A Vector Space for Distributional Semantics for Entailment *

James Henderson and **Diana Nicoleta Popa**
Xerox Research Centre Europe
james.henderson@xrce.xerox.com and diana.popa@xrce.xerox.com

Abstract

Distributional semantics creates vector-space representations that capture many forms of semantic similarity, but their relation to semantic entailment has been less clear. We propose a vector-space model which provides a formal foundation for a distributional semantics of entailment. Using a mean-field approximation, we develop approximate inference procedures and entailment operators over vectors of probabilities of features being known (versus unknown). We use this framework to reinterpret an existing distributional-semantic model (Word2Vec) as approximating an entailment-based model of the distributions of words in contexts, thereby predicting lexical entailment relations. In both unsupervised and semi-supervised experiments on hyponymy detection, we get substantial improvements over previous results.

1 Introduction

Modelling entailment is a fundamental issue in computational semantics. It is also important for many applications, for example to produce abstract summaries or to answer questions from text, where we need to ensure that the input text entails the output text. There has been a lot of interest in modelling entailment in a vector-space, but most of this work takes an empirical, often ad-hoc, approach to this problem, and achieving good results has been difficult (Levy et al., 2015). In this work, we propose a new framework for modelling entailment in a vector-space, and illustrate its effectiveness with a distributional-semantic model of hyponymy detection.

$\Rightarrow$	*unk*	f	g	$\neg f$
unk	1	0	0	0
f	1	1	0	0
g	1	0	1	0
$\neg f$	1	0	0	1

Table 1: Pattern of logical entailment between nothing known (*unk*), two different features f and g known, and the complement of f ($\neg f$) known.

Unlike previous vector-space models of entailment, the proposed framework explicitly models what information is unknown. This is a crucial property, because entailment reflects what information is and is not known; a representation y entails a representation x if and only if everything that is known given x is also known given y. Thus, we model entailment in a vector space where each dimension represents something we might know. As illustrated in Table 1, knowing that a feature f is true always entails knowing that same feature, but never entails knowing that a different feature g is true. Also, knowing that a feature is true always entails not knowing anything (*unk*), since strictly less information is still entailment, but the reverse is never true. Table 1 also illustrates that knowing that a feature f is false ($\neg f$) patterns exactly the same way as knowing that an unrelated feature g is true. This illustrates that the relevant dichotomy for entailment is known versus unknown, and not true versus false.

Previous vector-space models have been very successful at modelling semantic similarity, in particular using distributional semantic models (e.g. (Deerwester et al., 1990; Schütze, 1993; Mikolov et al., 2013a)). Distributional semantics uses the distributions of words in contexts to induce vector-space embeddings of words, which have been

*This work was partially supported by French ANR grant CIFRE N 1324/2014.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2052–2062,
Berlin, Germany, August 7-12, 2016.

shown to be useful for a wide variety of tasks. Two words are predicted to be similar if the dot product between their vectors is high. But the dot product is an anti-symmetric operator, which makes it more natural to interpret these vectors as representing whether features are true or false, whereas the dichotomy known versus unknown is asymmetric. We surmise that this is why distributional semantic models have had difficulty modelling lexical entailment (Levy et al., 2015).

To develop a vector-space model of whether features are known or unknown, we start with discrete binary vectors, where 1 means known and 0 means unknown. Entailment between these discrete binary vectors can be calculated by independently checking each dimension. But as soon as we try to do calculations with *distributions* over these vectors, we need to deal with the case where the features are not independent. For example, if feature f has a 50% chance of being true and a 50% chance of being false, we can't assume that there is a 25% chance that both f and $\neg f$ are known. This simple case of mutual exclusion is just one example of a wide range of constraints between features which we need to handle in semantic models. These constraints mean that the different dimensions of our vector space are not independent, and therefore exact models are not factorised. Because the models are not factorised, exact calculations of entailment and exact inference of vectors are intractable.

Mean-field approximations are a popular approach to efficient inference for intractable models. In a mean-field approximation, distributions over binary vectors are represented using a single probability for each dimension. These vectors of real values are the basis of our proposed vector space for entailment.

In this work, we propose a vector-space model which provides a formal foundation for a distributional semantics of entailment. This framework is derived from a mean-field approximation to entailment between binary vectors, and includes operators for measuring entailment between vectors, and procedures for inferring vectors in an entailment graph. We validate this framework by using it to reinterpret existing Word2Vec (Mikolov et al., 2013a) word embedding vectors as approximating an entailment-based model of the distribution of words in contexts. This reinterpretation allows us to use existing word embeddings as an unsupervised model of lexical entailment, successfully predicting hyponymy relations using the proposed entailment operators in both unsupervised and semi-supervised experiments.

2 Modelling Entailment in a Vector Space

To develop a model of entailment in a vector space, we start with the logical definition of entailment in terms of vectors of discrete known features: y entails x if and only if all the known features in x are also included in y. We formalise this relation with binary vectors x, y where 1 means known and 0 means unknown, so this discrete entailment relation $(y \Rightarrow x)$ can be defined with the binary formula:

$$P((y \Rightarrow x) \mid x, y) = \prod_k (1 - (1 - y_k)x_k)$$

Given prior probability distributions $P(x), P(y)$ over these vectors, the exact joint and marginal probabilities for an entailment relation are:

$$P(x, y, (y \Rightarrow x)) = P(x)\ P(y) \prod_k (1 - (1 - y_k)x_k)$$

$$P((y \Rightarrow x)) = E_{P(x)} E_{P(y)} \prod_k (1 - (1 - y_k)x_k) \quad (1)$$

We cannot assume that the priors $P(x)$ and $P(y)$ are factorised, because there are many important correlations between features and therefore we cannot assume that the features are independent. As discussed in Section 1, even just representing both a feature f and its negation $\neg f$ requires two different dimensions k and k' in the vector space, because 0 represents unknown and not false. Given valid feature vectors, calculating entailment can consider these two dimensions separately, but to reason with distributions over vectors we need the prior $P(x)$ to enforce the constraint that x_k and $x_{k'}$ are mutually exclusive. In general, such correlations and anti-correlations exist between many semantic features, which makes inference and calculating the probability of entailment intractable.

To allow for efficient inference in such a model, we propose a mean-field approximation. This in effect assumes that the posterior distribution over vectors is factorised, but in practice this is a much weaker assumption than assuming the prior is factorised. The posterior distribution has less uncertainty and therefore is influenced less by non-factorised prior constraints. By assuming a factorised posterior, we can then represent distributions over feature vectors with simple vectors of

probabilities of individual features (or as below, with their log-odds). These real-valued vectors are the basis of the proposed vector-space model of entailment.

In the next two subsections, we derive a mean-field approximation for inference of real-valued vectors in entailment graphs. This derivation leads to three proposed vector-space operators for approximating the log-probability of entailment, summarised in Table 2. These operators will be used in the evaluation in Section 5. This inference framework will also be used in Section 3 to model how existing word embeddings can be mapped to vectors to which the entailment operators can be applied.

2.1 A Mean-Field Approximation

A mean-field approximation approximates the posterior P using a factorised distribution Q. First of all, this gives us a concise description of the posterior $P(x|\ldots)$ as a vector of continuous values $Q(x{=}1)$, where $Q(x{=}1)_k = Q(x_k{=}1) \approx E_{P(x|\ldots)}x_k = P(x_k{=}1|\ldots)$ (i.e. the marginal probabilities of each bit). Secondly, as is shown below, this gives us efficient methods for doing approximate inference of vectors in a model.

First we consider the simple case where we want to approximate the posterior distribution $P(x, y|y{\Rightarrow}x)$. In a mean-field approximation, we want to find a factorised distribution $Q(x, y)$ which minimises the KL-divergence $D_{KL}(Q(x, y)||P(x, y|y{\Rightarrow}x))$ with the true distribution $P(x, y|y{\Rightarrow}x)$.

$$\begin{aligned}
L &= D_{KL}(Q(x,y)||P(x,y|(y{\Rightarrow}x))) \\
&\propto \sum_x Q(x) \log \frac{Q(x,y)}{P(x,y,(y{\Rightarrow}x))} \\
&= \sum_k E_{Q(x_k)} \log Q(x_k) + \sum_k E_{Q(y_k)} \log Q(y_k) \\
&\quad - E_{Q(x)} \log P(x) - E_{Q(y)} \log P(y) \\
&\quad - \sum_k E_{Q(x_k)} E_{Q(y_k)} \log(1{-}(1{-}y_k)x_k)
\end{aligned}$$

In the final equation, the first two terms are the negative entropy of Q, $-H(Q)$, which acts as a maximum entropy regulariser, the final term enforces the entailment constraint, and the middle two terms represent the prior for x and y. One approach (generalised further in the next subsection) to the prior terms $-E_{Q(x)} \log P(x)$ is to bound them by assuming $P(x)$ is a function in the exponential family, giving us:

$$\begin{aligned}
E_{Q(x)} \log P(x) &\geq E_{Q(x)} \log \frac{\exp(\sum_k \theta^x_k x_k)}{\mathcal{Z}_\theta} \\
&= \sum_k E_{Q(x_k)} \theta^x_k x_k - \log \mathcal{Z}_\theta
\end{aligned}$$

where the $\log \mathcal{Z}_\theta$ is not relevant in any of our inference problems and thus will be dropped below.

As typically in mean-field approximations, inference of $Q(x)$ and $Q(y)$ can't be done efficiently with this exact objective L, because of the non-linear interdependence between x_k and y_k in the last term. Thus, we introduce two approximations to L, one for use in inferring $Q(x)$ given $Q(y)$ (forward inference), and one for the reverse inference problem (backward inference). In both cases, the approximation is done with an application of Jensen's inequality to the log function, which gives us an upper bound on L, as is standard practice in mean-field approximations. For forward inference:

$$\begin{aligned}
L \leq & -H(Q) - Q(x_k{=}1)\theta^x_k - E_{Q(y_k)}\theta^y_k y_k \\
& - Q(x_k{=}1) \log Q(y_k{=}1)\ )
\end{aligned} \quad (2)$$

which we can optimise for $Q(x_k{=}1)$:

$$Q(x_k{=}1) = \sigma(\ \theta^x_k + \log Q(y_k{=}1)\ ) \quad (3)$$

where $\sigma()$ is the sigmoid function. The sigmoid function arises from the entropy regulariser, making this a specific form of maximum entropy model. And for backward inference:

$$\begin{aligned}
L \leq & -H(Q) - E_{Q(x_k)}\theta^x_k x_k - Q(y_k{=}1)\theta^y_k \\
& - (1{-}Q(y_k{=}1)) \log(1{-}Q(x_k{=}1))\ )
\end{aligned} \quad (4)$$

which we can optimise for $Q(y_k{=}1)$:

$$Q(y_k{=}1) = \sigma(\ \theta^y_k - \log(1{-}Q(x_k{=}1))\ ) \quad (5)$$

Note that in equations (2) and (4) the final terms, $Q(x_k{=}1) \log Q(y_k{=}1)$ and $(1{-}Q(y_k{=}1)) \log(1{-}Q(x_k{=}1))$ respectively, are approximations to the log-probability of the entailment. We define two vector-space operators, $\circledcirc$ and $\oslash$, to be these same approximations.

$$\begin{aligned}
&\log Q(y{\Rightarrow}x) \\
&\quad \approx \sum_k E_{Q(x_k)} \log(E_{Q(y_k)}(1 - (1{-}y_k)x_k)) \\
&\quad = Q(x{=}1) \cdot \log Q(y{=}1) \ \equiv X \circledcirc Y
\end{aligned}$$

$$\begin{aligned}
&\log Q(y{\Rightarrow}x) \\
&\quad \approx \sum_k E_{Q(y_k)} \log(E_{Q(x_k)}(1 - (1{-}y_k)x_k)) \\
&\quad = (1{-}Q(y{=}1)) \cdot \log(1{-}Q(x{=}1)) \ \equiv Y \oslash X
\end{aligned}$$

$$
\begin{aligned}
X \circledcirc Y &\equiv \sigma(X) \cdot \log \sigma(Y) \\
Y \circledS X &\equiv \sigma(-Y) \cdot \log \sigma(-X) \\
Y \barwedge X &\equiv \sum_k \log(1 - \sigma(-Y_k)\sigma(X_k))
\end{aligned}
$$

Table 2: The proposed entailment operators, approximating $\log P(y \Rightarrow x)$.

We parametrise these operators with the vectors X, Y of log-odds of $Q(x), Q(y)$, namely $X = \log \frac{Q(x=1)}{Q(x=0)} = \sigma^{-1}(Q(x=1))$. The resulting operator definitions are summarised in Table 2.

Also note that the probability of entailment given in equation (1) becomes factorised when we replace P with Q. We define a third vector-space operator, $\barwedge$, to be this factorised approximation, also shown in Table 2.

2.2 Inference in Entailment Graphs

In general, doing inference for one entailment is not enough; we want to do inference in a graph of entailments between variables. In this section we generalise the above mean-field approximation to entailment graphs.

To represent information about variables that comes from outside the entailment graph, we assume we are given a prior $P(x)$ over all variables x_i in the graph. As above, we do not assume that this prior is factorised. Instead we assume that the prior $P(x)$ is itself a graphical model which can be approximated with a mean-field approximation.

Given a set of variables x_i each representing vectors of binary variables x_{ik}, a set of entailment relations $r = \{(i,j)|(x_i \Rightarrow x_j)\}$, and a set of negated entailment relations $\bar{r} = \{(i,j)|(x_i \not\Rightarrow x_j)\}$, we can write the joint posterior probability as:

$$
\begin{aligned}
P(x, r, \bar{r}) = \frac{1}{\mathcal{Z}} P(x) \prod_i \Big(& \\
& \Big(\prod_{j:r(i,j)} \prod_k P(x_{ik} \Rightarrow x_{jk} | x_{ik}, x_{jk}) \Big) \\
& \Big(\prod_{j:\bar{r}(i,j)} (1 - \prod_k P(x_{ik} \Rightarrow x_{jk} | x_{ik}, x_{jk})) \Big) \Big)
\end{aligned}
$$

We want to find a factorised distribution Q that minimises $L = D_{KL}(Q(x)||P(x|r,\bar{r}))$. As above, we bound this loss for each element $X_{ik} = \sigma^{-1}(Q(x_{ik}=1))$ of each vector we want to infer, using analogous Jensen's inequalities for the terms involving nodes i and j such that $r(i,j)$ or $r(j,i)$. For completeness, we also propose similar inequalities for nodes i and j such that $\bar{r}(i,j)$ or $\bar{r}(j,i)$, and bound them using the constants

$$C_{ijk} \geq \prod_{k' \neq k} (1 - \sigma(-X_{ik'})\sigma(X_{jk'})).$$

To represent the prior $P(x)$, we use the terms

$$\theta_{ik}(X_{\bar{ik}}) \leq \log \frac{E_{Q(x_{\bar{ik}})} P(x_{\bar{ik}}, x_{ik}=1)}{1 - E_{Q(x_{\bar{ik}})} P(x_{\bar{ik}}, x_{ik}=1)}$$

where $x_{\bar{ik}}$ is the set of all $x_{i'k'}$ such that either $i' \neq i$ or $k' \neq k$. These terms can be thought of as the log-odds terms that would be contributed to the loss function by including the prior's graphical model in the mean-field approximation.

Now we can infer the optimal X_{ik} as:

$$
\begin{aligned}
X_{ik} \;=\; & \theta_{ik}(X_{\bar{ik}}) + \sum_{j:r(i,j)} -\log \sigma(-X_{jk}) \qquad (6) \\
& + \sum_{j:r(j,i)} \log \sigma(X_{jk}) + \sum_{j:\bar{r}(j,i)} \log \frac{1 - C_{ijk}\sigma(X_{jk})}{1 - C_{ijk}} \\
& + \sum_{j:\bar{r}(i,j)} -\log \frac{1 - C_{ijk}\sigma(-X_{jk})}{1 - C_{ijk}}
\end{aligned}
$$

In summary, the proposed mean-field approximation does inference in entailment graphs by iteratively re-estimating each X_i as the sum of: the prior log-odds, $-\log \sigma(-X_j)$ for each entailed variable j, and $\log \sigma(X_j)$ for each entailing variable j.[1] This inference optimises $X_i \circledcirc X_j$ for each entailing j plus $X_i \circledS X_j$ for each entailed j, plus a maximum entropy regulariser on X_i. Negative entailment relations, if they exist, can also be incorporated with some additional approximations. Complex priors can also be incorporated through their log-odds, simulating the inclusion of the prior within the mean-field approximation.

Given its dependence on mean-field approximations, it is an empirical question to what extent we should view this model as computing real entailment probabilities and to what extent we should view it as a well-motivated non-linear mapping for which we simply optimise the input-output behaviour (as for neural networks (Henderson and Titov, 2010)). In Sections 3 and 5 we argue for the former (stronger) view.

3 Interpreting Word2Vec Vectors

To evaluate how well the proposed framework provides a formal foundation for the distributional semantics of entailment, we use it to re-interpret an

[1] It is interesting to note that $-\log \sigma(-X_j)$ is a non-negative transform of X_j, similar to the ReLU nonlinearity which is popular in deep neural networks (Glorot et al., 2011). $\log \sigma(X_j)$ is the analogous non-positive transform.

existing model of distributional semantics in terms of semantic entailment. There has been a lot of work on how to use the distribution of contexts in which a word occurs to induce a vector representation of the semantics of words. In this paper, we leverage this previous work on distributional semantics by re-interpreting a previous distributional semantic model and using this understanding to map its vector-space word embeddings to vectors in the proposed framework. We then use the proposed operators to predict entailment between words using these vectors. In Section 5 below, we evaluate these predictions on the task of hyponymy detection. In this section we motivate three different ways to interpret the Word2Vec (Mikolov et al., 2013a; Mikolov et al., 2013b) distributional semantic model as an approximation to an entailment-based model of the semantic relationship between a word and its context.

Distributional semantics learns the semantics of words by looking at the distribution of contexts in which they occur. To model this relationship, we assume that the semantic features of a word are (statistically speaking) redundant with those of its context words, and consistent with those of its context words. We model these properties using a hidden vector which is the consistent unification of the features of the middle word and the context. In other words, there must exist a hidden vector which entails both of these vectors, and is consistent with prior constraints on vectors. We split this into two steps, inference of the hidden vector Y from the middle vector X_m, context vectors X_c and prior, and computing the log-probability (7) that this hidden vector entails the middle and context vectors:

$$\max_{Y}(\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c)) \tag{7}$$

We interpret Word2Vec's Skip-Gram model as learning its context and middle word vectors so that the log-probability of this entailment is high for the observed context words and low for other (sampled) context words. The word embeddings produced by Word2Vec are only related to the vectors X_m assigned to the middle words; context vectors are computed but not output. We model the context vectors X'_c as combining (as in equation (5)) information about a context word itself with information which can be inferred from this word given the prior, $X'_c = \theta_c - \log\sigma(-X_c)$.

The numbers in the vectors output by Word2Vec are real numbers between negative infinity and infinity, so the simplest interpretation of them is as the log-odds of a feature being known. In this case we can treat these vectors directly as the X_m in the model. The inferred hidden vector Y can then be calculated using the model of backward inference from the previous section.

$$\begin{aligned} Y &= \theta_c - \log\sigma(-X_c) - \log\sigma(-X_m) \\ &= X'_c - \log\sigma(-X_m) \end{aligned}$$

Since the unification Y of context and middle word features is computed using backward inference, we use the backward-inference operator $\circledcirc$ to calculate how successful that unification was. This gives us the final score:

$$\begin{aligned} &\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c) \\ &\quad\approx\ Y{\circledcirc}X_m + Y{\circledcirc}X_c + -\sigma(-Y){\cdot}\theta_c \\ &\quad=\ Y{\circledcirc}X_m + -\sigma(-Y){\cdot}X'_c \end{aligned}$$

This is a natural interpretation, but it ignores the equivalence in Word2Vec between pairs of positive values and pairs of negative values, due to its use of the dot product. As a more accurate interpretation, we interpret each Word2Vec dimension as specifying whether its feature is known to be true or known to be false. Translating this Word2Vec vector into a vector in our entailment vector space, we get one copy Y^+ of the vector representing known-to-be-true features and a second negated duplicate Y^- of the vector representing known-to-be-false features, which we concatenate to get our representation Y.

$$\begin{aligned} Y^+ &= X'_c - \log\sigma(-X_m) \\ Y^- &= -X'_c - \log\sigma(X_m) \\ &\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c) \\ &\quad\approx\ Y^+{\circledcirc}X_m + -\sigma(-Y^+){\cdot}X'_c \\ &\qquad + Y^-{\circledcirc}(-X_m) + -\sigma(-Y^-){\cdot}(-X'_c) \end{aligned}$$

As a third alternative, we modify this latter interpretation with some probability mass reserved for unknown in the vicinity of zero. By subtracting 1 from both the original and negated copies of each dimension, we get a probability of unknown of $1{-}\sigma(X_m{-}1) - \sigma(-X_m{-}1)$. This gives us:

$$\begin{aligned} Y^+ &= X'_c - \log\sigma(-(X_m{-}1)) \\ Y^- &= -X'_c - \log\sigma(-(-X_m{-}1)) \\ &\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c) \\ &\quad\approx\ Y^+{\circledcirc}(X_m{-}1) + -\sigma(-Y^+){\cdot}X'_c \\ &\qquad + Y^-{\circledcirc}(-X_m{-}1)) + -\sigma(-Y^-){\cdot}(-X'_c) \end{aligned}$$

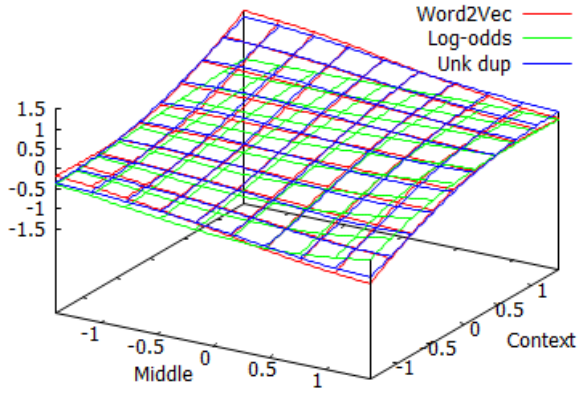

Figure 1: The learning gradients for Word2Vec, the *log-odds* ⊘, and the *unk dup* ⊘ interpretation of its vectors.

To understand better the relative accuracy of these three interpretations, we compared the training gradient which Word2Vec uses to train its middle-word vectors to the training gradient for each of these interpretations. We plotted these gradients for the range of values typically found in Word2Vec vectors for both the middle vector and the context vector. Figure 1 shows three of these plots. As expected, the second interpretation is more accurate than the first because its plot is anti-symmetric around the diagonal, like the Word2Vec gradient. In the third alternative, the constant 1 was chosen to optimise this match, producing a close match to the Word2Vec training gradient, as shown in Figure 1 (*Word2Vec* versus *Unk dup*).

Thus, Word2Vec can be seen as a good approximation to the third model, and a progressively worse approximation to the second and first models. Therefore, if the entailment-based distributional semantic model we propose is accurate, then we would expect the best accuracy in hyponymy detection using the third interpretation of Word2Vec vectors, and progressively worse accuracy for the other two interpretations. As we will see in Section 5, this prediction holds.

4 Related Work

There has been a significant amount of work on using distributional-semantic vectors for hyponymy detection, using supervised, semi-supervised or unsupervised methods (e.g. (Yu et al., 2015; Necsulescu et al., 2015; Vylomova et al., 2015; Weeds et al., 2014; Fu et al., 2015; Rei and Briscoe, 2014)). Because our main concern is modelling entailment within a vector space, we do not do a thorough comparison to models which use measures computed outside the vector space (e.g. symmetric measures (LIN (Lin, 1998)), asymmetric measures (WeedsPrec (Weeds and Weir, 2003; Weeds et al., 2004), balAPinc (Kotlerman et al., 2010), invCL (Lenci and Benotto, 2012)) and entropy-based measures (SLQS (Santus et al., 2014))), nor to models which encode hyponymy in the parameters of a vector-space operator or classifier (Fu et al., 2015; Roller et al., 2014; Baroni et al., 2012)). We also limit our evaluation of lexical entailment to hyponymy, not including other related lexical relations (cf. (Weeds et al., 2014; Vylomova et al., 2015; Turney and Mohammad, 2014; Levy et al., 2014)), leaving more complex cases to future work on compositional semantics. We are also not concerned with models or evaluations which require supervised learning about individual words, instead limiting ourselves to semi-supervised learning where the words in the training and test sets are disjoint.

For these reasons, in our evaluations we replicate the experimental setup of Weeds et al. (2014), for both unsupervised and semi-supervised models. Within this setup, we compare to the results of the models evaluated by Weeds et al. (2014) and to previously proposed vector-space operators. This includes one vector space operator for hyponymy which doesn't have trained parameters, proposed by Rei and Briscoe (2014), called *weighted cosine*. The dimensions of the dot product (normalised to make it a cosine measure) are weighted to put more weight on the larger values in the entailed (hypernym) vector.

We base this evaluation on the Word2Vec (Mikolov et al., 2013a; Mikolov et al., 2013b) distributional semantic model and its publicly available word embeddings. We choose it because it is popular, simple, fast, and its embeddings have been derived from a very large corpus. Levy and Goldberg (2014) showed that it is closely related to the previous PMI-based distributional semantic models (e.g. (Turney and Pantel, 2010)).

The most similar previous work, in terms of motivation and aims, is that of Vilnis and McCallum (2015). They also model entailment directly using a vector space, without training a classifier. But instead of representing words as a point in a vector space (as in this work), they represent words as a Gaussian distribution over points in a vector space. This allows them to represent the extent to which a feature is known versus unknown as the

amount of variance in the distribution for that feature's dimension. While nicely motivated theoretically, the model appears to be more computationally expensive than the one proposed here, particularly for inferring vectors. They do make unsupervised predictions of hyponymy relations with their learned vector distributions, using KL-divergence between the distributions for the two words. They evaluate their models on the hyponymy data from (Baroni et al., 2012). As discussed further in section 5.2, our best models achieve non-significantly better average precision than their best models.

The semi-supervised model of Kruszewski et al. (2015) also models entailment in a vector space, but they use a discrete vector space. They train a mapping from distributional semantic vectors to Boolean vectors such that feature inclusion respects a training set of entailment relations. They then use feature inclusion to predict hyponymy, and other lexical entailment relations. This approach is similar to the one used in our semi-supervised experiments, except that their discrete entailment prediction operator is very different from our proposed entailment operators.

5 Evaluation

To evaluate whether the proposed framework is an effective model of entailment in vector spaces, we apply the interpretations from Section 3 to publicly available word embeddings and use them to predict the hyponymy relations in a benchmark dataset. This framework predicts that the more accurate interpretations of Word2Vec result in more accurate unsupervised models of hyponymy. We evaluate on detecting hyponymy relations between words because hyponymy is the canonical type of lexical entailment; most of the semantic features of a hypernym (e.g. "animal") must be included in the semantic features of the hyponym (e.g. "dog"). We evaluate in both a fully unsupervised setup and a semi-supervised setup.

5.1 Hyponymy with Word2Vec Vectors

For our evaluation on hyponymy detection, we replicate the experimental setup of Weeds et al. (2014), using their selection of word pairs[2] from the BLESS dataset (Baroni and Lenci, 2011).[3] These noun-noun word pairs include positive hyponymy pairs, plus negative pairs consisting of some other hyponymy pairs reversed, some pairs in other semantic relations, and some random pairs. Their selection is balanced between positive and negative examples, so that accuracy can be used as the performance measure. For their semi-supervised experiments, ten-fold cross validation is used, where for each test set, items are removed from the associated training set if they contain any word from the test set. Thus, the vocabulary of the training and testing sets are always disjoint, thereby requiring that the models learn about the vector space and not about the words themselves. We had to perform our own 10-fold split, but apply the same procedure to filter the training set.

We could not replicate the word embeddings used in Weeds et al. (2014), so instead we use publicly available word embeddings.[4] These vectors were trained with the Word2Vec software applied to about 100 billion words of the Google-News dataset, and have 300 dimensions.

The hyponymy detection results are given in Table 3, including both unsupervised (upper box) and semi-supervised (lower box) experiments. We report two measures of performance, hyponymy detection accuracy (*50% Acc*) and direction classification accuracy (*Dir Acc*). Since all the operators only determine a score, we need to choose a threshold to get detection accuracies. Given that the proportion of positive examples in the dataset has been artificially set at 50%, we threshold each model's score at the point where the proportion of positive examples output is 50%, which we call "*50% Acc*". Thus the threshold is set after seeing the testing inputs but not their target labels.

Direction classification accuracy (*Dir Acc*) indicates how well the method distinguishes the relative abstractness of two nouns. Given a pair of nouns which are in a hyponymy relation, it classifies which word is the hypernym and which is the hyponym. This measure only considers positive examples and chooses one of two directions, so it is inherently a balanced binary classification task. Classification is performed by simply comparing the scores in both directions. If both directions produce the same score, the expected random accuracy (50%) is used.

As representative of previous work, we report

[2] https://github.com/SussexCompSem/learninghypernyms

[3] Of the 1667 word pairs in this data, 24 were removed because we do not have an embedding for one of the words.

[4] https://code.google.com/archive/p/word2vec/

operator	supervision	50% Acc	Dir Acc
Weeds et.al.	None	58%	–
log-odds $\oslash$	None	54.0%	55.9%
weighted cos	None	55.5%	57.9%
dot	None	56.3%	50%
dif	None	56.9%	59.6%
log-odds $\vec{\Rightarrow}$	None	57.0%	59.4%
log-odds $\oslash$	None	60.1%*	62.2%
dup $\oslash$	None	61.7%	68.8%
unk dup $\vec{\Rightarrow}$	None	63.4%*	68.8%
unk dup $\oslash$	None	64.5%	68.8%
Weeds et.al.	SVM	75%	–
mapped dif	cross ent	64.3%	72.3%
mapped $\oslash$	cross ent	74.5%	91.0%
mapped $\vec{\Rightarrow}$	cross ent	77.5%	92.3%
mapped $\oslash$	cross ent	80.1%	90.0%

Table 3: Accuracies on the BLESS data from Weeds et al. (2014), for hyponymy detection (*50% Acc*) and hyponymy direction classification (*Dir Acc*), in the unsupervised (upper box) and semi-supervised (lower box) experiments. For unsupervised accuracies, * marks a significant difference with the previous row.

the best results from Weeds et al. (2014), who try a number of unsupervised and semi-supervised models, and use the same testing methodology and hyponymy data. However, note that their word embeddings are different. For the semi-supervised models, Weeds et al. (2014) trains classifiers, which are potentially more powerful than our linear vector mappings. We also compare the proposed operators to the dot product (*dot*),[5] vector differences (*dif*), and the weighted cosine of Rei and Briscoe (2014) (*weighted cos*), all computed with the same word embeddings as for the proposed operators.

In Section 3 we argued for three progressively more accurate interpretations of Word2Vec vectors in the proposed framework, the log-odds interpretation (*log-odds* $\oslash$), the negated duplicate interpretation (*dup* $\oslash$), and the negated duplicate interpretation with unknown around zero (*unk dup* $\oslash$). We also evaluate using the factorised calculation of entailment (*log-odds* $\vec{\Rightarrow}$, *unk dup* $\vec{\Rightarrow}$), and the backward-inference entailment operator (*log-odds* $\oslash$), neither of which match the proposed interpretations. For the semi-supervised case, we train a linear vector-space mapping into a new vector space, in which we apply the operators (*mapped* operators). All these results are discussed in the next two subsections.

[5]We also tested the cosine measure, but results were very slightly worse than *dot*.

5.2 Unsupervised Hyponymy Detection

The first set of experiments evaluate the vector-space operators in unsupervised models of hyponymy detection. The proposed models are compared to the dot product, because this is the standard vector-space operator and has been shown to capture semantic similarity very well. However, because the dot product is a symmetric operator, it always performs at chance for direction classification. Another vector-space operator which has received much attention recently is vector differences. This is used (with vector sum) to perform semantic transforms, such as "king - male + female = queen", and has previously been used for modelling hyponymy (Vylomova et al., 2015; Weeds et al., 2014). For our purposes, we sum the pairwise differences to get a score which we use for hyponymy detection.

For the unsupervised results in the upper box of table 3, the best unsupervised model of Weeds et al. (2014), and the operators *dot*, *dif* and *weighted cos* all perform similarly on accuracy, as does the log-odds factorised entailment calculation (*log-odds* $\vec{\Rightarrow}$). The forward-inference entailment operator (*log-odds* $\oslash$) performs above chance but not well, as expected given the backward-inference-based interpretation of Word2Vec vectors. By definition, *dot* is at chance for direction classification, but the other models all perform better, indicating that all these operators are able to measure relative abstractness. As predicted, the $\oslash$ operator performs significantly better than all these results on accuracy, as well as on direction classification, even assuming the log-odds interpretation of Word2Vec vectors.

When we move to the more accurate interpretation of Word2Vec vectors as specifying both original and negated features (*dup* $\oslash$), we improve (non-significantly) on the log-odds interpretation. Finally, the third and most accurate interpretation, where values around zero can be unknown (*unk dup* $\oslash$), achieves the best results in unsupervised hyponymy detection, as well as for direction classification. Changing to the factorised entailment operator (*unk dup* $\vec{\Rightarrow}$) is worse but also signifi-

cantly better than the other accuracies.

To allow a direct comparison to the model of Vilnis and McCallum (2015), we also evaluated the unsupervised models on the hyponymy data from (Baroni et al., 2012). Our best model achieved 81% average precision on this dataset, non-significantly better than the 80% achieved by the best model of Vilnis and McCallum (2015).

5.3 Semi-supervised Hyponymy Detection

Since the unsupervised learning of word embeddings may reflect many context-word correlations which have nothing to do with hyponymy, we also consider a semi-supervised setting. Adding some supervision helps distinguish features that capture semantic properties from other features which are not relevant to hyponymy detection. But even with supervision, we still want the resulting model to be captured in a vector space, and not in a parametrised scoring function. Thus, we train mappings from the Word2Vec word vectors to new word vectors, and then apply the entailment operators in this new vector space to predict hyponymy. Because the words in the testing set are always disjoint from the words in the training set, this experiment measures how well the original unsupervised vector space captures features that generalise entailment across words, and not how well the mapping can learn about individual words.

Our objective is to learn a mapping to a new vector space in which an operator can be applied to predict hyponymy. We train linear mappings for the $\oslash$ operator (*mapped* $\oslash$) and for vector differences (*mapped dif*), since these were the best performing proposed operator and baseline operator, respectively, in the unsupervised experiments. We do not use the duplicated interpretations because these transforms are subsumed by the ability to learn a linear mapping.[6] Previous work on using vector differences for semi-supervised hyponymy detection has used a linear SVM (Vylomova et al., 2015; Weeds et al., 2014), which is mathematically equivalent to our vector-differences model, except that we use cross entropy loss and they use a large-margin loss and SVM training.

The semi-supervised results in the bottom box of table 3 show a similar pattern to the unsupervised results.[7] The $\oslash$ operator achieves the best generalisation from training word vectors to testing word vectors. The *mapped* $\oslash$ model has the best accuracy, followed by the factorised entailment operator *mapped* $\vec{\Rightarrow}$ and Weeds et al. (2014). Direction accuracies of all the proposed operators (*mapped* $\oslash$, *mapped* $\vec{\Rightarrow}$, *mapped* $\oslash$) reach into the 90's. The *dif* operator performs particularly poorly in this *mapped* setting, perhaps because both the mapping and the operator are linear. These semi-supervised results again support our distributional-semantic interpretations of Word2Vec vectors and their associated entailment operator $\oslash$.

6 Conclusion

In this work, we propose a vector-space model which provides a formal foundation for a distributional semantics of entailment. We developed a mean-field approximation to probabilistic entailment between vectors which represent known versus unknown features. And we used this framework to derive vector operators for entailment and vector inference equations for entailment graphs. This framework allows us to reinterpret Word2Vec as approximating an entailment-based distributional semantic model of words in context, and show that more accurate interpretations result in more accurate unsupervised models of lexical entailment, achieving better accuracies than previous models. Semi-supervised evaluations confirm these results.

A crucial distinction between the semi-supervised models here and much previous work is that they learn a mapping into a vector space which represents entailment, rather than learning a parametrised entailment classifier. Within this new vector space, the entailment operators and inference equations apply, thereby generalising naturally from these lexical representations to the compositional semantics of multi-word expressions and sentences. Further work is needed to explore the full power of these abilities to extract information about entailment from both unlabelled text and labelled entailment data, encode it all in a single vector space, and efficiently perform complex inferences about vectors and entailments. This future work on compositional distributional semantics should further demonstrate the full power of the proposed framework for modelling entailment in a vector space.

[6] Empirical results confirm that this is in practice the case, so we do not include these results in the table.

[7] It is not clear how to measure significance for cross-validation results, so we do not attempt to do so.

References

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, GEMS '11, pages 1–10. Association for Computational Linguistics.

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 (EACL)*, pages 23–32, Avignon, France. Association for Computational Linguistics.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2015. Learning semantic hierarchies: A continuous vector space approach. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(3):461–471.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323.

James Henderson and Ivan Titov. 2010. Incremental sigmoid belief networks for grammar learning. *Journal of Machine Learning Research*, 11(Dec):3541–3570.

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

Germn Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, SemEval '12, pages 75–79. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for open ie propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 87–97, Ann Arbor, Michigan. Association for Computational Linguistics.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado, May–June. Association for Computational Linguistics.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 768–774. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 182–192, Denver, Colorado. Association for Computational Linguistics.

Marek Rei and Ted Briscoe. 2014. Looking for hyponyms in vector space. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 68–77, Ann Arbor, Michigan. Association for Computational Linguistics.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1025–1036, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 38–42.

Hinrich Schütze. 1993. Word space. In *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan Kaufmann.

Peter D. Turney and Saif M. Mohammad. 2014. Experiments with three approaches to recognizing lexical entailment. *CoRR*, abs/1401.8269.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188.

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

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2015. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *CoRR 2015*.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 81–88.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, pages 1015–1021. Association for Computational Linguistics.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI 2015. AAAI Press / International Joint Conferences on Artificial Intelligence.