?

Ideally such an embedding would capture the multiple *senses* of a word, while being effective at computational tasks that use inter-word spacing of embeddings. Loosely, a sense is a set of semantically similar words that collectively evoke a bigger picture than individual words in the reader's mind. In this work, we mathematically define a sense to be a probability distribution over the vocabulary, just as topics in topic models. A human can relate to a sense through the words with maximum probability in the sense's probability distribution. Table 1 presents the top 10 words for a few senses.

We describe precisely such an embedding of words in a space where each dimension corresponds to a sense. Words are represented as probability distributions over senses so that the magnitude of each coordinate represents the relative importance of the corresponding sense to the word. Such embeddings would naturally capture the polysemous nature of words. For instance, the embedding for a word such as *cell* with many senses – e.g. "biological entity", "mobile phones", "excel sheet", "blocks", "prison" and "battery" (see Table 1) – will have support over all such senses.

To recover senses from a corpus and to represent word embeddings as (sparse) probability distributions over senses, we propose a generative model (Figure 1) for the co-occurrence matrix: (1) associate with each word w a sense distribution θ_w with Dirichlet prior; (2) form a context around a target word w by sampling senses z according to θ_w, and sample words from the distribution of sense z. This allows us to use fast inference tools such as WarpLDA (Chen et al., 2016) to recover few thousand fine-grained senses from large cor-

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5692–5705
Florence, Italy, July 28 - August 2, 2019.

Word	Rank	Top 10 words with the highest probability in the sense's distribution
Tie	1	hitch, tying, magnet, tied, knots, tie, loops, rope, knot, loop
	2	shirts, wore, shoes, jacket, trousers, worn, shirt, dress, wearing, wear
	3	against, scored, round, 2-1, champions, match, finals, final, win, cup
Bat	1	species, myotis, roosts, pipistrelle, reservoir, roost, dam, horseshoe, bats, bat
	2	smuggling, smoked, cigars, smokers, cigar, smoke, smoking, cigarette, cigarettes, tobacco
	3	bowled, bowler, first-class, bowling, batsman, wicket, overs, innings, cricket, wickets
Apple	1	player, micro, zen, portable, shuffle, mini, nano, mp3, apple, ipod
	2	graphics, g4, pc, hardware, pci, macintosh, intel, os, apple, mac
	3	vegetables, lemon, grapes, citrus, orange, apple, apples, fruits, juice, fruit
Star	1	vulcan, archer, picard, enterprise, voyager, starship, spock, kirk, star trek
	2	obi-wan, luke, anakin, skywalker, sith, vader, darth, star, jedi, wars
	3	cluster, nebula, dwarf, magnitude, ngc, constellation, star, stars, galaxies, galaxy
	4	inn, guest, star, b&b, rooms, bed, accommodation, breakfast, hotels, hotel
Cell	1	plasma, cellular, membranes, molecular, cells, molecules, cell, protein, membrane, proteins
	2	kinase, immune, gene, activation, proteins, receptors, protein, receptor, cell, cells
	3	transfusion, renal, liver, donor, transplantation, bone, kidney, marrow, transplant, blood
	8	top, squares, stack, bottom, the, table, columns, row, column, rows
	10	inmate, correctional, workhouse, jail, prisoner, hmp, inmates, prisons, prisoners, prison
	12	aaa, powered, nimh, mains, lithium, aa, rechargeable, charger, batteries, battery
	15	handset, bluetooth, ericsson, ringtones, samsung, mobile, phones, phone, motorola, nokia

Table 1: Top senses of polysemous words as identified Word2Sense embeddings. Each row lists the rank of the sense in terms of its weight in the word's embedding, and the top 10 words in the senses' probability distribution.

pora and construct the embeddings.

$Word2Sense$ embeddings are extremely sparse despite residing in a higher dimensional space (few thousand), and the number of non-zeros in the embeddings is no more than 100. In comparison, $Word2vec$ performs best on most tasks when computed in 500 dimensions.

These sparse single prototype embeddings effectively capture the senses a word can take in the corpus, and can outperform probabilistic embeddings (Athiwaratkun and Wilson, 2017) at tasks such as word entailment, and compete with $Word2vec$ embeddings and multi-prototype embeddings (Neelakantan et al., 2015) in similarity and relatedness tasks.

Unlike prior work such as $Word2vec$ and GloVe, our generative model has a natural extension for disambiguating the senses of a polysemous word in a short context. This allows the refinement of the embedding of a polysemous word to a $WordCtx2Sense$ embedding that better reflects the senses of the word relevant in the context. This is useful for tasks such as Stanford contextual word similarity (Huang et al., 2012) and word sense induction (Manandhar et al., 2010).

Our methodology does not suffer from computational constraints unlike $Word2GM$ (Athiwaratkun and Wilson, 2017) and $MSSG$ (Neelakantan et al., 2015) which are constrained to learning 2-3 senses for a word. The key idea that gives us this advantage is that rather than constructing a per-word representation of senses, we construct a global pool of senses from which the senses a word takes in the corpus are inferred. Our methodology takes just 5 hours on one multicore processor to recover senses and embeddings from a concatenation of UKWAC (2.5B tokens) and Wackypedia (1B tokens) co-occurrence matrices (Baroni et al., 2009) with a vocabulary of 255434 words that occur at least 100 times.

Our major contributions include:

- A single prototype word embedding that encodes information about the senses a word takes in the training corpus in a human interpretable way. This embedding outperforms $Word2vec$ in rare word similarity task and word relatedness task and is within 2% in other similarity and relatedness tasks; and outperforms $Word2GM$ on the entailment task of (Baroni et al., 2012).
- A generative model that allows for disambiguating the sense of a polysemous word in a short context that outperforms the state-of-the-art unsupervised methods on Word Sense Induction for Semeval-2010 (Manandhar et al., 2010) and MakeSense-2016 (Mu et al., 2017) datasets and is within 1% of the best models for the contextual word similarity task of (Huang et al., 2012).

2 Related Work

Several unsupervised methods generate dense single prototype word embeddings. These include $Word2vec$ (Mikolov et al., 2013), which learns embeddings that maximize the cosine similarity of embeddings of co-occurring words, and $Glove$ (Pennington et al., 2014) and $Swivel$ (Shazeer et al., 2016) that learn embeddings by factorizing the word co-occurrence matrix. (Dhillon et al., 2015; Stratos et al., 2015) use canonical correlation analysis (CCA) to learn word embeddings that maximize correlation with context. (Levy and Goldberg, 2014; Levy et al., 2015) showed that SVD based methods can compete with neural embeddings. (Lebret and Collobert, 2013) use Hellinger PCA, and claim that Hellinger distance is a better metric than Euclidean distance in discrete probability space.

Multiple works have considered converting the existing embeddings to interpretable ones. Murphy et al. (2012) use non-negative matrix factorization of the word-word co-occurrence matrix to derive interpretable word embeddings. (Sun et al., 2016; Han et al., 2012) change the loss function in Glove to incorporate sparsity and non negativity respectively to capture interpretability. (Faruqui et al., 2015) propose Sparse Overcomplete Word Vectors ($SPOWV$), by solving an optimization problem in dictionary learning setting to produce sparse non-negative high dimensional projection of word embeddings. (Subramanian et al., 2018) use a k-sparse denoising autoencoder to produce sparse non-negative high dimensional projection of word embeddings, which they called SParse Interpretable Neural Embeddings ($SPINE$). However, all these methods lack a natural extension for disambiguating the sense of a word in a context.

In a different line of work, Vilnis and McCallum (2015) proposed representing words as Gaussian distributions to embed uncertainty in dimensions of the embedding to better capture concepts like entailment. However, Athiwaratkun and Wilson (2017) argued that such a single prototype model can't capture multiple distinct meanings and proposed $Word2GM$ to learn multiple Gaussian embeddings per word. The prototypes were generalized to ellipical distributions in (Muzellec and Cuturi, 2018). A major limitation with such an approach is the restriction on the number of prototypes per word that can be learned, which is limited to 2 or 3 due to computational constraints. Many words such as 'Cell' can have more than 5 senses. Another open issue is that of disambiguating senses of a polysemous word in a context – there is no obvious way to embed phrases and sentences with such embeddings.

Multiple works have proposed multi-prototype embeddings to capture the senses of a polysemous word. For example, Neelakantan et al. (2015) extends the skipgram model to learn multiple embeddings of a word, where the number of senses of a word is either fixed or is learned through a non-parametric approach. Huang et al. (2012) learns multi-prototype embeddings by clustering the context window features of a word. However, these methods can't capture concepts like entailment. Tian et al. (2014) learns a probabilistic version of skipgram for learning multi-sense embeddings and hence, can capture entailment. However, all these models suffer from computational constraints and either restrict the number of prototypes learned for each word to 2-3 or restrict the words for which multiple prototypes are learned to the top k frequent words in the vocabulary.

Prior attempts at representing polysemy include (Pantel and Lin, 2002), who generate global senses by figuring out the best representative words for each sense from co-occurrence graph, and (Reisinger and Mooney, 2010), who generate senses for each word by clustering the context vectors of the occurrences of the word. Further attempts include Arora et al. (2018), who express single prototype dense embeddings, such as $Word2vec$ and $Glove$, as linear combinations of sense vectors. However, their underlying linearity assumption breaks down in real data, as shown by Mu et al. (2017). Further, the linear coefficients can be negative and have values far greater than 1 in magnitude, making them difficult to interpret. Neelakantan et al. (2015) and Huang et al. (2012) represent a context by the average of the embeddings of the words to disambiguate the sense of a target word present in the context. On the other hand, Mu et al. (2017) suggest representing sentences as a hyperspace, rather than a single vector, and represent words by the intersection of the hyperspaces representing the sentences it occurs in.

A number of works use naïve Bayesian method (Charniak et al., 2013) and topic models (Brody and Lapata, 2009; Yao and Van Durme, 2011; Pedersen, 2000; Lau et al., 2012, 2013, 2014) to learn senses from local contexts, treating each in-

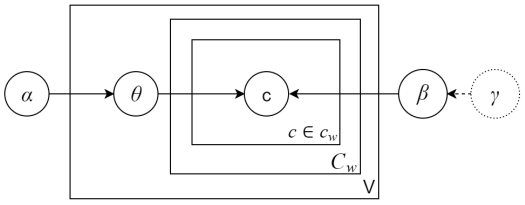

Figure 1: Generative model for co-occurrence matrix. Dirichlet prior γ is used in WarpLDA.

stance of a word within a context as a pseudo-document, and achieve state of the art results in WSI task (Manandhar et al., 2010). Since this approach requires training a single topic model per target word, it does not scale to all the words in the vocabulary.

In a different line of work, (Tang et al., 2014; Guo and Diab, 2011; Wang et al., 2015; Tang et al., 2015; Xun et al., 2017) transform topic models to learn local context level information through sense latent variable, in addition to the document level information through topic latent variable, for producing more fine grained topics from the corpus.

3 Notation

Let $V = \{w_1, w_2, ..w_{|V|}\}$ denote the set of unique tokens in corpus (vocabulary). Let C denote the word-word co-occurrence matrix constructed from the corpus, i.e., C_{ij} is the number of times w_j has occurred in the context of w_i. We define a context around a token w as the set of n words to the left and n words to the right of w. We denote the size of context window by n. Typically $n = 5$.

Our algorithm uses LDA to infer a *sense model* β – essentially a set of k probability distributions over V – from the corpus. It then uses the sense model to encode a word w as a k'-dimensional μ-sparse vector θ_w. Here, we use α and γ, respectively, to denote the Dirichlet priors of θ_w, the sense distribution of a word w, and β_z, context word distribution in a sense z. JS is a $k \times k$ matrix that measures the similarity between senses. We denote the z^{th} row of a matrix M by M_z.

4 Recovering senses

To recover senses, we suppose the following generative model for generating words in a context of size n (see Figure 1).

1. For each word $w \in V$, generate a distribution over senses θ_w from the Dirichlet distribution with prior α.
2. For each context c_w around target word w, and for each of the $2n$ tokens $\in c_w$, do
   (a) Sample sense $z \sim Multinomial(\theta_w)$.
   (b) Sample token $c \sim Multinomial(\beta_{z_n})$.

Such a generative model will generate a co-occurrence matrix C that can also be generated by another model. C is a matrix whose columns C_w are interpreted as a *document* formed from the count of all the tokens that have occurred in a context centered at w. Given a Dirichlet prior of parameter α on sense distribution of C_w and β, the distribution over context words for each sense, document C_w (and thus the co-occurrence matrix C) is generated as follows:

1. Generate $\theta_w \sim Dirichlet(\alpha)$.
2. Repeat N times to generate C_w:
   (a) Sample sense $z \sim Multinomial(\theta_w)$.
   (b) Sample token $c \sim Multinomial(\beta_z)$.

Based on this generative model, given the co-occurrence matrix C, we infer the matrix β and the maximum aposteriori estimate θ_w for each word using a fast variational inference tool such as WarpLDA (Chen et al., 2016).

5 $Word2Sense$ embeddings

$Word2Sense$ embeddings are probability distributions over senses. We discuss how to use the senses recovered by inference on the generative model in section 4 to construct word embeddings. We demonstrate that the embeddings so computed are competitive with various multi-modal embeddings in semantic similarity and entailment tasks.

5.1 Computing $Word2Sense$ embeddings

Denote the probability of occurrence of a word in the corpus by $p(w)$. We approximate the probability of the word $p(w)$ by its empirical estimate $\|C_w\|_1 / \sum_{w' \in V} \|C_{w'}\|_1$. We define the global probability $p_Z(z)$ of a sense z as the probability that a randomly picked token in the corpus has that sense in it's context window. We approximate the global distribution of generated senses using the following formulation.

$$p_Z(z) = \sum_{w \in V} \theta_w[z] p(w) \quad \forall z \in \{1..k\}.$$

Then, for each word $w \in V$, we compute $p_c(w)$, its sense distribution (when acting as a context word) as follows:

$$p_c(z|w) = \frac{p(w|z)p_Z(z)}{p(w)} = \frac{\beta_{w,z}p_Z(z)}{p(w)}.$$

Eliminating redundant senses. LDA returns a number of topics that are very similar to each other. Examples of such topics are given in Table 11 in appendix. These topics need to be merged, since inferring two similar words against such senses can cause them to be (predominantly) assigned to two different topic ids, causing them to look more dissimilar than they actually are. In order to eliminate redundant senses, we use the similarity of topics according to the Jensen Shannon (JS) divergence. We construct the topic similarity matrix $JS \in \mathbb{R}^{k \times k}$, whose $[i,j]-th$ entry $JS[i,j]$ is the JS divergence between senses β_i and β_j. Recall that JS divergence $JSdiv(p,q)$ between two multinomial distributions $p, q \in \mathbb{R}^k$ is given by

$$\sum_{i=1}^{k} -p_i \log \frac{2p_i}{p_i+q_i} - q_i \log \frac{2q_i}{p_i+q_i}. \quad (1)$$

We run agglomerative clustering on the JS matrix to merge similar topics. We use the following distance metric to merge two clusters D_i and D_j:

$$d(D_i, D_j) = \frac{1}{|D_i||D_j|} \sum_{a \in D_i,\ b \in D_j} JS[a,b]^{0.5}$$

Let $D_{i=1..k'}$ denote the final set of k' clusters obtained after clustering. We approximate the occurrence probability of the merged cluster of senses D_i by $p_D(D_i) = \sum_{a \in D_i} p_Z(a)$. Table 11 in appendix shows some clusters formed after clustering. Using the merged senses, we compute the embedding v_w of word w — a distribution over senses indexed by $z \in \{1..k\}$ — as follows:

$$\begin{aligned} \hat{v}_w[z] &= p_c(z|w) + \theta_w[z] \\ v'_w &= Truncate_\mu(Project(\hat{v}_w) \oslash p_D(.)) \\ v_w &= v'_w / ||v'_w||_1. \end{aligned} \quad (2)$$

$Project$ is the function that maps $v \in \mathbb{R}^k$ to $v' \in \mathbb{R}^{k'}$ by merging the coordinates corresponding to the merged senses: $v'[i] = \sum_{a \in D_i} v[a]$. $Truncate_\mu$ sparsifies the input by truncating it to the μ highest non-zeros in the vector.

5.2 Evaluation

We compare $Word2Sense$ embeddings with the state-of-the-art on word similarity and entailment tasks as well as on benchmark downstream tasks.

5.2.1 Hyperparameters

We train $Word2vec$ Skip-Gram embeddings with 10 passes over the data, using separate embeddings for the input and output contexts, 5 negative samples per positive example, window size n = 2 and the same sub-sampling and dynamic window procedure as in (Mikolov et al., 2013). For $Word2GM$, we make 5 passes over the data (due to very long training time of the published code [1]), using 2 modes per word, 1 negative sample per positive example, spherical covariance model, window size n = 10 and the same sub-sampling and dynamic window procedure as in (Athiwaratkun and Wilson, 2017). Since there is no recommended dimension in these papers, we report the numbers for the best performing embedding size. We report the performance of $Word2vec$ and $Word2GM$ at dimension 500 and 400 respectively[2]. We report the performance of $SPOWV$ and $SPINE$ in benchmark downstream tasks, that use $Word2vec$ as base embeddings, using the recommended settings as given in (Faruqui et al., 2015) and (Subramanian et al., 2018) respectively[3]. For Multi-Sense Skip-Gram model (MSSG) (Neelakantan et al., 2015), we use pre-trained word and sense representations [4].

We found $k = 3000, \alpha = 0.1$ and $\gamma = 0.001$ to be good hyperparamters for WarpLDA to recover fine-grained senses from the corpus. A choice of $k' \approx \frac{3}{4}k$ that merges $k/4$ senses improved results. We use a context window size $n = 5$ and truncation parameter $\mu = 75$. We think $\mu = 75$ works best because we found the average sparsity of $p_c(.|w)$ to be around 100. Since we decrease the number of senses by $1/4^{th}$ after post-processing, the average sparsity reduces to close to 75. If a word is not present in the vocabulary, we take an embedding on the unit simplex, that contains equal values in all the dimensions.

5.2.2 Word Similarity

We evaluate our embeddings at scoring the similarity or relatedness of pairs of words on several

[1] https://github.com/benathi/word2gm

[2] We tried 100, 200, 300, 400, 500 dimensions for $Word2vec$, and 50, 100, 200, 400 dims for $Word2GM$

[3] The two models don't perform better than $Word2vec$ in similarity tasks and don't show performance in entailment.

[4] bitbucket.org/jeevan_shankar/multi-sense-skipgram

Dataset	Word 2Sense	Word 2Vec	$Word2GM$	MSSG 300-dim 30K	MSSG 300-dim 6K
WS353-S	0.747	**0.769**	0.756(0.767)	0.753	0.761
WS353-R	**0.708**	0.703	0.609(0.717)	0.598	0.607
WS353	0.723	**0.732**	0.669(0.734)	0.685	0.694
Simlex-999	0.388	0.393	**0.399**(0.293)	0.350	0.351
MT-771	0.685	**0.688**	0.686(0.608)	0.646	0.645
MEN	0.772	**0.780**	0.740(0.736)	0.665	0.675
RG	0.790	**0.824**	0.755(0.745)	0.719	0.714
MC	0.806	**0.827**	0.819(0.791)	0.684	0.763
RW	**0.374**	0.365	0.339(0.286[a])	0.15	0.15

Table 2: Comparison of word embeddings on word similarity evaluation datasets. For MSSG learned for top $30K$ and $6k$ words, we report the similarity of the global vectors of word, which we find to be better than comparing all the local vectors of words. For $Word2GM$, we report numbers from our tuning as well as from the paper (in paranthesis). Note that we report higher numbers in all cases, except on WS353-S and WS353-R datasets. We attribute this to fewer passes over the data and possibly different pre-processing. [a] 0.353 with a different metric.

Method	Best AP	Best F1
(Baroni et al., 2012)	0.751	-
$Word2GM$(10)-Cos	0.729	0.757
$Word2GM$(10)-KL	0.747	0.763
$Word2Sense$	0.751	0.761
$Word2Sense$ -full	**0.791**	**0.798**

Table 3: Comparison of embeddings on word entailment. The number reported for (Baroni et al., 2012) has been taken from original paper and uses the balAP-inc metric. For $Word2GM$, we were able to reproduce results in the original paper; we report results using both Cosine and KL divergence metrics. For $Word2Sense$, we use KL divergence.

datasets annotated with human scores: Simlex-999 (Hill et al., 2015), WS353-S and WS353-R (Finkelstein et al., 2002), MC (Miller and Charles, 1991), RG (Rubenstein and Goodenough, 1965), MEN (Bruni et al., 2014), RW (Luong et al., 2013) and MT-771 (Radinsky et al., 2011; Halawi et al., 2012).

We predict similarity/relatedness score of a pair of words $\{w_1, w_2\}$ by computing the JS divergence (see Equation 1) between the embeddings $\{v_{w_1}, v_{w_2}\}$ as computed in Equation 2. For other embeddings, we use cosine similarity metric to measure similarity between embeddings. The final prediction effectiveness of an embedding is given by computing Spearman correlation between the predicted scores and the human annotated scores.

Table 2 compares our embeddings to multimodal Gaussian mixture (Word2GM) model (Athiwaratkun and Wilson, 2017) and Word2vec (Mikolov et al., 2013). We extensively tune hyperparameters of prior work, often achieving better results than previously reported. We concluded from this exercise that SkipGram ($Word2vec$) is the best among all the unsupervised embeddings at similarity and relatedness tasks. We see that while being interpretable and sparser than the 500-dimensional $Word2vec$, $Word2Sense$ embeddings is competitive with $Word2vec$ on all the datasets.

5.2.3 Word entailment

Given two words w_1 and w_2, w_2 entails w_1 (denoted by $w_1 \models w_2$) if all instances of w_1 are w_2. We compare $Word2Sense$ embeddings with $Word2GM$ on the entailment dataset provided by (Baroni et al., 2012). We use KL divergence to generate entailment scores between words w_1 and w_2. For $Word2GM$, we use both cosine similarity and KL divergence, as used in the original paper. We report the F1 scores and Average Precision(AP) scores for reporting the quality of prediction. Table 3 compares the performance of our embedding with $Word2GM$. We notice that $Word2Sense$ embeddings with $\mu = k'$ (denoted $Word2Sense$ -full in the table), i.e., with no truncation, yields the best results. We do not compare with hyperbolic embeddings (Tifrea et al., 2019; Dhingra et al., 2018) because these embeddings are designed mainly to perform well on entailment tasks, but are far off from the performance of Euclidean embeddings on similarity tasks.

5.2.4 Downstream tasks

We compare the performance of $Word2Sense$ with $Word2vec$, $SPINE$ and $SPOWV$ embeddings on the following downstream classification tasks: sentiment analysis (Socher et al., 2013), news classification[5], noun phrase chunking (Lazaridou et al., 2013) and question classification (Li and Roth, 2006). We do not compare with $Word2GM$ and MSSG as there is no obvious way to compute sentence embeddings from multi-modal word embeddings. The sentence embedding needed for text classification is the average of the embeddings of words in the sentence,

[5]http://qwone.com/ jason/20Newsgroups/

Task	Word 2Sense	Word 2Vec	SPOWV	SPINE
Sports news	**0.865**	0.826	0.834	0.810
Computer news	0.861	0.838	**0.862**	0.856
Religion news	0.965	**0.975**	0.966	0.936
NP Bracketing	**0.693**	0.686	0.687	0.665
Sentiment analysis	0.815	0.812	**0.816**	0.778
Question clf.	0.970	0.969	**0.980**	0.940

Table 4: Comparison on benchmark downstream tasks.

as in (Subramanian et al., 2018). We pick the best among SVMs, logistic regression and random forest classifier to classify the sentence embeddings based on accuracy on the development set. Table 4 reports the accuracies on the test set. More details of the tasks are provided in Appendix E.

6 Interpretability

We evaluate the interpretability of the $Word2Sense$ embeddings against $Word2vec$, $SPINE$ and $SPOWV$ models using the word intrusion test following the procedure in (Subramanian et al., 2018). We select the 15k most frequent words in the intersection of our vocabulary and the Leipzig corpus (Goldhahn et al., 2012). We select a set H of 300 random dimensions or senses from 2250 senses. For each dimension $h \in H$, we sort the words in the 15k vocabulary based on their weight in dimension h. We pick the top 4 words in the dimension and add to this set a random intruder word that lies in the bottom half of the dimension h and in the top 10 percentile of some other dimension $h' \in H \setminus \{h\}$ (Fyshe et al., 2014; Faruqui et al., 2015). For the dimension h to be claimed interpretable, independent judges must be able to easily separate the intruder word from the top 4 words.

We split the 300 senses into ten sets of 30 senses, and assigned 3 judges to annotate the intruder in each of the 30 senses in a set (we used a total of 30 judges). For each question, we take the majority voted word as the predicted intruder. If a question has 3 different annotations, we count that dimension as non interpretable[6]. Since, we followed the procedure as in (Subramanian et al., 2018), we compare our performance with the results reported in their paper. Table 5 shows that $Word2Sense$ is competitive with the best interpretable embeddings.

[6](Subramanian et al., 2018) used a randomly picked intruder in this case.

Method	Agreement	Precision
$Word2vec$	0.77/0.18	0.261
$SPOWV$	0.79/0.28	0.418
$SPINE$	**0.91**/0.48	0.748
$Word2Sense$	0.891/**0.589**	**0.753**

Table 5: Comparison of embeddings on for Word Intrusion tasks. The second column indicates the inter annotator agreement – the first number is the fraction of questions for which at least 2 annotators agreed and the second indicates the fraction on which all three agreed. The last column is the precision of the majority vote.

6.1 Qualitative evaluation

We show the effectiveness of our embeddings at capturing multiple senses of a polysemous word in Table 1. For e.g. "tie" can be used as a verb to mean tying a rope, or drawing a match, or as a noun to mean clothing material. These three senses are captured in the top 3 dimensions of $Word2Sense$ embedding for "tie". Similarly, the embedding for "cell" captures the 5 senses discussed in section 1 within the top 15 dimensions of the embedding. The remaining top senses capture fine grained senses such as different kinds of biological cells – e.g. bone marrow cell, liver cell, neuron – that a subject expert might relate to.

7 $WordCtx2Sense$ embeddings

A word with several senses in the training corpus, when used in a context, would have a narrower set of senses. It is therefore important to be able to refine the representation of a word according to its usage in a context. Note that $Word2vec$ and $Word2GM$ models do not have such a mechanism. Here, we present an algorithm that generates an embedding for a target word $\hat{w}$ in a short context $T = \{w_1, .., w_N\}$ that reflects the sense in which the target word was used in the context. For this, we suppose that the senses of the word $\hat{w}$ in context T are an *intersection* of the senses of $\hat{w}$ and T. We therefore infer the sense distribution of T by restricting the support of the distribution to those senses $\hat{w}$ can take.

7.1 Methodology

We suppose that the words in the context T were picked from a mixture of a small number of senses. Let $S_k = \{\psi = (\psi_1, \psi_2, ..., \psi_k) : \psi_z \geq 0; \sum_z \psi_z = 1\}$ be the unit positive simplex. The generative model is as follows. Pick a $\psi \in S_k$, and let $P = \beta\psi$, where β is the collection of sense probability distributions recovered by LDA from

the corpus. Pick N words from P independently.

$$\text{Let} \quad A \sim P = \beta\psi, \quad \psi \in S_k, \tag{3}$$

where A is a vocabulary-sized vector containing the count of each word, normalized to sum 1. We do not use the Dirichlet prior over sense distribution as in the generative model in section 4, as we found its omission to be better at inferring the sense distribution of contexts.

Given A and β, we want to infer the sense distribution $\psi \in S_k$ that minimizes the log perplexity $f(\psi; A, \beta) = -\sum_i^{|V|} A_i log(\beta\psi)_i$ according to the generative model in Equation 3. The MWU – multiplicative weight update – algorithm (See Appendix A for details) is a natural choice to find such a distribution ψ, and has an added advantage. The MWU algorithm's estimate of a variable ψ w.r.t. a function f after t iterations (denoted $\psi^{(t)}$) satisfies

$$\psi^{(t)}[i] = 0, \text{ if } \psi^{(0)}[i] = 0 \, \forall i \in \{1..k\} \text{ and } \forall t \geq 0.$$

Therefore, to limit the set of possible senses in the inference of ψ to the μ senses that $\hat{w}$ can take, we initialize $\psi^{(0)}$ to the embedding $v_{\hat{w}}$. We used the embedding obtained in Equation 2 without the $Project$ operator that adds probabilities of similar senses, to correspond with the use of the original matrix β for MWU.

Further, to keep iterates close to the initial $\psi^{(0)}$, we add a regularizer to log perplexity. This is necessary to bias the final inference towards the senses that the target word has higher weights on. Thus the loss function on which we run MWU with starting point $\psi^{(0)} = v_{\hat{w}}$ is

$$f(\psi; A, \beta) = -\sum_{i=1}^{|V|} A_i log(\beta\psi)_i + \lambda KL(\psi, \psi^{(0)}) \tag{4}$$

where the second term is the KL divergence between two distributions scaled by a hyperparameter λ. Recall that $KL(p, q) = -\sum_{i=1}^{k} p_i \, log(p_i/q_i)$ for two distributions $p, q \in \mathbb{R}^k$. We use the final estimate $\psi^{(t)}$ as the Word-Ctx2Sense distribution of a word in the context.

7.2 Evaluation

We demonstrate that the above construction of a word's representation disambiguated in a context is useful by comparing with state-of-the-art unsupervised methods for polysemy disambiguation on two tasks: Word Sense Induction and contextual similarity. Specifically, we compare with MSSG, the K-Grassmeans model of (Mu et al., 2017), and the sparse coding method of (Arora et al., 2018). [7]

7.2.1 Hyperparameters

We use the same hyperparameter values for α, β, k and n as in section 5.2.1. We use $\mu = 100$ since we do not merge senses in this construction. We tune the hyperparameter λ to the task at hand.

7.2.2 Word Sense Induction

The WSI task requires clustering a collection of (say 40) short texts, all of which share a common polysemous word, in such a way that each cluster uses the common word in the same sense. Two datasets for this task are Semeval-2010 (Manandhar et al., 2010) and MakeSense-2016 (Mu et al., 2017). The evaluation criteria are F-score (Artiles et al., 2009) and V-Measure (Rosenberg and Hirschberg, 2007). V-measure measures the quality of a cluster as the harmonic mean of homogeneity and coverage, where homogeneity checks if all the data-points that belong to a cluster belong to the same class and coverage checks if all the data-points of the same class belong to a single cluster. F-score is the harmonic mean of precision and recall on the task of classifying whether the instances in a pair belong to the same cluster or not. F-score tends to be higher with a smaller number of clusters and the V-Measure tends to be higher with a larger number of clusters, and it is important to show performance in both metrics.

For each text corresponding to a polysemous word, we learn a sense distribution ψ using the steps in section 7.1. We tuned the parameter λ and found the best performance at $\lambda = 10^{-2}$. We use hard decoding to assign a cluster label to each text, i.e., we assign a label $k^\star = \text{argmax}_k \, \psi_k$ to a text with inferred sense vector ψ_k.

Suppose that this yields $\hat{k}$ distinct clusters for the instances corresponding to a polysemous word. We cluster them using agglomerative clustering into a final set of K clusters. The distance metric used to group two clusters D_i and D_j is

$$d(D_i, D_j) = max_{a \in D_i, b \in D_j} (JS[a, b])^{0.5}$$

[7] Note that we report baseline numbers from the original papers. These papers have trained their models on newer versions of Wikipedia dump that contain more than 3 billion tokens (MSSG uses a 1 billion token corpus). However, our model has been trained on a combined dataset of wiki-2009 dump and ukWaC, which contains around 3B tokens. Hence, there might be minor differences in comparing our model to the baseline models.

		MakeSense-2016		SemEval-2010	
Method	K	F-scr	V-msr	F-scr	V-msr
(Huang et al., 2012)	-	47.40	15.50	38.05	10.60
(Neelakantan et al., 2015)					
300D.30K.key	-	54.49	19.40	47.26	9.00
300D.6K.key	-	57.91	14.40	48.43	6.90
(Mu et al., 2017)	2	64.66	28.80	57.14	7.10
	5	58.25	34.30	44.07	14.50
(Arora et al., 2018)	2	-	-	58.55	6.1
	5	-	-	46.38	11.5
$WordCtx2Sense$	2	63.71	22.20	**59.38**	6.80
$\lambda = 0.0$	5	59.75	32.90	46.47	13.20
	6	59.13	34.20	44.04	14.30
$WordCtx2Sense$	2	**65.27**	24.40	59.15	6.70
$\lambda = 10^{-2}$	5	62.88	35.00	47.34	13.70
	6	61.43	**35.30**	44.70	**15.00**

Table 6: Comparison of $WordCtx2Sense$ with the state-of-the-art methods for Word Sense Induction on MakeSense-2016 and SemEval-2010 dataset. We report F-score and V-measure scores multiplied by 100.

Method	Pearson-coefficient
$WordCtx2Sense$ (a)	0.666
$WordCtx2Sense$ (b)	0.670
$Word2Sense$	0.644
$Word2vec$	0.651
(Mu et al., 2017)	0.637
(Huang et al., 2012)	0.657
(Arora et al., 2018)	0.652
$Word2GM$	0.655
MSSG.300D.30K	$\mathbf{0.679}^a$
MSSG.300D.6K	0.678^a

Table 7: Comparison on the SCWS task. Setting (a) for $WordCtx2Sense$ uses $\lambda = 0.1$ for all pairs, and setting (b) uses $\lambda = 10^{-3}$ for pairs containing same target words and $\lambda = 0.1$ for all other pairs. $Word2Sense$, $Word2Vec$ and $Word2GM$ neglect context and compare target words. ^{a} numbers reported from (Mu et al., 2017) whose experimental setup we could replicate.

where JS is the similarity matrix defined in section 5.

Results Table 6 shows the results of clustering on WSI SemEval-2010 dataset. WordCtx2Sense outperforms (Arora et al., 2018) and (Mu et al., 2017) on both F-score and V-measure scores by a considerable margin. We observe similar improvements on the MakeSense-2016 dataset.

7.2.3 Word Similarity in Context

The Stanford Contextual Word Similarity task (Huang et al., 2012) consists of 2000 pairs of words, along with the contexts the words occur in. Ten human raters were asked to rate the similarity of each pair words according to their use in the corresponding contexts, and their average score (on a 1 to 10 scale) is provided as the ground-truth similarity score. The goal of a contextual embedding would be to score these examples to maximize the correlation with this ground-truth.

We compute the $WordCtx2Sense$ of each word in its respective context as in section 7.1. For comparing the meaning of two words in context, we use the JS divergence between their $WordCtx2Sense$ embeddings. We report the coefficient between the ground-truth and $WordCtx2Sense$ according to two different settings of λ. (a) $\lambda = 0.1$, and b) $\lambda = 10^{-3}$ for inferring the contextual embedding of a word in those pairs that contain same target words, and $\lambda = 0.1$ for all other pairs. The main idea is to reduce unnecessary bias for comparing sense of a polysemous word in two different contexts.

Results Table 7 shows that sense embeddings using context information perform better than all the existing models, except MSSG models (Neelakantan et al., 2015). Also, computing the embeddings of a word using the contextual information improves results by aprox. 0.025, compared to the case when words embeddings are used directly.

8 Conclusion and future work

We motivated an efficient unsupervised method to embed words, in and out of context, in a way that captures their multiple senses in a corpus in an interpretable manner. We demonstrated that such interpretable embeddings can be competitive with dense embeddings like $Word2vec$ on similarity tasks and can capture entailment effectively. Further, the construction provides a natural mechanism to refine the representation of a word in a short context by disambiguating its senses. We have demonstrated the effectiveness of such contextual representations.

A natural extension to this work would be to capture the sense distribution of sentences using the same framework. This will make our model more comprehensive by enabling the embedding of words and short texts in the same space.

9 Acknowledgements

We thank Monojit Choudhury, Ravindran Kannan, Adithya Pratapa and Anshul Bawa for many helpful discussions. We thank all anonymous reviewers for their constructive comments.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2018. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association of Computational Linguistics*, 6:483–495.

Javier Artiles, Enrique Amigó, and Julio Gonzalo. 2009. The role of named entities in web people search. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 534–542. Association for Computational Linguistics.

Ben Athiwaratkun and Andrew Gordon Wilson. 2017. Multimodal word distributions. *arXiv preprint arXiv:1704.08424*.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32. Association for Computational Linguistics.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 103–111. Association for Computational Linguistics.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Eugene Charniak et al. 2013. Naive Bayes word sense induction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1437.

Jianfei Chen, Kaiwei Li, Jun Zhu, and Wenguang Chen. 2016. WarpLDA: a cache efficient O(1) algorithm for Latent Dirichlet Allocation. *Proceedings of the VLDB Endowment*, 9(10):744–755.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Paramveer S Dhillon, Dean P Foster, and Lyle H Ungar. 2015. Eigenwords: Spectral word embeddings. *The Journal of Machine Learning Research*, 16(1):3035–3078.

Bhuwan Dhingra, Christopher J Shallue, Mohammad Norouzi, Andrew M Dai, and George E Dahl. 2018. Embedding text in hyperbolic spaces. *arXiv preprint arXiv:1806.04313*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah Smith. 2015. Sparse overcomplete word vector representations. *arXiv preprint arXiv:1506.02004*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

Alona Fyshe, Partha P Talukdar, Brian Murphy, and Tom M Mitchell. 2014. Interpretable semantic vectors from a joint model of brain-and text-based meaning. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2014, page 489. NIH Public Access.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In *LREC*, volume 29, pages 31–43.

Weiwei Guo and Mona Diab. 2011. Semantic topic models: Combining word distributional statistics and dictionary definitions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 552–561. Association for Computational Linguistics.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.

Lushan Han, Tim Finin, Paul McNamee, Anupam Joshi, and Yelena Yesha. 2012. Improving word similarity by augmenting PMI with estimates of word polysemy. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1307–1322.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

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 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013. unimelb: Topic modelling-based word sense induction for web snippet clustering. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 217–221.

Jey Han Lau, Paul Cook, Diana McCarthy, Spandana Gella, and Timothy Baldwin. 2014. Learning word sense distributions, detecting unattested senses and identifying novel senses using topic models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 259–270.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word sense induction for novel sense detection. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 591–601. Association for Computational Linguistics.

Angeliki Lazaridou, Eva Maria Vecchi, and Marco Baroni. 2013. Fish transporters and miracle homes: How compositional distributional semantics can help NP parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1908–1913.

Rémi Lebret and Ronan Collobert. 2013. Word emdeddings through Hellinger PCA. *arXiv preprint arXiv:1312.5542*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Suresh Manandhar, Ioannis P Klapaftis, Dmitriy Dligach, and Sameer S Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 63–68. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. Geometry of polysemy. In *Proceedings of the 5th International Conference on Learning Representations*. OpenReview.net.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. *Proceedings of COLING 2012*, pages 1933–1950.

Boris Muzellec and Marco Cuturi. 2018. Generalizing point embeddings using the wasserstein space of elliptical distributions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10237–10248. Curran Associates, Inc.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*.

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. 2000. A simple approach to building ensembles of Naive Bayesian classifiers for word sense disambiguation. *arXiv preprint cs/0005006*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

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. Association for Computational Linguistics.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. 2016. Swivel: Improving embeddings by noticing what's missing. *arXiv preprint arXiv:1602.02215*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1282–1291.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. 2018. Spine: Sparse interpretable neural embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Sparse word embeddings using L1 regularized online learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2915–2921. AAAI Press.

Guoyu Tang, Yunqing Xia, Jun Sun, Min Zhang, and Thomas Fang Zheng. 2014. Topic models incorporating statistical word senses. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 151–162. Springer.

Guoyu Tang, Yunqing Xia, Jun Sun, Min Zhang, and Thomas Fang Zheng. 2015. Statistical word sense aware topic models. *Soft Computing*, 19(1):13–27.

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 *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 151–160.

Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. 2019. Poincaré GloVe: Hyperbolic word embeddings. In *Proceedings of the 7th International Conference on Learning Representations*.

Luke Vilnis and Andrew McCallum. 2015. Word representations via Gaussian embedding. In *Proceedings of the 3rd International Conference on Learning Representations*.

Jing Wang, Mohit Bansal, Kevin Gimpel, Brian D Ziebart, and Clement T Yu. 2015. A sense-topic model for word sense induction with unsupervised data enrichment. *Transactions of the Association for Computational Linguistics*, 3:59–71.

Guangxu Xun, Yaliang Li, Jing Gao, and Aidong Zhang. 2017. Collaboratively improving topic discovery and word embeddings by coordinating global and local contexts. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–543. ACM.

Xuchen Yao and Benjamin Van Durme. 2011. Nonparametric bayesian word sense induction. In *Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing*, pages 10–14. Association for Computational Linguistics.

A Multiplicate Weight Update

Algorithm 1 Multiplicative Weight update

function MWU(k, L_f, f, $\theta^{(0)}$, ITER)
▷ k denotes dimension of variable θ, f denotes a function of θ, L_f if lipschitz constant of f, θ_0 denotes initial starting point of θ, ITER denotes the number of iterations to run
  for t **do** = 1 .. ITER
    $\eta = \frac{1}{k}\sqrt{2\,log(k/t)}$
    $\hat{\theta}^{(t)} = \hat{\theta}^{(t-1)} \exp\left(\eta \nabla f(\theta)|_{\theta=\theta^{(t-1)}}\right)$
    $\theta_t = \hat{\theta}^{(t)} / \|\hat{\theta}^{(t)}\|_1$
  end for
end function

B Hyper-parameter tuning for $Word2vec$

We use the default hyperparameters for training $Word2vec$, as given in Mikolov et al. (2013). We tuned the embedding size, to see if the performance improves with increasing number of dimensions. Table 8 shows that there is minor improvement in performance in different similarity and relatedness tasks as the embedding size is increased from 100 to 500.

C Hyper-parameter tuning for $Word2GM$

We use the default hyperparameters for training $Word2GM$, as given in Athiwaratkun and Wilson (2017). We tuned the embedding size, to see if the performance improves with increasing number of dimensions. Table 9 shows that there is minor improvement in performance of $Word2GM$, when the embedding size is increased from 100 to 400.

D Hyper-parameter tuning for $Word2Sense$

For generating senses, we use WarpLDA that has 3 different hyperparameters, a) Number of topics k b) α, the dirichlet prior of sense distribution of each word and c) γ, the dirichlet prior of word distribution of each sense. We keep k fixed at 3000 and vary α and β. We show a small subset of the hyperparameter space searched for α and β. We report the performance of word embeddings computed by Equation 3, without the $Project$ step, in different similarity tasks. Table 10 shows that the performance slowly decreases as we increase β and somewhat stays constant with α. Hence, we choose $\alpha = 0.1$ amd $\gamma = 0.001$ for carrying out our experiments.

E Benchmark downstream tasks

In this section, we discuss about the different downstream tasks considered. We follow the same procedure as (Faruqui et al., 2015) and (Subramanian et al., 2018)[8].

- **Sentiment analysis** This is a binary classification task on Sentiment Treebank dataset (Socher et al., 2013). The task is to give a sentence a positive or a negative sentiment label. We used the provided train, dev. and test splits of sizes 6920, 872 and 1821 sentences respectively.
- **Noun phrase bracketing** NP bracketing task (Lazaridou et al., 2013) involves classifying a noun phrase of 3 words as left bracketed or right bracketed. The dataset contains 2,227 noun phrases split into 10 folds. We append the word vectors of three words to get feature representation (Faruqui et al., 2015). We report 10-fold cross validation accuracy.
- **Question classification** Question classification task (Li and Roth, 2006) involves classifying a question into six different types, e.g., whether the question is about a location, about a person or about some numeric information. The training dataset consists of 5452 labeled questions, and the test dataset consists of 500 questions.
- **News classification** We consider three binary categorization tasks from the 20 Newsgroups dataset. Each task involves categorizing a document according to two related categories (a) Sports: baseball vs. hockey (958/239/796) (b) Comp.: IBM vs. Mac (929/239/777) (c) Religion: atheism vs. christian (870/209/717), where the brackets show training/dev./test splits.

[8] We use the evaluation code given in https://github.com/harsh19/SPINE

Dataset	$Word2vec-100$	$Word2vec-200$	$Word2vec-300$	$Word2vec-400$	$Word2vec-500$
SCWS	0.638	0.646	0.648	0.649	0.651
Simlex-999	0.365	0.388	0.387	0.393	0.393
MEN	0.749	0.760	0.763	0.767	0.780
RW	0.361	0.361	0.363	0.365	0.365
MT-771	0.684	0.685	0.681	0.681	0.688
WS353	0.705	0.719	0.721	0.733	0.732
WS353-S	0.744	0.766	0.768	0.768	0.769
WS353-R	0.669	0.679	0.670	0.696	0.703

Table 8: Performance of $Word2vec$ at different embedding size, in similarity tasks.

Dataset	$Word2GM-100$	$Word2GM-200$	$Word2GM-400$
SL	0.345	0.385	0.398
WS353	0.664	0.672	0.669
WS353-S	0.727	0.735	0.751
WS353-R	0.626	0.625	0.607
MEN	0.740	0.755	0.761
MC	0.812	0.802	0.826
RG	0.730	0.772	0.750
MT-771	0.638	0.664	0.682
RW	0.303	0.338	0.338

Table 9: Performance of $Word2GM$, with spherical covariance matrix for each embeddding, at different embedding sizes in similarity tasks

α, γ	SCWS	MT-771	WS353	RG	MC	WS353-S	WS353-R	MEN
0.1, 0.001	0.596	0.623	0.654	0.794	0.767	0.685	0.662	0.754
0.1, 0.005	0.595	0.625	0.647	0.809	0.758	0.699	0.638	0.748
0.1, 0.1	0.584	0.609	0.601	0.733	0.671	0.618	0.626	0.738
1.0, 0.001	0.596	0.607	0.651	0.815	0.700	0.692	0.658	0.743
1.0, 0.005	0.613	0.620	0.640	0.792	0.691	0.676	0.653	0.749
1.0, 0.05	0.559	0.562	0.583	0.730	0.742	0.609	0.581	0.711
1.0, 0.1	0.587	0.602	0.602	0.755	0.720	0.641	0.605	0.727
10.0, 0.001	0.595	0.610	0.628	0.822	0.772	0.664	0.639	0.747
10.0, 0.005	0.608	0.635	0.657	0.808	0.826	0.708	0.648	0.739
10.0, 0.05	0.562	0.539	0.544	0.786	0.710	0.573	0.551	0.717
10.0, 0.1	0.573	0.606	0.570	0.773	0.696	0.612	0.593	0.724

Table 10: Performance of $Word2Sense$ as computed in eq. 3 without the $Project$ step in similarity tasks, at different hyperparameter settings.

Cluster size	Top 10 words with the highest probability in the sense's distribution
6	tennessee, kentucky, alabama, mississippi, georgia, arkansas, nashville, memphis, louisville, atlanta state, idaho, oregon, montana, wisconsin, utah, nevada, wyoming, states, california illinois, chicago, wisconsin, michigan, milwaukee, rapids, madison, detroit, iowa, grand
19	lol, im, thats, dont, mrplow, yeah, cant, it, ive, ur im, dont, ive, cant, didnt, thats, lol, ur, my, cos my, ve, have, it, me, n't, ll, just, blog, but
5	lgame, games, adventure, gameplay, 3d, players, play, arcade, of, fun game, multiplayer, games, gameplay, gaming, xbox, shooter, gamers, mode, halo cheats, mario, super, game, arcade, unlock, mode, nintendo, cheat, bros
7	charlton, striker, midfield, defender, leeds, midfielder, darren, goal, bowyer, danny swansea, derby, leicester, city, wolves, watford, burnley, boss, stoke, swans manager, albion, club, coach, season, football, boss, fa, robson, gary

Table 11: Examples of clusters formed after agglomerative clustering. Each group of rows shows a randomly picked cluster, it's size and top 10 words of 3 randomly picked senses from the cluster. The clusters represent U.S. states, generic words, video games, and soccer respectively.