DISTRIBUTIONAL INCLUSION VECTOR EMBEDDING FOR UNSUPERVISED HYPERNYMY DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Modeling hypernymy, such as poodle is-a dog, is an important generalization aid to many NLP tasks, such as entailment, relation extraction, and question answering. Supervised learning from labeled hypernym sources, such as WordNet, limit the coverage of these models, which can be addressed by learning hypernyms from unlabeled text. Existing unsupervised methods either do not scale to large vocabularies or yield unacceptably poor accuracy. This paper introduces *distributional inclusion vector embedding (DIVE)*, a simple-to-implement unsupervised method of hypernym discovery via per-word non-negative vector embeddings which preserve the inclusion property of word contexts. In experimental evaluations more comprehensive than any previous literature of which we are aware—evaluating on 11 datasets using multiple existing as well as newly proposed scoring functions—we find that our method provides up to double the precision of previous unsupervised methods, and the highest average performance, using a much more compact word representation, and yielding many new state-of-the-art results. In addition, the meaning of each dimension in DIVE is interpretable, which leads to a novel approach on word sense disambiguation as another promising application of DIVE.

1 INTRODUCTION

Numerous applications benefit from compactly representing context distributions, which assign meaning to objects under the rubric of *distributional semantics*. In natural language processing, distributional semantics has long been used to assign meanings to words (that is, to *lexemes* in the dictionary, not individual instances of word tokens). The meaning of a word in the distributional sense is often taken to be the set of textual contexts (nearby tokens) in which that word appears, represented as a large sparse bag of words (SBOW). Without any supervision, word2vec (Mikolov et al., 2013), among other approaches based on matrix factorization (Levy et al., 2015a), successfully compress the SBOW into a much lower dimensional embedding space, increasing the scalability and applicability of the embeddings while preserving (or even improving) the correlation of geometric embedding similarities with human word similarity judgments.

While embedding models have achieved impressive results, context distributions capture more semantic features than just word similarity. The *distributional inclusion hypothesis* (DIH) (Weeds & Weir, 2003; Geffet & Dagan, 2005; Cimiano et al., 2005) posits that the context set of a word tends to be a subset of the contexts of its hypernyms. For a concrete example, most adjectives that can be applied to poodle can also be applied to dog, because dog is a hypernym of poodle. For instance, both can be obedient. However, the converse is not necessarily true — a dog can be straight-haired but a poodle cannot. Therefore, dog tends to have a broader context set than poodle. Many asymmetric scoring functions comparing SBOW based on DIH have been developed for automatic hypernymy detection (Weeds & Weir, 2003; Geffet & Dagan, 2005; Santus et al., 2017).

Hypernymy detection plays a key role in many challenging NLP tasks, such as textual entailment (Sammons et al., 2011), coreference (Ponzetto & Strube, 2006), relation extraction (Demeester et al., 2016) and question answering (Huang et al., 2008). Leveraging the variety of contexts and inclusion properties in context distributions can greatly increase the ability to discover taxonomic structure among words (Santus et al., 2017). The inability to preserve these features limits the semantic representation power and downstream applicability of some popular existing unsupervised learning approaches such as word2vec.

Several recently proposed methods aim to encode hypernym relations between words in dense embeddings, such as Gaussian embedding (Vilnis & McCallum, 2015; Athiwaratkun & Wilson, 2017), order embedding (Vendrov et al., 2016), H-feature detector (Roller & Erk, 2016), HyperScore (Nguyen et al., 2017), dual tensor (Glavaš & Ponzetto, 2017), Poincaré embedding (Nickel & Kiela, 2017), and LEAR (Vulić & Mrkšić, 2017). However, the methods focus on supervised or semi-supervised setting (Vendrov et al., 2016; Roller & Erk, 2016; Nguyen et al., 2017; Glavaš & Ponzetto, 2017; Vulić & Mrkšić, 2017), do not learn from raw text (Nickel & Kiela, 2017) or lack comprehensive experiments on the hypernym detection task (Vilnis & McCallum, 2015; Athiwaratkun & Wilson, 2017).

Recent studies (Levy et al., 2015b; Santus et al., 2017) have underscored the difficulty of generalizing supervised hypernymy annotations to unseen pairs — classifiers often effectively memorize prototypical hypernyms ('general' words) and ignore relations between words. These findings motivate us to develop more accurate and scalable unsupervised embeddings to detect hypernymy and propose several scoring functions to analyze the embeddings from different perspectives.

1.1 CONTRIBUTIONS

- A novel unsupervised low-dimensional embedding method to model inclusion relations among word contexts via performing non-negative matrix factorization (NMF) on a weighted PMI matrix, which can be efficiently optimized using modified skip-grams.
- Several new asymmetric comparison functions to measure inclusion and generality properties and to evaluate different aspects of unsupervised embeddings.
- Extensive experiments on 11 datasets demonstrate the learned embeddings and comparison functions achieve state-of-the-art performances on unsupervised hypernym detection while requiring much less memory and compute than approaches based on the full SBOW.
- A qualitative experiment illustrates DIVE can be used to solve word sense disambiguation, especially when efficiently modeling word senses at multiple granularities is desirable.

2 METHOD

The *distributional inclusion hypothesis* (DIH) suggests that the context set of a hypernym tends to contain the context set of its hyponyms. That is, when representing a word as the counts of contextual co-occurrences, the count in every dimension of hypernym y tends to be larger than or equal to the corresponding count of its hyponym x:

$$x \preceq y \iff \forall c \in V,\ \#(x,c) \leq \#(y,c), \tag{1}$$

where $x \preceq y$ means y is a hypernym of x, V is the set of vocabulary, and $\#(x,c)$ indicates the number of times that word x and its context word c co-occur in a small window with size $|W|$ in corpus D.

Our goal is to produce lower-dimensional embeddings that preserve the inclusion property that the embedding of hypernym y is larger than or equal to the embedding of its hyponym x in every dimension. Formally, the desirable property can be written as

$$x \preceq y \iff \mathbf{x}[i] \leq \mathbf{y}[i]\ , \forall i \in \{1, ..., d_0\}, \tag{2}$$

where d_0 is number of dimensions in the embedding space. We add additional non-negativity constraints, i.e. $x[i] \geq 0, y[i] \geq 0, \forall i$, in order to increase the interpretability of the embeddings (the reason will be explained later in this section).

This is a challenging task. In reality, there are a lot of noise and systematic biases which cause the violation of DIH in Equation (1) (i.e. $\#(x,c) > \#(y,c)$ for some neighboring word c), but the general trend can be discovered by processing several thousands of neighboring words in SBOW together. After the compression, the same trend has to be estimated in a much smaller embedding space which discards most of the information in SBOW, so it is not surprising to see most of the unsupervised hypernymy detection studies use SBOW (Santus et al., 2017) and the existing unsupervised embeddings like Gaussian embedding have degraded accuracy (Vulić et al., 2016).

2.1 INCLUSION PRESERVING MATRIX FACTORIZATION

Popular methods of unsupervised word embedding are usually based on matrix factorization (Levy et al., 2015a). The approaches first compute a co-occurrence statistic between the wth word and the cth context word as the (w, c)th element of the matrix $M[w, c]$. Next, the matrix M is factorized such that $M[w, c] \approx \mathbf{w}^T\mathbf{c}$, where $\mathbf{w}$ is the low dimension embedding of wth word and $\mathbf{c}$ is the cth context embedding.

The statistic in $M[w, c]$ is usually related to pointwise mutual information: $PMI(w, c) = \log(\frac{P(w,c)}{P(w)\cdot P(c)})$, where $P(w, c) = \frac{\#(w,c)}{|D|}$, $|D| = \sum_{w\in V}\sum_{c\in V} \#(w, c)$ is number of co-occurrence word pairs in the corpus, $P(w) = \frac{\#(w)}{|D|}$, $\#(w) = \sum_{c\in V} \#(w, c)$ is the frequency of the word w times the window size $|W|$, and similarly for $P(c)$. For example, $M[w, c]$ could be set as positive PMI (PPMI), $\max(PMI(w, c), 0)$, or shifted PMI, $PMI(w, c) - \log(k)$, like skip-grams with negative sampling (SGNS) (Levy et al., 2015a). Intuitively, since $M[w, c] \approx \mathbf{w}^T\mathbf{c}$, larger embedding values of $\mathbf{w}$ at every dimension seems to imply larger $\mathbf{w}^T\mathbf{c}$, larger $M[w, c]$, larger $PMI(w, c)$, and thus larger co-occurrence count $\#(w, c)$. However, the derivation has two flaws: (1) $\mathbf{c}$ could be negative and (2) lower $\#(w, c)$ could still lead to larger $PMI(w, c)$ as long as the $\#(w)$ is small enough.

To preserve DIH, we propose a novel word embedding method, *distributional inclusion vector embedding (DIVE)*, which fixes the two flaws by performing non-negative factorization (NMF) on the matrix M, where

$$M[w, c] = \log(\frac{P(w, c)}{P(w) \cdot P(c)} \cdot \frac{\#(w)}{k \cdot Z}) = \log(\frac{\#(w, c)|V|}{\#(c)k}), \tag{3}$$

where k is a constant which shifts PMI value like SGNS, $Z = \frac{|D|}{|V|}$ is the average word frequency, and $|V|$ is the vocabulary size.

The design encourages the inclusion property in DIVE (i.e. Equation (2)) to be satisfied because the property implies that Equation (1) (DIH) holds if the matrix is reconstructed perfectly. The derivation is simple: Since context vector $\mathbf{c}$ is non-negative, if the embedding of hypernym $\mathbf{y}$ is greater than or equal to the embedding of its hyponym $\mathbf{x}$ in every dimension, $\mathbf{x}^T\mathbf{c} \leq \mathbf{y}^T\mathbf{c}$. Then, $M[x, c] \leq M[y, c]$ tends to be true because $\mathbf{w}^T\mathbf{c} \approx M[w, c]$. This leads to $\#(x, c) \leq \#(y, c)$ because $M[w, c] = \log(\frac{\#(w,c)|V|}{\#(c)k})$ and only $\#(w, c)$ change with w.

2.2 OPTIMIZATION

Due to its appealing scalability properties during training time (Levy et al., 2015a), we optimize our embedding based on the skip-gram with negative sampling (SGNS) (Mikolov et al., 2013). The objective function of SGNS is

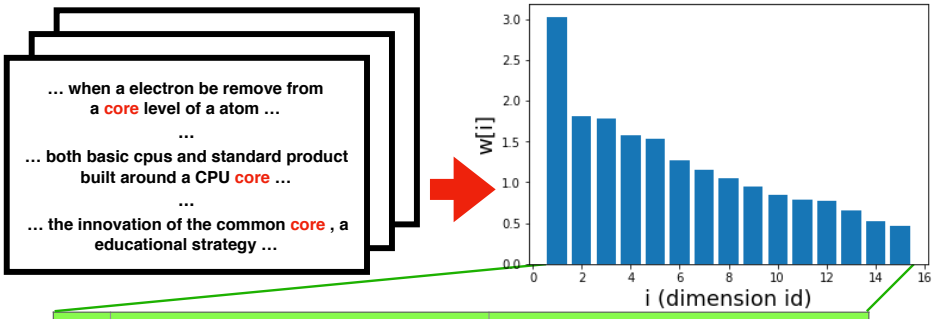
$$l_{SGNS} = \sum_{w\in V}\sum_{c\in V} \#(w, c) \log\sigma(\mathbf{w}^T\mathbf{c}) \; + \sum_{w\in V} k' \sum_{c\in V} \#(w, c) \mathop{\mathbb{E}}_{c_N\sim P_D}[\log\sigma(-\mathbf{w}^T\mathbf{c_N})], \tag{4}$$

where $\mathbf{w} \in \mathbb{R}$, $\mathbf{c} \in \mathbb{R}$, $\mathbf{c_N} \in \mathbb{R}$, k' is a constant hyper-parameter indicating the ratio between positive and negative samples.

Levy & Goldberg (2014) prove SGNS is equivalent to factorizing a shifted PMI matrix M', where $M'[w, c] = \log(\frac{P(w,c)}{P(w)\cdot P(c)} \cdot \frac{1}{k'})$. By setting $k' = \frac{k\cdot Z}{\#(w)}$ and applying non-negativity constraints to the embeddings, DIVE can be optimized using the similar objective function:

$$l_{DIVE} = \sum_{w\in V}\sum_{c\in V} \#(w, c) \log\sigma(\mathbf{w}^T\mathbf{c}) \; + k \sum_{w\in V} \frac{Z}{\#(w)} \sum_{c\in V} \#(w, c) \mathop{\mathbb{E}}_{c_N\sim P_D}[\log\sigma(-\mathbf{w}^T\mathbf{c_N})], \tag{5}$$

where $\mathbf{w} \geq 0, \mathbf{c} \geq 0, \mathbf{c_N} \geq 0$, σ is the logistic sigmoid function, and k is a constant hyper-parameter. P_D is the distribution of negative samples, which we set to be the corpus word frequency distribution in this paper. Equation (5) is optimized by ADAM (Kingma & Ba, 2015), a variant of stochastic gradient descent (SGD). The non-negativity constraint is implemented by projection (i.e., clipping any embedding which crosses the zero boundary after an update).



id	Top 1-5 words	Top 101-105 words
1	element, gas, atom, rock, carbon	methane, crystalline, complex, surround, proton
2	system, architecture, develop, base, language	functional, requirement, processing, compatible, api
3	also, well, several, early, see	fall, eventually, main, rise, mostly
4	part, almost, see, addition, except	incorporate, stage, instead, opening, add
5	star, orbit, sun, orbital, planet	bright, position, centauri, interaction, universe
6	several, main, province, include, consist	designate, exist, swiss, branch, thai
7	science, philosophy, theory, philosopher, term	ethical, advocate, topic, basic, universe
8	version, game, release, original, file	cassette, virtual, code, project, kb
9	electron, current, electric, circuit, voltage	anode, wire, ac, perform, resistor
10	tank, cylinder, wheel, engine, steel	aluminum, automatic, pilot, prevent, remove
11	school, university, student, education, college	doctorate, doctoral, middle, arts, compulsory
12	network, user, server, datum, protocol	technology, rout, agent, microsoft, command
13	high, low, temperature, energy, speed	atmospheric, fast, blood, m, population
14	acid, carbon, product, use, zinc	ph, monoxide, phosphorus, bond, manufacture
15	access, need, require, allow, program	size, ability, format, run, typically

Figure 1: The top 15 dimensions in the *distributional inclusion vector embedding (DIVE)* of the word core trained by the co-occurrence statistics of context words. The index of dimensions is sorted by the embedding values. The words in each row of the table are sorted by its embedding value in the dimension.

The optimization process provides an alternative angle to explain how DIVE preserves DIH. The gradients for the word embedding $\mathbf{w}$ is

$$\frac{dl_{DIVE}}{d\mathbf{w}} = \sum_{c \in V} \#(w,c)(1 - \sigma(\mathbf{w}^T\mathbf{c}))\mathbf{c} \; - k \sum_{c_N \in V} \frac{\#(c_N)}{|V|}\sigma(\mathbf{w}^T\mathbf{c_N})\mathbf{c_N}. \tag{6}$$

Assume hyponym x and hypernym y satisfy DIH in Equation (1) and the embeddings $\mathbf{x}$ and $\mathbf{y}$ are the same at some point during the gradient ascent. In the case, the gradients coming from negative sampling (the second term) decrease the same amount of embedding values for both x and y because k is a constant hyper-parameter. However, the embedding of hypernym $\mathbf{y}$ would get higher or equal positive gradients from the first term than $\mathbf{x}$ in every dimension because $\#(x,c) \leq \#(y,c)$. This means Equation (1) tends to imply Equation (2). Combining the analysis from the matrix factorization viewpoint, DIH in Equation (1) is approximately equivalent to the inclusion property in DIVE (i.e. Equation (2)).

2.3 PMI Filtering

For a frequent target word, there must be many neighboring words that incidentally appear near the target word without being semantically meaningful, especially when a large context window size is used. The unrelated context words cause noise in both the word vector and the context vector of DIVE. We address this issue by filtering out context words c for each target word w when the PMI of the co-occurring words is too small (i.e., $\log(\frac{P(w,c)}{P(w)\cdot P(c)}) < \log(k_f)$). That is, we set $\#(w,c) = 0$ in the objective function. This preprocessing step is similar with computing PPMI in SBOW (Bullinaria & Levy, 2007), where low PMI co-occurrences are removed from the count-based representation.

2.4 INTERPRETABILITY

After applying the non-negativity constraint, we observe that each dimension roughly corresponds to a topic, as previous findings suggest (Pauca et al., 2004; Murphy et al., 2012). This gives rise to a natural and intuitive interpretation of our word embeddings: the word embeddings can be seen as unnormalized probability distributions over topics. By removing the normalization of the target word frequency in the shifted PMI matrix, specific words have values in few dimensions (topics), while general words appear in more topics and correspondingly have high values in more dimensions, so the concreteness level of two words can be easily compared using the magnitude of their embeddings. In other words, general words have more diverse context distributions, so we need more dimensions to store the information in order to compress SBOW well (Nalisnick & Ravi, 2015).

In Figure 1, we present three mentions of the word core and its surrounding contexts. These various context words increase the embedding values in different dimensions. Each dimension of the learned embeddings roughly corresponds to a topic, and the more general or representative words for each topic tend to have the higher value in the corresponding dimension (e.g. words in the second column of the table). The embedding is able to capture the common contexts where the word core appears. For example, the context of the first mention is related to the atom topic (dimension id 1) and the electron topic (id 9), while the second and third mention occur in the computer architecture topic (id 2) and education topic (id 11), respectively.

3 EXPERIMENT SETUP

We describe four experiments in Section 4-7. The first 3 experiments compare DIVE with other unsupervised embeddings and SBOW using different hypernymy scoring functions. In these experiments, unsupervised approaches refer to the methods that only train on plaintext corpus without using any hypernymy or lexicon annotation. The last experiment presents qualitative results on word sense disambiguation.

3.1 DATASETS AND TESTING SETUP

The SBOW and embeddings are tested on 11 datasets. The first 4 datasets come from the recent review of Santus et al. (2017): BLESS (Baroni & Lenci, 2011), EVALution (Santus et al., 2015), Lenci/Benotto (Benotto, 2015), and Weeds (Weeds et al., 2014). The next 4 datasets are downloaded from the code repository of the H-feature detector (Roller & Erk, 2016): Medical (i.e., Levy 2014) (Levy et al., 2014), LEDS (also referred to as ENTAILMENT or Baroni 2012) (Baroni et al., 2012), TM14 (i.e., Turney 2014) (Turney & Mohammad, 2015), and Kotlerman 2010 (Kotlerman et al., 2010). In addition, the performance on the test set of HyperNet (Shwartz et al., 2016) (using the random train/test split), the test set of WordNet (Vendrov et al., 2016), and all pairs in HyperLex (Vulić et al., 2016) are also evaluated.

The F1 and accuracy measurements are sometimes very similar even though the quality of prediction varies, so average precision AP@all is adopted as the main evaluation metric. The HyperLex dataset has a continuous score on each candidate word pair, so we adopt Spearman rank coefficient ρ as suggested by the review study of Vulić et al. (2016). Any OOV (out-of-vocabulary) word encountered in the testing data is pushed to the bottom of the prediction list (effectively assuming the word pair does not have a hypernym relation).

3.2 TRAINING SETUP

We use WaCkypedia corpus (Baroni et al., 2009), a 2009 Wikipedia dump, to compute SBOW and train the embedding. For the datasets without Part of Speech (POS) information (i.e. Medical, LEDS, TM14, Kotlerman 2010, and HyperNet), the training data of SBOW and embeddings are raw text. For other datasets, we concatenate each token with the Part of Speech (POS) of the token before training the models except the case when we need to match the training setup of another paper.

All words are lower cased. Stop words and rare words (occurs less than 10 times) are removed during our preprocessing step. The number of embedding dimensions in DIVE d_0 is set to be 100. Other hyper-parameters used in the experiments are listed in the supplementary materials. The

Table 1: Comparison with previous unsupervised embeddings. All values are percentages. AP@all (%) for 10 datasets and Spearman ρ (%) for HyperLex. Word2Vec+C scores the word pairs using the cosine similarity on skip-grams (SGNS). GE+C and GE+KL computes cosine similarity and negative KL divergence on Gaussian embedding, respectively.

Dataset	BLESS	EVALution	LenciBenotto	Weeds	Medical	LEDS
Random	5.3	26.6	41.2	51.4	8.5	50.5
Word2Vec + C	9.2	25.4	40.8	51.6	11.2	71.8
GE + C	10.5	26.7	43.3	52.0	14.9	69.7
GE + KL	7.6	29.6	45.1	51.3	15.7	64.6
DIVE + C·ΔS	**16.3**	**33.0**	**50.4**	**65.5**	**25.3**	**83.5**

Dataset	TM14	Kotlerman 2010	HyperNet	WordNet	HyperLex
Random	52.0	30.8	24.5	55.2	0
Word2Vec + C	52.1	**39.5**	20.7	**63.0**	16.3
GE + C	53.9	36.0	21.6	58.2	16.4
GE + KL	52.0	39.4	23.7	54.4	9.6
DIVE + C·ΔS	**57.2**	36.6	**41.9**	60.9	**32.8**

hyper-parameters of DIVE were decided based on the performance of HyperNet training set. To train embeddings more efficiently, we chunk the corpus into subsets/lines of 100 tokens instead of using sentence segmentation. Preliminary experiments show that this implementation simplification does not hurt the performance.

In the following experiments, we train both SBOW and DIVE on only the first 512,000 lines (51.2 million tokens) because we find this way of training setting provides better performances (for both SBOW and DIVE) than training on the whole WaCkypedia or training on randomly sampled 512,000 lines. We suspect this is due to the corpus being sorted by the Wikipedia page titles, which makes some categorical words such as animal and mammal occur 3-4 times more frequently in the first 51.2 million tokens than the rest. The performances of training SBOW PPMI on the whole WaCkypedia is also provided for reference in Table 4 and Table 5.

4 Experiment 1: Comparison with Unsupervised Embeddings

If a pair of words has the hypernym relation, the words tend to be similar and the hypernym should be more general than the hyponym. As in HyperScore (Nguyen et al., 2017), we score the hypernym candidates by multiplying two factors corresponding to these properties. The C·ΔS (i.e. the cosine similarity multiply the difference of summation) scoring function is defined as

$$C \cdot \Delta S(\mathbf{w}_q \rightarrow \mathbf{w}_p) = \frac{\mathbf{w}_q^T \mathbf{w}_p}{||\mathbf{w}_q||_2 \cdot ||\mathbf{w}_p||_2} \cdot (\|\mathbf{w}_p\|_1 - \|\mathbf{w}_q\|_1), \tag{7}$$

where $\mathbf{w}_p$ is the embedding of hypernym and $\mathbf{w}_q$ is the embedding of hyponym.

As far as we know, Gaussian embedding (GE) is the only unsupervised embedding method which can capture the asymmetric relations between a hypernym and its hyponyms. Using the same training and testing setup, we use the code implemented by Athiwaratkun & Wilson (2017)[1] to train Gaussian embedding on the first 51.2 million tokens and test the embeddings on 11 datasets. Its hyper-parameters are determined using the same way as DIVE (i.e. maximizing the AP on HyperNet training set). We compare DIVE with GE[2] in Table 1, and the performances of random scores and only measuring word similarity using skip-grams are also presented for reference. As we can see, DIVE is usually significantly better than other baselines.

5 Experiment 2: Hypernymy Scoring Functions Analysis

In Experiment 1, we show that there exists a scoring function (C·ΔS) which detects hypernymy accurately using the embedding space of DIVE. Nevertheless, different scoring functions measure

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

[2] Notice that higher AP is reported in the previous literature: 80 (Vilnis & McCallum, 2015) in LEDS, 74.2 (Athiwaratkun & Wilson, 2017) in LEDS, and 20.6 (Vulić et al., 2016) in HyperLex. The difference might be caused by different training and testing setup (e.g. their hyper-parameters might be determined by (parts of) testing dataset).

different signals in SBOW or embeddings. Since there are so many scoring functions and datasets available in the domain, we first introduce and test the performances of various scoring functions so as to select the representative ones for a more comprehensive evaluation of DIVE on the hypernymy detection tasks. We denote the embedding/context vector of the hypernym candidate and the hyponym candidate as $\mathbf{w}_p$ and $\mathbf{w}_q$, respectively. The SBOW model which represents a word by the frequency of its neighboring words is denoted as SBOW Freq, while the SBOW which uses PPMI of its neighboring words as the features (Bullinaria & Levy, 2007) is denoted as SBOW PPMI.

5.1 Unsupervised Scoring Functions

5.1.1 Similarity

A hypernym tends to be similar to its hyponym, so we measure the cosine similarity between word vectors of the SBOW features (Levy et al., 2015b) or DIVE. We refer to the symmetric scoring function as Cosine or C for short in the following tables. We also train the original skip-grams with 100 dimensions and measure the cosine similarity between the resulting word2vec embeddings. This scoring function is referred to as Word2vec or W.

5.1.2 Generality

The *distributional informativeness hypothesis* (Santus et al., 2014) observes that in many corpora, semantically 'general' words tend to appear more frequently and in more varied contexts. Thus, Santus et al. (2014) advocate using entropy of context distributions to capture the diversity of context. We adopt the two variations of the approach proposed by Santus et al. (2017): SLQS Row and SLQS Sub functions. We also refer to SLQS Row as ΔE because it measures the entropy difference of context distributions. For SLQS Sub, the number of top context words is fixed as 100.

Although effective at measuring diversity, the entropy totally ignores the frequency signal from the corpus. To leverage the information, we measure the generality of a word by its L1 norm ($|\mathbf{w}_p|_1$) and L2 norm ($||\mathbf{w}_p||_2$). Recall that Equation (2) indicates that the embedding of the hypernym $\mathbf{y}$ should have a larger value at every dimension than the embedding of the hyponym $\mathbf{x}$. When the inclusion property holds, $|\mathbf{y}|_1 = \sum_i \mathbf{y}[i] \geq \sum_i \mathbf{x}[i] = |\mathbf{x}|_1$ and similarly $||\mathbf{y}||_2 \geq ||\mathbf{x}||_2$. Thus, we propose two scoring functions, difference of vector summation ($|\mathbf{w}_p|_1 - |\mathbf{w}_q|_1$) and the difference of vector 2-norm ($||\mathbf{w}_p||_2 - ||\mathbf{w}_q||_2$). Notice that when applying the difference of vector summations (denoted as ΔS) to SBOW Freq, it is equivalent to computing the word frequency difference between the hypernym candidate pair.

5.1.3 Combination

The combination of 2 similarity functions (Cosine and Word2vec) and the 3 generality functions (difference of entropy, summation, and 2-norm of vectors) leads to six different scoring functions as shown in Table 2, and C·ΔS is the same scoring function we used in Experiment 1. It should be noted that if we use skip-grams with negative sampling (word2vec) as the similarity measurement (i.e., $W \cdot \Delta$ {E,S,Q}), the scores are determined by two embedding/feature spaces together (word2vec and DIVE/SBOW).

5.1.4 Inclusion

Several scoring functions are proposed to measure inclusion properties of SBOW based on DIH. Weeds Precision (Weeds & Weir, 2003) and CDE (Clarke, 2009) both measure the magnitude of the intersection between feature vectors ($|\mathbf{w}_p \cap \mathbf{w}_q|$). For example, $\mathbf{w}_p \cap \mathbf{w}_q$ is defined by the element-wise minimum in CDE. Then, both scoring functions divide the intersection by the magnitude of the potential hyponym vector ($|\mathbf{w}_q|$). invCL (Lenci & Benotto, 2012) (A variant of CDE) is also tested.

We choose these 3 functions because they have been shown to detect hypernymy well in a recent study (Santus et al., 2017). However, it is hard to confirm that their good performances come from the inclusion property between context distributions — it is also possible that the context vectors of more general words have higher chance to overlap with all other words due to their high frequency. For instance, considering a one dimension feature which stores only the frequency of words, the naive embedding could still have reasonable performance on the CDE function, but the embedding

Table 2: Micro average AP@all (%) of 10 datasets using different scoring functions. The feature space is SBOW using word frequency.

Word2vec (W)	Cosine (C)	SLQS Sub	SLQS Row (ΔE)	Summation (ΔS)	Two norm (ΔQ)
24.8	26.7	27.4	27.6	31.5	31.2
W·ΔE	C·ΔE	W·ΔS	C·ΔS	W·ΔQ	C·ΔQ
28.8	29.5	**31.6**	31.2	31.4	31.1
Weeds	CDE	invCL	Asymmetric L1 (AL_1)		
19.0	31.1	30.7	28.2		

in fact only memorizes the general words without modeling relations between words (Levy et al., 2015b) and loses lots of inclusion signals in the word co-occurrence statistics.

In order to measure the inclusion property without the interference of the word frequency signal from the SBOW or embeddings, we propose a new measurement called asymmetric L_1 distance. We first get context distributions $\mathbf{d}_p$ and $\mathbf{d}_q$ by normalizing $\mathbf{w}_p$ and $\mathbf{w}_q$, respectively. Ideally, the context distribution of the hypernym $\mathbf{d}_p$ will include $\mathbf{d}_q$. This suggests the hypernym distribution $\mathbf{d}_p$ is larger than context distribution of the hyponym with a proper scaling factor $a\mathbf{d}_q$ (i.e., $\max(a\mathbf{d}_q - \mathbf{d}_p, 0)$ should be small). Furthermore, both distributions should be similar, so $a\mathbf{d}_q$ should not be too different from $\mathbf{d}_p$ (i.e., $\max(\mathbf{d}_p - a\mathbf{d}_q, 0)$ should also be small). Therefore, we define asymmetric L1 distance as

$$AL_1 = \min_a \sum_c w_0 \cdot \max(a\mathbf{d}_q[c] - \mathbf{d}_p[c], 0) + \max(\mathbf{d}_p[c] - a\mathbf{d}_q[c], 0), \tag{8}$$

where w_0 is a constant which emphasizes the inclusion penalty. If $w_0 = 1$ and $a = 1$, AL_1 is equivalent to L1 distance. The lower AL_1 distance implies a higher chance of observing the hypernym relation. We tried $w_0 = 5$ and $w_0 = 20$. $w_0 = 20$ produces a worse micro-average AP@all on SBOW Freq, SBOW PPMI and DIVE, so we fix w_0 to be 5 in all experiments. An efficient way to solve the optimization in AL_1 is presented in the supplementary materials.

5.2 Results and Discussions

We show the micro average AP@all on 10 datasets using different hypernymy scoring functions in Table 2. We can see the combination functions such as C·ΔS and W·ΔS perform the best overall. Among the unnormalized inclusion based scoring functions, CDE works the best. AL_1 performs well compared with other functions which remove the frequency signal such as Word2vec, Cosine, and SLQS Row. The summation is the most robust generality measurement. In the table, the scoring functions are applied to SBOW Freq, but the performances of hypernymy scoring functions on the other feature spaces (e.g. DIVE) have a similar trend.

6 Experiment 3: Comparison with SBOW

6.1 Comparison with previously reported results

In Table 3, DIVE with two of the best scoring functions (C·ΔS and W·ΔS) is compared with the previous unsupervised state-of-the-art approaches based on SBOW on different datasets.

There are several reasons which might cause the large performance gaps in some datasets. In addition to the effectiveness of DIVE, some improvements come from our proposed scoring functions. The fact that every paper uses a different training corpus also affects the performances. Furthermore, Santus et al. (2017) select the scoring functions and feature space for the first 4 datasets based on AP@100, which we believe is too sensitive to the hyper-parameter settings of different methods. To isolate the impact of each factor, we perform a more comprehensive comparison next.

6.2 Performance analysis

In this experiment, we examine whether DIVE successfully preserves the signals for hypernymy detection tasks, which are measured by the same scoring functions designed for SBOW. Summation difference (ΔS) and CDE perform the best among generality and inclusion functions in Table 2,

Table 3: Comparison with previous methods based on sparse bag of word (SBOW). All values are percentages. The results of invCL (Lenci & Benotto, 2012), APSyn (Santus et al., 2016), and CDE (Clarke, 2009) are selected because they have the best AP@100 in the first 4 datasets (Santus et al., 2017). Cosine similarity (Levy et al., 2015b), balAPinc (Kotlerman et al., 2010) in 3 datasets (Turney & Mohammad, 2015), SLQS (Santus et al., 2014) in HyperNet dataset (Shwartz et al., 2016), and Freq ratio (FR) (Vulić et al., 2016) are compared.

Dataset	BLESS	EVALution	LenciBenotto	Weeds	Medical
Metric	AP@all				F1
Baselines	invCL		APSyn	CDE	Cosine
	5.1	**35.3**	38.2	44.1	23.1
DIVE + C·ΔS	16.3	33.0	50.4	65.5	25.3
DIVE + W·ΔS	**18.6**	32.3	**51.5**	**68.6**	**25.7**

Dataset	LEDS	TM14	Kotlerman 2010	HyperNet	HyperLex
Metric	AP@all			F1	Spearman ρ
Baselines	balAPinc			SLQS	Freq ratio
	73	56	37	22.8	27.9
DIVE + C·ΔS	83.5	57.2	36.6	**41.9**	32.8
DIVE + W·ΔS	**86.4**	**57.3**	**37.4**	38.6	**33.3**

respectively. AL_1 could be used to examine the inclusion properties after removing the frequency signal. Therefore, we will present the results using these 3 scoring functions, along with W·ΔS and C·ΔS.

6.2.1 BASELINES

In addition to classic representations such as SBOW Freq and SBOW PPMI, we compare *distributional inclusion vector embedding* (DIVE) with additional 4 baselines in Table 4.

- SBOW PPMI with additional frequency weighting (PPMI w/ FW). Specifically, $\mathbf{w}[c] = \max(\log(\frac{P(w,c)}{P(w)*P(c)*\frac{Z}{\#(w)}}), 0)$. This forms the matrix reconstructed by DIVE when $k = 1$.
- DIVE without the PMI filter (DIVE w/o PMI)
- NMF on shifted PMI: Non-negative matrix factorization (NMF) on the shifted PMI without frequency weighting for DIVE (DIVE w/o FW). This is the same as applying the non-negative constraint on the skip-gram model.
- K-means (Freq NMF): The method first uses Mini-batch k-means (Sculley, 2010) to cluster words in skip-gram embedding space into 100 topics, and hashes each frequency count in SBOW into the corresponding topic. If running k-means on skip-grams is viewed as an approximation of clustering the SBOW context vectors, the method can be viewed as a kind of NMF (Ding et al., 2005).

  Let the $N \times N$ context matrix be denoted as M_c, where the (i, j)th element stores the count of word j appearing beside word i. K-means hashing creates a $N \times 100$ matrix G with orthonormal rows ($G^T G = I$), where the (i, k)th element is 0 if the word i does not belong to cluster k. The orthonormal G is also an approximated solution of a type of NMF ($M_c \approx FG^T$) (Ding et al., 2005). Hashing context vectors into topic vectors can be written as $M_c G \approx FG^T G = F$.

In the experiment, we also tried to apply a constant $\log(k)$ shifting to SBOW PPMI (i.e. $\max(PMI - \log(k), 0)$). We found that the performance degrades as k increases. Similarly, applying PMI filter to SBOW PPMI (set context feature to be 0 if the value is lower than $\log(k_f)$) usually makes the performances worse, especially when k_f is large. Applying PMI filter to SBOW Freq only makes its performances closer to (but still much worse than) SBOW PPMI, so we omit this baseline as well.

6.2.2 RESULTS AND DISCUSSIONS

In Table 4, we first confirm the finding of the previous review study of Santus et al. (2017): there is no single hypernymy scoring function which always outperforms others. One of the main reasons is that different datasets collect negative samples differently. This is also why we evaluate our

Table 4: AP@all (%) of 10 datasets. The box at lower right corner compares the micro average AP across all 10 datasets. Numbers in different rows come from different feature or embedding spaces. Numbers in different columns come from different datasets and unsupervised scoring functions. We also present the micro average AP across the first 4 datasets (BLESS, EVALution, Lenci/Benotto and Weeds). All wiki means SBOW using PPMI features trained on the whole WaCkypedia. FW refers to frequency weighting on the shifted PMI matrix.

AP@all (%)		BLESS					EVALution					Lenci/Benotto				
		CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW	Freq	6.3	7.3	5.6	11.0	5.9	35.3	**32.6**	36.2	**33.0**	36.3	51.8	**47.6**	51.0	51.8	51.1
	PPMI	**13.6**	5.1	5.6	17.2	15.3	30.4	27.7	34.1	31.9	34.3	47.2	39.7	50.8	51.1	**52.0**
	PPMI w/ FW	6.2	5.0	5.5	12.4	5.8	**36.0**	27.5	**36.3**	32.9	**36.4**	**52.0**	43.1	50.9	**51.9**	50.7
	All wiki	12.1	5.2	6.9	12.5	13.4	28.5	27.1	30.3	29.9	31.0	47.1	39.9	48.5	48.7	51.1
DIVE	Full	9.3	**7.6**	6.0	**18.6**	**16.3**	30.0	27.5	34.9	32.3	33.0	46.7	43.2	**51.3**	51.5	50.4
	w/o PMI	7.8	6.9	5.6	16.7	7.1	32.8	32.2	35.7	32.5	35.4	47.6	44.9	50.9	51.6	49.7
	w/o FW	9.0	6.2	**7.3**	6.2	7.3	24.3	25.0	22.9	23.5	23.9	38.8	38.1	38.2	38.2	38.4
Kmean (Freq NMF)		6.5	7.3	5.6	10.9	5.8	33.7	27.2	36.2	**33.0**	36.2	49.6	42.5	51.0	51.8	51.2

AP@all (%)		Weeds					Micro Average (4 datasets)					Medical				
		CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW	Freq	**69.5**	**58.0**	68.8	68.2	68.4	23.1	**21.8**	**22.9**	25.0	23.0	19.4	**19.2**	**14.1**	18.4	15.3
	PPMI	61.0	50.3	**70.3**	**69.2**	69.3	**24.7**	17.9	22.3	28.1	**27.8**	**23.4**	8.7	13.2	20.1	**24.4**
	PPMI w/ FW	67.6	52.2	69.4	68.7	67.7	23.2	18.2	**22.9**	25.8	22.9	22.8	10.6	13.7	18.6	17.0
	All wiki	61.3	48.6	70.0	68.5	**70.4**	23.4	17.7	21.7	24.6	25.8	22.3	8.9	12.2	17.6	21.1
DIVE	Full	59.2	55.0	69.7	68.6	65.5	22.1	19.8	22.8	**28.9**	27.6	11.7	9.3	13.7	**21.4**	19.2
	w/o PMI	60.4	56.4	69.3	68.6	64.8	22.2	21.0	22.7	28.0	23.1	10.7	8.4	13.3	19.8	16.2
	w/o FW	49.2	47.3	45.1	45.1	44.9	18.9	17.3	17.2	16.8	17.5	10.9	9.8	7.4	7.6	7.7
Kmean (Freq NMF)		69.4	51.1	68.8	68.2	68.9	22.5	19.3	**22.9**	24.9	23.0	12.6	10.9	14.0	18.1	14.6

AP@all (%)	LEDS					TM14					Kotlerman 2010				
	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW Freq	82.7	70.4	70.7	83.3	73.3	55.6	**53.2**	54.9	55.7	55.0	35.9	**40.5**	**34.5**	37.0	35.4
SBOW PPMI	**84.4**	50.2	72.2	**86.5**	**84.5**	**56.2**	52.3	54.4	57.0	**57.6**	**39.1**	30.9	33.0	37.0	36.3
All wiki	83.1	49.7	67.9	82.9	81.4	54.7	50.5	52.6	55.1	54.9	38.5	31.2	32.2	35.4	35.3
DIVE	83.3	**74.7**	**72.7**	86.4	83.5	55.3	52.6	**55.2**	**57.3**	57.2	35.3	31.6	33.6	**37.4**	**36.6**

AP@all (%)	HyperNet					WordNet					**Micro Average (10 datasets)**				
	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS	CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW Freq	**37.5**	**28.3**	46.9	**35.9**	**43.4**	56.6	55.2	55.5	56.2	55.6	**31.1**	**28.2**	31.5	31.6	31.2
SBOW PPMI	23.8	24.0	47.0	32.5	33.1	57.7	53.9	55.6	56.8	57.2	30.1	23.0	31.1	32.9	**33.5**
All wiki	23.0	24.5	40.5	30.5	29.7	57.4	53.1	56.0	56.4	57.3	29.0	23.1	29.2	30.2	31.1
DIVE	25.3	24.2	**49.3**	33.6	32.0	**60.2**	**58.9**	**58.4**	**61.1**	**60.9**	27.6	25.3	**32.1**	**34.1**	32.7

method on many datasets to make sure our conclusions hold in general. For example, if negative samples come from random word pairs (e.g. WordNet dataset), a symmetric similarity measure is already a pretty good scoring function. On the other hand, negative samples come from related or similar words in HyperNet, EVALution, Lenci/Benotto, and Weeds, so only computing generality difference leads to the best (or close to the best) performance. The negative samples in many datasets are composed of both random samples and similar words (such as BLESS), so the combination of similarity and generality difference yields the most stable results.

DIVE performs similar or better on all the scoring functions compared with SBOW consistently across all datasets in Table 4, while using many fewer dimensions (see Table 6). Its results on combination scoring functions outperform SBOW Freq. Meanwhile, its results on AL_1 outperform SBOW PPMI. The fact that combination scoring functions (i.e., W·ΔS or C·ΔS) usually outperform generality functions suggests that only memorizing general words is not sufficient. The best average performance on 4 and 10 datasets are both produced by W·ΔS on DIVE.

SBOW PPMI improves the combination functions from SBOW Freq but sacrifices AP on the inclusion functions. It generally hurts performance to change the frequency sampling of PPMI (PPMI w/ FW) or compute SBOW PPMI on the whole WaCkypedia (all wiki) instead of the first 51.2 million tokens. The similar trend can also be seen in Table 5. Note that AL_1 completely fails in HyperLex dataset using SBOW PPMI, which suggests that PPMI might not necessarily preserve the distributional inclusion property, even though it can have good performance on combination functions.

Removing the PMI filter from DIVE slightly drops the overall precision while removing frequency weights on shifted PMI (w/o FW) leads to poor performances. K-means (Freq NMF) produces similar AP compared with SBOW Freq, but has worse AL_1 scores. Its best AP scores on different datasets are also significantly worse than the best AP of DIVE. This means that only making word2vec (skip-grams with negative sampling) non-negative or naively accumulating topic distribution in contexts cannot lead to satisfactory embeddings.

Table 5: Spearman ρ (%) in HyperLex.

Spearman ρ (%)	HyperLex				
	CDE	AL_1	ΔS	W·ΔS	C·ΔS
SBOW Freq	**31.7**	**19.6**	27.6	29.6	27.3
SBOW PPMI	28.1	-2.3	**31.8**	**34.3**	**34.5**
All wiki	25.3	-2.2	28.0	30.5	31.0
DIVE	28.9	18.7	31.2	33.3	32.8

Table 6: The average number of non-zero dimensions across all testing words in 10 datasets.

SBOW Freq	SBOW PPMI	DIVE
5799	3808	**20**

Table 7: Spectral clustering on the non-zero DIVE dimensions of the query word. When the number of clusters is set to be 2, we present the top 4 dimensions in each cluster, which have the highest values on the query word embedding. Otherwise, the top 2 dimensions are presented. CID refers to cluster ID. The top 5 words with the highest values of each dimension are presented.

Query	CID	Top 5 words in the top dimensions	
rock	1	element, gas, atom, rock, carbon	sea, lake, river, area, water
		find, specie, species, animal, bird	point, side, line, front, circle
	2	band, song, album, music, rock	write, john, guitar, band, author
		early, work, century, late, begin	include, several, show, television, film
bank	1	county, area, city, town, west	several, main, province, include, consist
		building, build, house, palace, site	sea, lake, river, area, water
	2	money, tax, price, pay, income	company, corporation, system, agency, service
		united, states, country, world, europe	state, palestinian, israel, right, palestine
apple	1	food, fruit, vegetable, meat, potato	goddess, zeus, god, hero, sauron
		war, german, ii, germany, world	write, john, guitar, band, author
	2	version, game, release, original, file	car, company, sell, manufacturer, model
		system, architecture, develop, base, language	include, several, show, television, film
star	1	film, role, production, play, stage	character, series, game, novel, fantasy
		wear, blue, color, instrument, red	write, john, guitar, band, author
	2	element, gas, atom, rock, carbon	star, orbit, sun, orbital, planet
		give, term, vector, mass, momentum	light, image, lens, telescope, camera
tank	1	tank, cylinder, wheel, engine, steel	industry, export, industrial, economy, company
		acid, carbon, product, use, zinc	network, user, server, datum, protocol
	2	army, force, infantry, military, battle	aircraft, navy, missile, ship, flight
		however, attempt, result, despite, fail	war, german, ii, germany, world
race	1	win, world, cup, play, championship	two, one, three, four, another
	2	railway, line, train, road, rail	car, company, sell, manufacturer, model
	3	population, language, ethnic, native, people	female, age, woman, male, household
run	1	system, architecture, develop, base, language	access, need, require, allow, program
	2	railway, line, train, road, rail	also, well, several, early, see
	3	game, team, season, win, league	game, player, run, deal, baseball
tablet	1	bc, source, greek, ancient, date	book, publish, write, work, edition
	2	use, system, design, term, method	version, game, release, original, file
	3	system, blood, vessel, artery, intestine	patient, symptom, treatment, disorder, may

7 Experiment 4: Word Sense Disambiguation

In addition to hypernymy detection, Athiwaratkun & Wilson (2017) show that the mixture of Gaussian distributions can also be used to discover multiple senses of each word. In our qualitative experiment, we show that DIVE can achieve the similar goal without fixing the number of senses before training the embedding.

Recall that each dimension roughly corresponds to one topic. Given a query word, the higher embedding value on a dimension implies higher likelihood to observe the word in the context of the topic. The embedding of a polysemy would have high values on different groups of topics/dimensions. This allows us to discover the senses by clustering the topics/dimensions of the polysemy. We use the embedding values as the feature each dimension, compute the pairwise similarity between dimensions, and apply spectral clustering (Stella & Shi, 2003) to group topics as shown in the Table 7. See more implementation details in the supplementary materials.

In the word sense disambiguation tasks, it is usually challenging to determine how many senses/clusters each word should have. Many existing approaches fix the number of senses before training the embedding (Tian et al., 2014; Athiwaratkun & Wilson, 2017). Neelakantan et al. (2014) make the number of clusters approximately proportional to the diversity of the context, but the assumption does not always hold. Furthermore, the training process cannot capture different granularity of senses. For instance, race in the car context could share the same sense with the race in the game topic because they all mean contest, but the race in the car context actually refers to the specific contest of speed. Therefore, they can also be viewed as separate senses (like the results in Table 7). This means the correct number of clusters is not unique, and the methods, which fixes the cluster numbers, need to re-train the embedding many times to capture such granularity.

In our approach, clustering dimensions is done after the training process of DIVE is completed, so it is fairly efficient to change the cluster numbers and hierarchical clustering is also an option. Similar to our method, Pelevina et al. (2016) also discover word senses by graph-based clustering. The main difference is that they cluster the top n words which are most related to the query word instead of topics. However, choosing the hyper-parameter n is difficult. Large n would make graph clustering algorithm inefficient, while small n would make less frequent senses difficult to discover.

8 Related work

Most previous unsupervised approaches focus on designing better hypernymy scoring functions for sparse bag of word (SBOW) features. They are well summarized in the recent study (Santus et al., 2017). Santus et al. (2017) also evaluate the influence of different contexts, such as changing the window size of contexts or incorporating dependency parsing information, but neglect scalability issues inherent to SBOW methods.

A notable exception is the Gaussian embedding model (Vilnis & McCallum, 2015). The context distribution of each word is encoded as a multivariate Gaussian distribution, where the embeddings of hypernyms tend to have higher variance and overlap with the embedding of their hyponyms. However, since a Gaussian distribution is normalized, it is difficult to retain frequency information during the embedding process, and experiments on HyperLex (Vulić et al., 2016) demonstrate that a simple baseline only relying on word frequency can achieve good results. Follow-up work models contexts by a mixture of Gaussians (Athiwaratkun & Wilson, 2017) relaxing the unimodality assumption but achieves little improvement on hypernym detection tasks.

Kiela et al. (2015) show that images retrieved by a search engine can be a useful source of information to determine the generality of lexicons, but the resources might not be available for some corpora such as scientific literature.

Order embedding (Vendrov et al., 2016) is a supervised approach to encode many annotated hypernym pairs (e.g. all of the whole WordNet (Miller, 1995)) into a compact embedding space, where the embedding of a hypernym should be smaller than the embedding of its hyponym in every dimension. Our method learns embedding from raw text, where a hypernym embedding should be larger than the embedding of its hyponym in every dimension. Thus, DIVE can be viewed as an unsupervised and reversed form of order embedding.

Other semi-supervised hypernym detection methods aim to generalize from sets of annotated word pairs using raw text corpora. The goal of HyperScore (Nguyen et al., 2017) is similar to our model: the embedding of a hypernym should be similar to its hyponym but with higher magnitude. However, their training process relies heavily on annotated hypernym pairs, and the performance drops significantly when reducing the amount of supervision. In addition to context distributions, previous work also leverages training data to discover useful text pattern indicating is-a relation (Shwartz et al., 2016; Roller & Erk, 2016), but it remains challenging to increase recall of hypernym detection because commonsense facts like cat is-a animal might not appear in the corpus.

Non-negative matrix factorization (NMF) has a long history in NLP, for example in the construction of topic models (Pauca et al., 2004). Non-negative sparse embedding (NNSE) (Murphy et al., 2012) and Faruqui et al. (2015) indicate that non-negativity can make embeddings more interpretable and improve word similarity evaluations. The sparse NMF is also shown to be effective in cross-lingual lexical entailment tasks but does not necessarily improve monolingual hypernymy detection (Vyas &

Carpuat, 2016). In our study, a new type of NMF is proposed, and the comprehensive experimental analysis demonstrates its state-of-the-art performances on unsupervised hypernymy detection.

9 CONCLUSIONS

Compressing unsupervised SBOW models into a compact representation is challenging while preserving the inclusion, generality, and similarity signals which are important for hypernym detection. Our experiments suggest that simple baselines such as accumulating K-mean clusters and non-negative skip-grams do not lead to satisfactory performances in this task.

To achieve this goal, we proposed an interpretable and scalable embedding method called *distributional inclusion vector embedding (DIVE)* by performing non-negative matrix factorization (NMF) on a weighted PMI matrix. We demonstrate that scoring functions which measure inclusion and generality properties in SBOW can also be applied to DIVE to detect hypernymy, and DIVE performs the best on average, slightly better than SBOW while using many fewer dimensions.

Our experiments also indicate that unsupervised scoring functions, which combine similarity and generality measurements, work the best in general, but no one scoring function dominates across all datasets. A combination of unsupervised DIVE with the proposed scoring functions produces new state-of-the-art performances on many datasets under the unsupervised setup.

Finally, a qualitative experiment shows that clusters of the topics discovered by DIVE often correspond to the word senses, which allow us to do word sense disambiguation without the need to know the number of senses before training the embeddings.

REFERENCES

Ben Athiwaratkun and Andrew Gordon Wilson. Multimodal word distributions. In *ACL*, 2017.

Marco Baroni and Alessandro Lenci. How we BLESSed distributional semantic evaluation. In *Workshop on GEometrical Models of Natural Language Semantics (GEMS)*, 2011.

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

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *EACL*, 2012.

Giulia Benotto. Distributional models for semantic relations: A study on hyponymy and antonymy. *PhD Thesis, University of Pisa*, 2015.

John A Bullinaria and Joseph P Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526, 2007.

Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res.(JAIR)*, 24(1):305–339, 2005.

Daoud Clarke. Context-theoretic semantics for natural language: an overview. In *workshop on geometrical models of natural language semantics*, pp. 112–119, 2009.

Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Lifted rule injection for relation embeddings. In *EMNLP*, 2016.

Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *ICDM*, 2005.

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

Maayan Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *ACL*, 2005.

Goran Glavaš and Simone Paolo Ponzetto. Dual tensor model for detecting asymmetric lexico-semantic relations. In *EMNLP*, 2017.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. Question classification using head words and their hypernyms. In *EMNLP*, 2008.

Douwe Kiela, Laura Rimell, Ivan Vulic, and Stephen Clark. Exploiting image generality for lexical entailment detection. In *ACL*, 2015.

Diederik Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. In *ICLR*, 2015.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389, 2010.

Alessandro Lenci and Giulia Benotto. Identifying hypernyms in distributional semantic spaces. In *SemEval*, 2012.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, 2014.

Omer Levy, Ido Dagan, and Jacob Goldberger. Focused entailment graphs for open IE propositions. In *CoNLL*, 2014.

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

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. Do supervised distributional methods really learn lexical inference relations? In *NAACL-HTL*, 2015b.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

George A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

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

Eric Nalisnick and Sachin Ravi. Infinite dimensional word embeddings. *arXiv preprint arXiv:1511.05392*, 2015.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*, 2014.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. Hierarchical embeddings for hypernymy detection and directionality. In *EMNLP*, 2017.

Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *NIPS*, 2017.

V. Paul Pauca, Farial Shahnaz, Michael W Berry, and Robert J. Plemmons. Text mining using non-negative matrix factorizations. In *ICDM*, 2004.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. Making sense of word embeddings. In *Workshop on Representation Learning for NLP*, 2016.

Simone Paolo Ponzetto and Michael Strube. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *ACL*, 2006.

Stephen Roller and Katrin Erk. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *EMNLP*, 2016.

Mark Sammons, V Vydiswaran, and Dan Roth. Recognizing textual entailment. *Multilingual Natural Language Applications: From Theory to Practice. Prentice Hall, Jun*, 2011.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte Im Walde. Chasing hypernyms in vector spaces with entropy. In *EACL*, 2014.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. EVALution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Workshop on Linked Data in Linguistics (LDL)*, 2015.

Enrico Santus, Tin-Shing Chiu, Qin Lu, Alessandro Lenci, and Chu-Ren Huang. Unsupervised measure of word similarity: how to outperform co-occurrence and vector cosine in vsms. *arXiv preprint arXiv:1603.09054*, 2016.

Enrico Santus, Vered Shwartz, and Dominik Schlechtweg. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *EACL*, 2017.

David Sculley. Web-scale k-means clustering. In *WWW*, 2010.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. Improving hypernymy detection with an integrated path-based and distributional method. In *ACL*, 2016.

X Yu Stella and Jianbo Shi. Multiclass spectral clustering. In *ICCV*, 2003.

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

Peter D Turney and Saif M Mohammad. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 21(3):437–476, 2015.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. In *ICLR*, 2016.

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. In *ICLR*, 2015.

Ivan Vulić and Nikola Mrkšić. Specialising word vectors for lexical entailment. *arXiv preprint arXiv:1710.06371*, 2017.

Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. Hyperlex: A large-scale evaluation of graded lexical entailment. *arXiv preprint arXiv:1608.02117*, 2016.

Yogarshi Vyas and Marine Carpuat. Sparse bilingual word representations for cross-lingual lexical entailment. In *HLT-NAACL*, 2016.

Julie Weeds and David Weir. A general framework for distributional similarity. In *EMNLP*, 2003.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. Learning to distinguish hypernyms and co-hyponyms. In *COLING*, 2014.

Table 8: Comparison with semi-supervised embeddings (with limited training data). All values are percentages. The number in parentheses beside each approach indicates the number of annotated hypernymy word pairs used to train the model. Semi-supervised embeddings include HyperScore (Nguyen et al., 2017) and H-feature (Roller & Erk, 2016). When we compare F1 with the results from other papers, we use 20 fold cross validation to determine prediction thresholds, as done by Roller & Erk (2016). Note that HyperScore ignores POS in the testing data, so we follow the setup when comparing with it.

Dataset	HyperLex	EVALution	LenciBenotto	Weeds	Medical
Metric	Spearman ρ	AP@all			F1
Baselines	HyperScore (1337)				H-feature (897)
(#Training Hypernymy)	30	**39**	44.8	58.5	**26**
DIVE + C·ΔS (0)	**34.5**	33.8	**52.9**	**70.0**	25.3

Table 9: We show the top 30 words with the highest embedding magnitude after dot product with the query embedding $\mathbf{q}$ (i.e. showing w such that $|\mathbf{w}^T\mathbf{q}|_1$ is one of the top 30 highest values). The rows with the empty query word sort words based on $|\mathbf{w}|_1$.

Query	Top 30 general words					
	use	name	system	include	base	city
	large	state	group	power	death	form
	american	life	may	small	find	body
	design	work	produce	control	great	write
	study	lead	type	people	high	create
species	specie	species	animal	find	plant	may
	human	bird	genus	family	organism	suggest
	gene	tree	name	genetic	study	occur
	fish	disease	live	food	cell	mammal
	evidence	breed	protein	wild	similar	fossil
system	system	use	design	provide	operate	model
	standard	type	computer	application	develop	method
	allow	function	datum	device	control	information
	process	code	via	base	program	software
	network	file	development	service	transport	law

A SUPPLEMENTARY MATERIALS

A.1 COMPARISON WITH SEMI-SUPERVISED EMBEDDINGS

In addition to the unsupervised approach, we also compare DIVE with semi-supervised approaches. When there are sufficient training data, there is no doubt that the semi-supervised embedding approaches such as HyperNet (Shwartz et al., 2016), H-feature detector (Roller & Erk, 2016), and HyperScore (Nguyen et al., 2017) can achieve better performance than all unsupervised methods. However, in many domains such as scientific literature, there are often not many annotated hypernymy pairs (e.g. Medical dataset (Levy et al., 2014)).

Since we are comparing an unsupervised method with semi-supervised methods, it is hard to fairly control the experimental setups and tune the hyper-parameters. In Table 8, we only show several performances which are copied from the original paper when training data are limited[3]. As we can see, the performance from DIVE is roughly comparable to the previous semi-supervised approaches trained on small amount of hypernym pairs. This demonstrates the robustness of our approach and the difficulty of generalizing hypernymy annotations with semi-supervised approaches.

A.2 GENERALITY ESTIMATION AND HYPERNYM DIRECTIONALITY DETECTION

In Table 9, we show the most general words in DIVE under different queries as constraints. We also present the accuracy of judging which word is a hypernym (more general) given word pairs with

[3]We neglect the performances from models trained on more than 10,000 hypernym pairs, models trained on the same datasets with more than 1000 hypernym pairs using cross-validation, and models using image classifier.

Table 10: Accuracy (%) of hypernym directionality prediction across 10 datasets.

Micro Average (10 datasets)			
SBOW Freq + SLQS Sub	SBOW Freq + $\triangle$S	SBOW PPMI + $\triangle$S	DIVE + $\triangle$S
64.4	66.8	66.8	**67.0**

Table 11: Dataset sizes. N denotes the number of word pairs in the dataset, and OOV shows how many word pairs are not processed by all the methods in Table 4 and Table 5.

BLESS		EVALution		Lenci/Benotto		Weeds		Avg (4 datasets)	
N	OOV	N	OOV	N	OOV	N	OOV	N	OOV
26554	1507	13675	2475	5010	1464	2928	643	48167	6089
Medical		LEDS		TM14		Kotlerman 2010		HyperNet	
N	OOV	N	OOV	N	OOV	N	OOV	N	OOV
12602	3711	2770	28	2188	178	2940	89	17670	9424
WordNet		Avg (10 datasets)		HyperLex					
N	OOV	N	OOV	N	OOV				
8000	3596	94337	24110	2616	59				

hypernym relations in Table 10. The direction is classified correctly if the generality score is greater than 0 (hypernym is indeed predicted as the more general word). For instance, summation difference (ΔS) classifies correctly if $|\mathbf{w}_p|_1 - |\mathbf{w}_q|_1 > 0$ ($|\mathbf{w}_p|_1 > |\mathbf{w}_q|_1$).

From the table, we can see that the simple summation difference performs better than SQLS Sub, and DIVE predicts directionality as well as SBOW. Notice that whenever we encounter OOV, the directionality is predicted randomly. If OOV is excluded, the accuracy of predicting directionality using unsupervised methods can reach around 0.7-0.75.

A.3 Experimental details for hypernymy detection

In HyperNet and WordNet, some hypernym relations are determined between phrases instead of words. Phrase embeddings are composed by averaging word embeddings or SBOW features. For WordNet, we assume the Part of Speech (POS) tags of the words are the same as the phrase. All part-of-speech (POS) tags in the experiments come from NLTK.

The window size $|W|$ of SBOW, DIVE, and GE are set as 20 (left 10 words and right 10 words). For DIVE, the number of epochs is 15, the learning rate is 0.001, the batch size is 128, the threshold in PMI filter k_f is set to be 30, and the ratio between negative and positive samples (k) is 1.5. The hyper-parameters of DIVE were decided based on the performance of HyperNet training set. The window size of skip-grams (word2vec) is 10. The number of negative samples (k') in skip-gram is set as 5. When composing skip gram into phrase embedding, average embedding is used.

For Gaussian embedding (GE), the number of mixture is 1, the number of dimension is 100, the learning rate is 0.01, the lowest variance is 0.1, the highest variance is 100, the highest Gaussian mean is 10, and other hyper-parameters are the default value in `https://github.com/benathi/word2gm`. The hyper-parameters of GE were also decided based on the performance of HyperNet training set. When determining the score between two phrases, we use the average score of every pair of tokens in two phrases.

The number of testing pairs N and the number of OOV word pairs is presented in Table 11.

A.4 Clustering dimensions for word sense disambiguation

We use all the default hyper-parameters of the spectral clustering library in Scikit-learn 0.18.2 (Pedregosa et al., 2011) except the number of clusters is set manually. A simple way to prepare the feature of each dimension $f(c_i)$ is to use the embedding values in that dimension $\mathbf{w}[c_i]$ of all the words in our vocabulary $w \in V$. That is,

$$f(c_i) = [\mathbf{w}[c_i]]_{w \in V}\,. \tag{9}$$

However, clustering on the global features might group topics together based on the co-occurrence of words which are unrelated to the query words and we want to make the similarity dependent on the

query word. For example, a country topic should be clustered together with a city topic if the query word is place, but it makes more sense to group the country topic with the money topic together if the query word is bank like we did in the word sense disambiguation experiment (Table 7). This means we want to focus on the geographical meaning of country when the query is related to geography, while focus on the economic meaning of country when the query is about economics.

To create query dependent similarity measurement, we only consider the embedding of words which are related to the query word when preparing the features of dimensions. Specifically, given a query word q, the feature vector of the ith dimension $f(c_i, q)$ is defined as:

$$f(c_i, q) = \bigoplus_{j=1}^{d_0} \left[\mathbf{w}[c_i] \cdot \mathbf{w}_q[c_j]\right]_{w \in C_j(n)}, \tag{10}$$

where $\mathbf{w}_q[c_j]$ is the value of jth dimension of query word embedding, $C_j(n)$ is the set of embeddings of top n words in the jth dimension, and the operator $\oplus$ means concatenation. This means instead of considering all the words in the vocabulary, we only take the top n words of every dimension j (n is fixed as 100 in the experiment), weight the feature based on how likely to observe query word in dimension j ($\mathbf{w}_q[c_j]$), and concatenate all features together. That is, when measuring the similarity of dimensions, we only consider the aspects related to query word (e.g. mostly considering words related to facility and money when the query word is bank).

After the features of all dimensions are collected, we normalize the feature of each dimension to have the norm 1, compute the pairwise similarity and run the spectral clustering to get the clustering results.

A.5 Efficient way to compute asymmetric L1 (AL_1)

Recall that Equation (8) defines AL_1 as follows:

$$AL_1 = \mathcal{L} = \min_a \sum_c w_0 \max(a\mathbf{d}_q[c] - \mathbf{d}_p[c], 0) + \max(\mathbf{d}_p[c] - a\mathbf{d}_q[c], 0),$$

where $\mathbf{d}_p[c]$ is one of dimension in the feature vector of hypernym $\mathbf{d}_p$, $a\mathbf{d}_q$ is the feature vector of hyponym after proper scaling. In Figure 2, an simple example is visualized to illustrate the intuition behind the distance function.

By adding slack variables ζ and ξ, the problem could be converted into a linear programming problem:

$$\begin{aligned}
\mathcal{L} = \min_{a,\zeta,\xi} \; & w_0 \sum_c \zeta_c + \sum_c \xi_c \\
& \zeta_c \geq a\mathbf{d}_q[c] - \mathbf{d}_p[c], \quad \zeta_c \geq 0 \\
& \xi_c \geq \mathbf{d}_p[c] - a\mathbf{d}_q[c], \quad \xi_c \geq 0 \\
& a \geq 0,
\end{aligned}$$

so it can be simply solved by a general linear programming library.

Nevertheless, the structure in the problem actually allows us to solve this optimization by a simple sorting. In this section, we are going to derive the efficient optimization algorithm.

By introducing Lagrangian multiplier for the constraints, we can rewrite the problem as

$$\begin{aligned}
\mathcal{L} = \min_{a,\zeta,\xi} \max_{\alpha,\beta,\gamma,\delta} \; & w_0 \sum_c \zeta_c + \sum_c \xi_c - \sum_c \alpha_c(\zeta_c - a\mathbf{d}_q[c] + \mathbf{d}_p[c]) \\
& - \sum_c \beta_c(\xi_c - \mathbf{d}_p[c] + a\mathbf{d}_q[c]) - \sum_c \gamma_c\zeta_c - \sum_c \delta_c\xi_c \\
\zeta_c \geq 0, \xi_c \geq 0, \alpha_c \geq 0, \; & \beta_c \geq 0, \gamma_c \geq 0, \delta_c \geq 0, a \geq 0
\end{aligned}$$

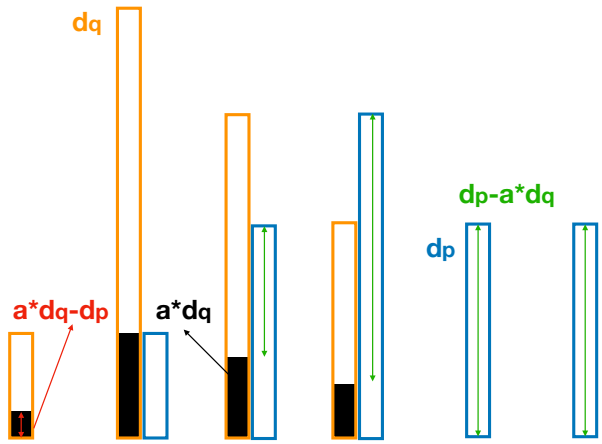


Figure 2: An example of AL_1 distance. If the word pair indeed has the hypernym relation, the context distribution of hyponym ($\mathbf{d}_q$) tends to be included in the context distribution of hypernym ($\mathbf{d}_p$) after proper scaling according to DIH. Thus, the context words only appear beside the hyponym candidate ($a\mathbf{d}_q[c] - \mathbf{d}_p[c]$) causes higher penalty.

First, we eliminate the slack variables by taking derivatives with respect to them:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \zeta_c} &= 0 = 1 - \beta_c - \delta_c \\
\delta_c &= 1 - \beta_c, \quad \beta_c \leq 1 \\
\frac{\partial \mathcal{L}}{\partial \xi_c} &= 0 = 1 - \gamma_c - \alpha_c \\
\gamma_c &= w_0 - \alpha_c, \quad \alpha_c \leq w_0.
\end{aligned}$$

By substituting in these values for γ_c and δ_c, we get rid of the slack variables and have a new Lagrangian:

$$\begin{gathered}
\mathcal{L} = \min_a \max_{\alpha,\beta} \; -\sum_c \alpha_c(-a\mathbf{d}_q[c] + \mathbf{d}_p[c]) - \sum_c \beta_c(-\mathbf{d}_p[c] + a\mathbf{d}_q[c]) \\
0 \leq \alpha_c \leq w_0, 0 \leq \beta_c \leq 1, a \geq 0
\end{gathered}$$

We can introduce a new dual variable $\lambda_c = \alpha_c - \beta_c + 1$ and rewrite this as:

$$\begin{gathered}
\mathcal{L} = \min_a \max_{\lambda} \; \sum_c (\lambda_c - 1)(a\mathbf{d}_q[c] - \mathbf{d}_p[c]) \\
0 \leq \lambda_c \leq w_0 + 1, a \geq 0
\end{gathered}$$

Let's remove the constraint on a and replace with a dual variable η:

$$\begin{gathered}
\mathcal{L} = \min_a \max_{\lambda} \; \sum_c (\lambda_c - 1)(a\mathbf{d}_q[c] - \mathbf{d}_p[c]) - \eta a \\
0 \leq \lambda_c \leq w_0 + 1, \eta \geq 0
\end{gathered}$$

Now let's differentiate with respect to a to get rid of the primal objective and add a new constraint:

$$\frac{\partial \mathcal{L}}{\partial a} = 0 = \sum_c \lambda_c \mathbf{d}_q[c] - \sum_c \mathbf{d}_q[c] - \eta$$
$$\sum_c \lambda_c \mathbf{d}_q[c] = \sum_c \mathbf{d}_q[c] + \eta$$
$$\mathcal{L} = \max_{\lambda} \sum_c \mathbf{d}_p[c] - \sum_c \lambda_c \mathbf{d}_p[c]$$
$$\sum_c \lambda_c \mathbf{d}_q[c] = \sum_c \mathbf{d}_q[c] + \eta$$
$$0 \leq \lambda_c \leq w_0 + 1, \eta \geq 0$$

Now we have some constant terms that are just the sums of $\mathbf{d}_p$ and $\mathbf{d}_q$, which will be 1 if they are distributions.

$$\mathcal{L} = \max_{\lambda} 1 - \sum_c \lambda_c \mathbf{d}_p[c]$$
$$\sum_c \lambda_c \mathbf{d}_q[c] = 1 + \eta$$
$$0 \leq \lambda_c \leq w_0 + 1, \eta \geq 0$$

Now we introduce a new set of variables $\mu_c = \lambda_c \mathbf{d}_q[c]$ and we can rewrite the objective as:

$$\mathcal{L} = \max_{\mu} 1 - \sum_c \mu_c \frac{\mathbf{d}_p[c]}{\mathbf{d}_q[c]}$$
$$\sum_c \mu_c = 1 + \eta$$
$$0 \leq \mu_c \leq (w_0 + 1)\mathbf{d}_q[c], \eta \geq 0$$

Note that for terms where $\mathbf{d}_q[c] = 0$ we can just set $\mathbf{d}_q[c] = \epsilon$ for some very small epsilon, and in practice, our algorithm will not encounter these because it sorts.

So μ we can think of as some fixed budget that we have to spend up until it adds up to 1, but it has a limit of how much we can spend for each coordinate, given by $(w_0 + 1)\mathbf{d}_q[c]$. Since we're trying to minimize the term involving μ, we want to allocate as much budget as possible to the smallest terms in the summand, and then 0 to the rest once we've spent the budget. This also shows us that our optimal value for the dual variable η is just 0 since we want to minimize the amount of budget we have to allocate.

To make presentation easier, lets assume we sort the vectors in order of increasing $\frac{\mathbf{d}_p[c]}{\mathbf{d}_q[c]}$, so that $\frac{\mathbf{d}_p[1]}{\mathbf{d}_q[1]}$ is the smallest element, etc. We can now give the following algorithm to find the optimal μ.

init $S = 0, c = 1, \mu = 0$
while $S \leq 1$:
  $\mu_c = \min(1 - S, (w_0 + 1)\mathbf{d}_q[c])$
  $S = S + \mu_c$
  $c = c + 1$

At the end we can just plug in this optimal μ to the objective to get the value of our scoring function.