Modeling Context Words as Regions: An Ordinal Regression Approach to Word Embedding

Shoaib Jameel and Steven Schockaert
School of Computer Science and Informatics
Cardiff University
{JameelS1, SchockaertS1}@cardiff.ac.uk

Abstract

Vector representations of word meaning have found many applications in the field of natural language processing. Word vectors intuitively represent the average context in which a given word tends to occur, but they cannot explicitly model the diversity of these contexts. Although region representations of word meaning offer a natural alternative to word vectors, only few methods have been proposed that can effectively learn word regions. In this paper, we propose a new word embedding model which is based on SVM regression. We show that the underlying ranking interpretation of word contexts is sufficient to match, and sometimes outperform, the performance of popular methods such as Skip-gram. Furthermore, we show that by using a quadratic kernel, we can effectively learn word regions, which outperform existing unsupervised models for the task of hypernym detection.

1 Introduction

Word embedding models such as Skip-gram (Mikolov et al., 2013b) and GloVe (Pennington et al., 2014) represent words as vectors of typically around 300 dimensions. The relatively low-dimensional nature of these word vectors makes them ideally suited for representing textual input to neural network models (Goldberg, 2016; Nayak, 2015). Moreover, word embeddings have been found to capture many interesting regularities (Mikolov et al., 2013b; Kim and de Marneffe, 2013; Gupta et al., 2015; Rothe and Schütze, 2016), which makes it possible to use them as a source of semantic and linguistic knowledge, and to align word embeddings with visual features (Frome et al., 2013) or across different languages (Zou et al., 2013; Faruqui and Dyer, 2014).

Notwithstanding the practical advantages of representing words as vectors, a few authors have advocated the idea that words may be better represented as regions (Erk, 2009), possibly with gradual boundaries (Vilnis and McCallum, 2015). One important advantage of region representations is that they can distinguish words with a broad meaning from those with a more narrow meaning, and should thus in principle be better suited for tasks such as hypernym detection and taxonomy learning. However, it is currently not well understood how such region based representations can best be learned. One possible approach, suggested in (Vilnis and McCallum, 2015), is to learn a multivariate Gaussian for each word, essentially by requiring that words which frequently occur together are represented by similar Gaussians. However, for large vocabularies, this is computationally only feasible with diagonal covariance matrices.

In this paper, we propose a different approach to learning region representations for words, which is inspired by a geometric view of the Skip-gram model. Essentially, Skip-gram learns two vectors p_w and $\tilde{p_w}$ for each word w, such that the probability that a word c appears in the context of a target word t can be expressed as a function of $p_t \cdot \tilde{p}_c$ (see Section 2). This means that for each threshold $\lambda \in [-1, 1]$ and context word c, there is a hyperplane H^c_λ which (approximately) separates the words t for which $p_t \cdot \tilde{p}_c \geq \lambda$ from the others. Note that this hyperplane is completely determined by the vector $\tilde{p}_c$ and the choice of λ. An illustration of this geometric view is shown in Figure 1(a), where e.g. the word c is strongly related to a (i.e. a has a high probability of occurring in the context of c) but not closely related to b. Note in particular that there is a half-space containing those words which are strongly related to a (w.r.t.

Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pages 123–133,
Vancouver, Canada, August 3 - August 4, 2017.

a given threshold λ).

Our contribution is twofold. First, we empirically show that effective word embeddings can be learned from purely ordinal information, which stands in contrast to the probabilistic view taken by e.g. Skip-gram and GloVe. Specifically, we propose a new word embedding model which uses (a ranking equivalent of) max-margin constraints to impose the requirement that $p_t \cdot \tilde{p}_c$ should be a monotonic function of the probability $P(c|t)$ of seeing c in the context of t. Geometrically, this means that, like Skip-gram, our model associates with each context word a number of parallel hyperplanes. However, unlike in the Skip-gram model, only the relative position of these hyperplanes is imposed (i.e. if $\lambda_1 < \lambda_2 < \lambda_3$ then $H_c^{\lambda_2}$ should occur between $H_c^{\lambda_1}$ and $H_c^{\lambda_3}$). Second, by using a quadratic kernel for the max-margin constraints, we obtain a model that can represent context words as a set of nested ellipsoids, as illustrated in Figure 1(b). From these nested ellipsoids we can then estimate a Gaussian which acts as a convenient region based word representation.

Note that our model thus jointly learns a vector representation for each word (i.e. the target word representations) as well as a region based representation (i.e. the nested ellipsoids representing the context words). We present experimental results which show that the region based representations are effective for measuring synonymy and hypernymy. Moreover, perhaps surprisingly, the region based modeling of context words also benefits the target word vectors, which match, and in some cases outperform the vectors obtained by standard word embedding models on various benchmark evaluation tasks.

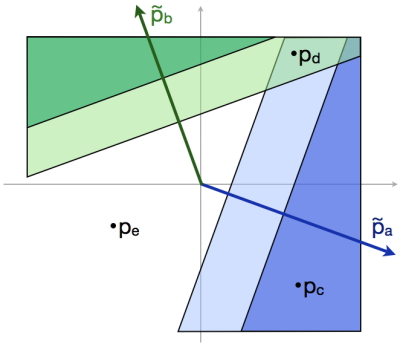
2 Background and Related Work

2.1 Word Embedding

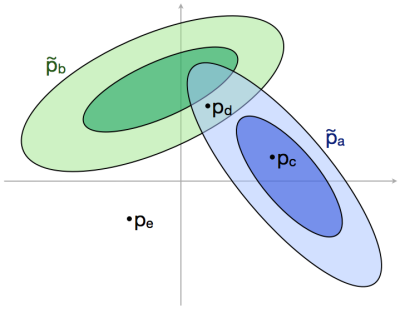
Various methods have already been proposed for learning vector space representations of words, e.g. based on matrix factorization (Turney and Pantel, 2010) or neural networks. Here we briefly review Skip-gram and GloVe, two popular models which share some similarities with our model.

The basic assumption of Skip-gram (Mikolov et al., 2013b) is that the probability $P(c|t)$ of seeing word c in the context of word t is given as:

$$P(c|t) = \frac{p_t \cdot \tilde{p}_c}{\sum_{c'} p_t \cdot \tilde{p}_{c'}}$$



(a) Linear kernel



(b) Quadratic kernel

Figure 1: The (dark) green region covers words that are (strongly) related to a. Similarly, the (dark) blue region expresses relatedness to b.

In principle, based on this view, the target vectors p_w and context vectors $\tilde{p}_w$ could be learned by maximizing the likelihood of a given corpus. Since this is computationally not feasible, however, it was proposed in (Mikolov et al., 2013b) to instead optimize the following objective:

$$\sum_{i=1}^{N} \sum_{c' \in C_i} \log(\sigma(p_{w_i} \cdot \tilde{p}_c)) + \sum_{c' \in \overline{C}_i} \log(-\sigma(p_{w_i} \cdot \tilde{p}_{c'}))$$

where the left-most summation is over all N word occurrences in the corpus, w_i is the i^{th} word in the corpus, C_i are the words appearing in the context of w_i and $\overline{C}_i$ consists of $k \cdot |C_i|$ randomly chosen words, called the negative samples for w_i. The context C_i contains the t_i words immediately preceding and succeeding w_i, where t_i is randomly sampled from $\{1, ..., t_{max}\}$ for each i (Goldberg and Levy, 2014). The probability of choosing word w as a negative sample is proportional to $\left(\frac{occ(w)}{N}\right)^{0.75}$, with $occ(w)$ the number of occurrences of word w in the corpus. Finally, to reduce the impact of frequent words, some word occurrences are removed from the corpus before applying the model, with the probability of removing an

occurrence of word w being $1 - \sqrt{\frac{\theta}{occ(w)}}$. Default parameter values are $t_{max} = 5$ and $\theta = 10^{-5}$.

GloVe is another popular model for word embedding (Pennington et al., 2014). Rather than explicitly considering all word occurrences, it directly uses a global co-occurrence matrix $X = (x_{ij})$ where x_{ij} is the number of times the word w_j appears in the context of w_i. Like Skip-gram, it learns both a target vector p_w and context vector $\tilde{p}_w$ for each word w, but instead learns these vectors by optimizing the following objective:

$$\sum_i \sum_j f(x_{ij})(p_{w_i} \cdot \tilde{p}_{w_j} + b_{w_i} + \tilde{b}_{w_j} - \log x_{ij})^2$$

where b_{w_i} and $\tilde{b}_{w_j}$ are bias terms, and f is a weighting function to reduce the impact of very rare terms, defined as:

$$f(x_{ij}) = \begin{cases} (\frac{x_{ij}}{x_{max}})^\alpha & \text{if } x_{ij} < x_{max} \\ 1 & \text{otherwise} \end{cases}$$

The default values are $x_{max} = 100$ and $\alpha = 0.75$.

2.2 Region Representations

The idea of representing words as regions was advocated in (Erk, 2009), as a way of modeling the diversity of the contexts in which a word appears. It was argued that such regions could be used to more accurately model the meaning of polysemous words and to model lexical entailment. Rather than learning region representations directly, it was proposed to use a vector space representation of word occurrences. Two alternatives were investigated for estimating a region from these occurrence vectors, respectively inspired by prototype and exemplar based models of categorization. The first approach defines the region as the set of points whose weighted distance to a prototype vector for the word is within a given radius, while the second approach relies on the k-nearest neighbor principle.

In contrast, (Vilnis and McCallum, 2015) proposed a method that directly learns a representation in which each word corresponds to a Gaussian. The model uses an objective function which requires the Gaussians of words that co-occur to be more similar than the Gaussians of words of negative samples (which are obtained as in the Skip-gram model). Two similarity measures are considered: the inner product of the Gaussians and the KL-divergence. It is furthermore argued that the asymmetric nature of KL-divergence makes it a natural choice for modeling hypernymy. In particular, it is proposed that the word embeddings could be improved by imposing that words that are in a hypernym relation have a low KL-divergence, allowing for a natural way to combine corpus statistics with available taxonomies.

Finally, another model that represents words using probability distributions was proposed in (Jameel and Schockaert, 2016). However, their model is aimed at capturing the uncertainty about vector representations, rather than at modeling the diversity of words. They show that capturing this uncertainty leads to vectors that outperform those of the GloVe model, on which their model is based. However, the resulting distributions are not suitable for modeling hypernymy. For example, since more information is available for general terms than for narrow terms, the distributions associated with general terms have a smaller variance, whereas approaches that are aimed at modeling the diversity of words have the opposite behavior.

2.3 Ranking Embedding

The model we propose only relies on the rankings induced by each context word, and tries to embed these rankings in a vector space. This problem of "ranking embedding" has already been studied by a few authors. An elegant approach for embedding a given set of rankings, based on the product order, is proposed in (Vendrov et al., 2016). However, this method is specifically aimed at completing partially ordered relations (such as taxonomies), based on observed statistical correlations, and would not be directly suitable as a basis for a word embedding method. The computational complexity of the ranking embedding problem was characterized in (Schockaert and Lee, 2015), where the associated decision problem was shown to be complete for the class $\exists\mathbb{R}$ (which sits between NP and PSPACE).

Note that the problem of ranking embedding is different from the learning-to-rank task (Liu, 2009). In the former case we are interested in learning a vector space representation that is somehow in accordance with a given completely specified set of rankings, whereas in the latter case the focus is on representing incompletely specified rankings in a given vector space representation.

3 Ordinal Regression Word Embedding

3.1 Learning the Embedding

In this section we explain how a form of ordinal regression can be used to learn both word vectors and word regions at the same time. First we introduce some notations.

Recall that the Positive Pointwise Mutual Information (PPMI) between two words w_i and w_j is defined as $\mathit{PPMI}(w_i, w_j) = \max(0, \mathit{PMI}(w_i, w_j))$, with $\mathit{PMI}(w_i, w_j)$ given by:

$$\log\left(\frac{n(w_i, w_j) \cdot (\sum_{w \in W} \sum_{w' \in W} n(w, w'))}{(\sum_{w \in W} n(w_i, w)) \cdot (\sum_{w \in W} n(w, w_j))}\right)$$

where we write $n(w_i, w_j)$ for the number of times word w_j occurs in the context of w_i, and W represents the vocabulary. For each word w_j, we write $W_0^j, ..., W_{n_j}^j$ for the stratification of the words in the vocabulary according to their PPMI value with w_j, i.e. we have that:

1. $\mathit{PPMI}(w, w_j) = 0$ for $w \in W_0^j$;
2. $\mathit{PPMI}(w, w_j) < \mathit{PPMI}(w', w_j)$ for $w \in W_i^j$ and $w' \in W_k^j$ with $i < k$; and
3. $\mathit{PPMI}(w, w_j) = \mathit{PPMI}(w', w_j)$ for $w, w' \in W_i^j$.

As a toy example, suppose $W = \{w_1, w_2, w_3, w_4, w_5\}$ and:

$$\begin{aligned} &\mathit{PPMI}(w_2, w_1) = 3.4 \quad \mathit{PPMI}(w_3, w_1) = 4.1 \\ &\mathit{PPMI}(w_4, w_1) = 0 \quad \mathit{PPMI}(w_5, w_1) = 0 \\ &\mathit{PPMI}(w_1, w_1) = 0 \end{aligned}$$

Then we would have $W_0^1 = \{w_1, w_4, w_5\}$, $W_1^1 = \{w_2\}$ and $W_2^1 = \{w_3\}$.

To learn the word embedding, we use the following objective function, which requires that for each context word w_j there is a sequence of parallel hyperplanes that separate the representations of the words in W_{i-1}^j from the representations of the words in W_i^j ($i \in \{1, ..., n_j\}$):

$$\sum_j \left(\sum_{i=1}^{n_j} \frac{\mathit{pos}(j, i-1) + \mathit{neg}(j, i)}{|W_{i-1}^j \cup W_i^j|}\right) + \lambda \|\tilde{p}_{w_j}\|^2$$

where

$$\mathit{pos}(j, i-1) = \sum_{w \in W_{i-1}^j} [1 - (\phi(p_w) \cdot \tilde{p}_{w_j} + b_j^i)]_+^2$$

$$\mathit{neg}(j, i) = \sum_{w \in W_i^j} [1 + (\phi(p_w) \cdot \tilde{p}_{w_j} + b_j^i)]_+^2$$

subject to[1] $b_j^1 < ... < b_j^{n_j}$ for each j. Note that we write $[x]_+$ for $\max(0, x)$ and ϕ denotes the feature map of the considered kernel function. In this paper, we will in particular consider linear and quadratic kernels. If a linear kernel is used, then ϕ is simply the identity function. Using a quadratic kernel leads to a quadratic increase in the dimensionality of $\phi(p_w)$ and $\tilde{p}_{w_j}$. In practice, we found our model to be about 3 times slower when a quadratic kernel is used, when the word vectors p_w are chosen to be 300-dimensional. Note that $\tilde{p}_{w_j}$ and b_j^i define a hyperplane, separating the kernel space into a positive and a negative half-space. The constraints of the form $\mathit{pos}(j, i-1)$ essentially encode that the elements from W_{i-1}^j should be represented in the positive half-space, whereas the constraints of the form $\mathit{neg}(j, i)$ encode that the elements from W_i^j should be represented in the negative half-space.

When using a linear kernel, the model is similar in spirit to Skip-gram, in the sense that it associates with each context word a sequence of parallel hyperplanes. In our case, however, only the ordering of these hyperplanes is specified, i.e. the specific offsets b_j^i are learned. In other words, we make the assumption that the higher $\mathit{PPMI}(w, w_j)$ the stronger w is related to w_j, but we do not otherwise assume that the numerical value of $\mathit{PPMI}(w, w_j)$ is relevant. When using a quadratic kernel, each context word is essentially modeled as a sequence of nested ellipsoids. This gives the model a lot more freedom to satisfy the constraints, which may potentially lead to more informative vectors.

The model is similar in spirit to the fixed margin variant for ranking with large-margin constraints proposed in (Shashua and Levin, 2002), but with the crucial difference that we are learning word vectors and hyperplanes at the same time, rather than finding hyperplanes for a given vector space representation. We use stochastic gradient descent to optimize the proposed objective. Note that we use a squared hinge loss, which makes optimizing the objective more straightforward. As usual, the parameter λ controls the trade-off between maintaining a wide margin and minimizing classifica-

[1] While it may seem at first glance that this constraint is redundant, this is not actually the case; see (Chu and Keerthi, 2005) for a counterexample in a closely related framework.

tion errors. Throughout the experiments we have kept λ at a default value of 0.5. We have also added L2 regularization for the word vectors w_t with a weight of 0.01, which was found to increase the stability of the model. In practice, W_0^j is typically very large (containing most of the vocabulary), which would make the model too inefficient. To address this issue, we replace it by a small subsample, which is similar in spirit to the idea of negative sampling in the Skip-gram model. In our experiments we use $2k$ randomly sampled words from W, where $k = \sum_{i=1}^{n_j} |W_i^j|$ is the total number of positive samples. We simply use a uniform distribution to obtain the negative samples, as initial experiments showed that using other sampling strategies had almost no effect on the result.

3.2 Using Region Representations

When using a quadratic kernel, the hyperplanes defined by the vector $\tilde{p}_{w_j}$ and offsets b_j^i define a sequence of nested ellipsoids. To represent the word w_j, we estimate a Gaussian from these nested ellipsoids. The use of Gaussian representations is computationally convenient and intuitively acts as a form of smoothing. In Section 3.2.1 we first explain how these Gaussians are estimated, after which we explain how they are used for measuring word similarity in Section 3.2.2

3.2.1 Estimating Gaussians

Rather than estimating the Gaussian representation of a given word w_j from the vector $\tilde{p}_{w_j}$ and offsets b_j^i directly, we will estimate it from the locations of the words that are inside the corresponding ellipsoids. In this way, we can also take into account the distribution of words within each ellipsoid. In particular, for each word w_j, we first determine a set of words w whose vector p_w is inside these ellipsoids. Specifically, for each word w that occurs at least once in the context of w_j, or is among the 10 closest neighbors in the vector space of such a word, we test whether $\phi(p_w) \cdot \tilde{p}_{w_j} < -b_j^1$, i.e. whether w is in the outer ellipsoid for w_j. Let M_{w_j} be the set of all words w for which this is the case. We then represent w_j as the Gaussian $G(.; \mu_{w_j}, C_{w_j})$, where μ_{w_j} and C_{w_j} are estimated as the sample mean and covariance of the set $\{p_w \mid w \in M_{w_j}\}$.

We also consider a variant in which each word w from M_{w_j} is weighted as follows. First, we determine the largest k in $\{1, ..., n_j\}$ for which $\phi(p_w) \cdot \tilde{p}_{w_j} < -b_j^k$; note that since $w \in M_{w_j}$ such a k exists. The weight λ_w of w is defined as the PPMI value that is associated with the set W_j^k. When using this weighted setting, the mean μ_{w_j} and covariance matrix C_{w_j} are estimated as:

$$\mu_{w_j} = \frac{\sum_{w \in M_{w_j}} \lambda_w p_w}{\sum_{w \in M_{w_j}} \lambda_w}$$

$$C_{w_j} = \frac{\sum_{w \in M_{w_j}} \lambda_w (p_w - \mu)(p_w - \mu)^T}{\sum_{w \in M_{w_j}} \lambda_w}$$

Note that the two proposed methods to estimate the Gaussian $G(.; \mu_{w_j}, C_{w_j})$ do not depend on the choice of kernel, hence they could also be applied in combination with a linear kernel. However, given the close relationships between Gaussians and ellipsoids, we can expect quadratic kernels to lead to higher-quality representations. This will be confirmed experimentally in Section 4.

3.2.2 Measuring similarity

To compute the similarity between w and w', based on the associated Gaussians, we consider two alternatives. First, following (Vilnis and McCallum, 2015), we consider the inner product, defined as follows:

$$E(w, w') = \int G(x; \mu_w, C_w) G(x; \mu_{w'}, C_{w'}) dx$$
$$= G(0; \mu_w - \mu_{w'}, C_w + C_{w'})$$

The second alternative is the Jensen-Shannon divergence, given by:

$$JS(w, w') = KL(f_w \| f_{w'}) + KL(f_{w'} \| f_w)$$

with $f_w = G(.; \mu_w, C_w)$, $f_{w'} = G(.; \mu_{w'}, C_{w'})$, and KL the Kullback-Leibler divergence. When computing the KL-divergence we add a small value δ to the diagonal elements of the covariance matrices, following (Vilnis and McCallum, 2015); we used 0.01. This is needed, as for rare words, the covariance matrix may otherwise be singular.

Finally, to measure the degree to which w entails w', we use KL-divergence, again in accordance with (Vilnis and McCallum, 2015).

4 Experiments

In this section we evaluate both the vector and region representations produced by our model. In our experiments, we have used the Wikipedia dump from November 2nd, 2015 consisting of 1,335,766,618 tokens. We used a basic text

preprocessing strategy, which involved removing punctuations, removing HTML/XML tags and lowercasing all tokens. We have removed words with less than 10 occurrences in the entire corpus. We used the Apache sentence segmentation tool[2] to detect sentence boundaries. In all our experiments, we have set the number of dimensions as 300, which was found to be a good choice in previous work, e.g. (Pennington et al., 2014). We use a context window of 10 words before and after the target word, but without crossing sentence boundaries. The number of iterations for SGD was set to 20. The results of all baseline models have been obtained using their publicly available implementations. We have used 10 negative samples in the word2vec code, which gave better results than the default value of 5. For the baseline models, we have used the default settings, apart from the D-GloVe model for which no default values were provided by the authors. For D-GloVe, we have therefore tuned the parameters using the ranges discussed in (Jameel and Schockaert, 2016). Specifically we have used the parameters that gave the best results on the Google Analogy Test Set (see below).

As baselines we have used the following standard word embedding models: the Skip-gram (SG) and Continuous Bag-of-Words (CBOW) models[3], proposed in (Mikolov et al., 2013a), the GloVe model[4], proposed in (Pennington et al., 2014), and the D-GloVe model[5] proposed in (Jameel and Schockaert, 2016). We have also compared against the Gaussian word embedding model[6] from (Vilnis and McCallum, 2015), using the means of the Gaussians as vector representations, and the Gaussians themselves as region representations. As in (Vilnis and McCallum, 2015), we consider two variants: one with diagonal covariance matrices (Gauss-D) and one with spherical covariance matrices (Gauss-S). For our model, we will consider the following configurations:

Reg-li-cos word vectors, obtained using linear kernel, compared using cosine similarity;

[2] https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.sentdetect

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

[4] https://nlp.stanford.edu/projects/glove/

[5] https://github.com/bashthebuilder/pGlove

[6] https://github.com/seomoz/word2gauss

Table 1: Results for the analogy completion task (accuracy). Reg-li-* and Reg-qu-* are our models with a linear and quadratic kernel.

	Gsem	Gsyn	MSR
SG	71.5	64.2	**68.6**
CBOW	74.2	62.3	66.2
GloVe	80.2	58.0	50.3
D-GloVe	**81.4**	59.1	59.6
Gauss-D-cos	61.5	53.6	50.7
Gauss-D-eucl	61.5	53.6	50.7
Gauss-S-cos	61.2	53.2	49.8
Gauss-S-eucl	61.4	53.3	49.8
Reg-li-cos	77.8	62.4	62.6
Reg-li-eucl	77.9	62.6	62.6
Reg-qu-cos	78.6	**65.7**	63.5
Reg-qu-eucl	78.7	**65.7**	63.6

Reg-li-eucl word vectors, obtained using linear kernel, compared using Euclidean distance;

Reg-qu-cos word vectors, obtained using quadratic kernel, compared using cosine similarity;

Reg-qu-eucl word vectors, obtained using quadratic kernel, compared using Euclidean distance;

Reg-li-prod Gaussian word regions, obtained using linear kernel, compared using the inner product E;

Reg-li-wprod Gaussian word regions estimated using the weighted variant, obtained using linear kernel, compared using the inner product E;

Reg-li-JS Gaussian word regions, obtained using linear kernel, compared using the Jensen-Shannon divergence;

Reg-li-wJS Gaussian word regions estimated using the weighted variant, obtained using linear kernel, compared using Jensen-Shannon divergence.

4.1 Analogy Completion

Analogy completion is a standard evaluation task for word embeddings. Given a pair (w_1, w_2) and a word w_3 the goal is to find the word w_4 such that w_3 and w_4 are related in the same way as w_1 and w_2. To solve this task, we predict the word w_4 which is most similar to $w_2 - w_1 + w_3$, either in terms of cosine similarity or Euclidean distance. The evaluation metric is accuracy. We use two popular benchmark data sets: the Google Analogy

Test Set[7] and the Microsoft Research Syntactic Analogies Dataset[8]. The former contains both semantic and syntactic relations, for which we show the results separately, respectively referred to as Gsem and Gsyn; the latter only contains syntactic relations and will be referred to as MSR. The results are shown in Table 1. Recall that the parameters of D-GloVe were tuned on the Google Analogy Test Set, hence the results reported for this model for Gsem and Gsyn might be slightly higher than what would normally be obtained. Note that for our model, we can only use word vectors for this task.

We outperform SG and CBOW for Gsem and Gsyn but not for MSR, and we outperform GloVe and D-GloVe for Gsyn and MSR but not for Gsem. The vectors from the Gaussian embedding model are not competitive for this task. For our model, using Euclidean distance slightly outperforms using cosine. For GloVe, SG and CBOW, we only show results for cosine, as this led to the best results. For D-GloVe, we used the likelihood-based similarity measure proposed in the original paper, which was found to outperform both cosine and Euclidean distance for that model.

For our model, the quadratic kernel leads to better results than the linear kernel, which is somewhat surprising since this task evaluates a kind of linear regularity. This suggests that the additional flexibility that results from the quadratic kernel leads to more faithful context word representations, which in turn improves the quality of the target word vectors.

4.2 Similarity Estimation

To evaluate our model's ability to measure similarity we use 12 standard evaluation sets[9], for which we will use the following abbreviations: S1: MTurk-287, S2:RG-65, S3:MC-30, S4:WS-353-REL, S5:WS-353-ALL, S6:RW-STANFORD, S7: YP-130, S8:SIMLEX-999, S9:VERB-143, S10: WS-353-SIM, S11:MTurk-771, S12:MEN-TR-3K. Each of these datasets contains similarity judgements for a number of word pairs. The task evaluates to what extent the similarity scores produced by a given word embedding model lead to the same ordering of the word pairs as the provided ground truth judgments. The evaluation metric is the Spearman ρ rank correlation coefficient. For this task, we can either use word vectors or word regions. The results are shown in Table 2.

For our model, the best results are obtained when using word vectors and the Euclidean distance (Reg-qu-eucl), although the differences with the word regions (Reg-qu-wprod) are small. We use *prod* to refer to the configuration where similarity is estimated using the inner product, whereas we write JS for the configurations that use Jensen-Shannon divergence. Moreover, we use *wprod* and wJS to refer to the weighted variant for estimating the Gaussians. We can again observe that using a quadratic kernel leads to better results than using a linear kernel. As the weighted versions for estimating the Gaussians do not lead to a clear improvement, for the remainder of this paper we will only consider the unweighted variant.

With the exception of S9, our model substantially outperforms the Gaussian word embedding model. Of the standard models SG and D-GloVe obtain the strongest performance. Compared to our model, these baseline models achieve similar results for S2, S10, S11 and S12, worse results for S1, S3, S4, S5, S6 and better results for S7, S8 and S9. Two general trends can be observed. First, the data sets where our model performs better tend to be datasets which describe semantic relatedness rather than pure synonymy. Second, the standard models appear to perform better on data sets that contain verbs and adjectives, as opposed to nouns.

4.3 Modeling properties

In (Rubinstein et al., 2015), it was analysed to what extent word embeddings can be used to identify concepts that satisfy a given attribute. While good results were obtained for taxonomic properties, attributive properties such as 'dangerous', 'round', or 'blue' proved to be considerably more problematic. We may expect region-based models to perform well on this task, since each of these attributes then explicitly corresponds to a region in space. To test this hypothesis, Table 3 shows the results for the same 7 taxonomic properties and 13 attributive properties as in (Rubinstein et al., 2015), where the positive and negative examples for all 20 properties are obtained from the McRae feature norms data (McRae et al., 2005). Following (Rubinstein et al., 2015), we use

[7] https://nlp.stanford.edu/projects/glove/

[8] http://research.microsoft.com/en-us/um/people/gzweig/Pubs/myz_naacl13_test_set.tgz

[9] https://github.com/mfaruqui/eval-word-vectors

Table 2: Results for similarity estimation (Spearman ρ). Reg-li-* and Reg-qu-* are our models with a linear and quadratic kernel.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
SG	0.656	0.773	0.789	0.648	0.709	0.459	0.500	**0.415**	**0.435**	0.773	0.655	0.731
CBOW	0.644	0.768	0.740	0.532	0.622	0.419	0.341	0.361	0.343	0.707	0.597	0.693
GloVe	0.595	0.755	0.746	0.515	0.577	0.318	0.533	0.382	0.354	0.690	0.652	0.724
D-GloVe	0.659	**0.788**	0.785	0.555	0.651	0.401	**0.535**	0.413	0.388	0.778	**0.656**	**0.746**
Gauss-D-cos	0.591	0.622	0.661	0.403	0.501	0.249	0.388	0.337	0.411	0.640	0.599	0.643
Gauss-D-eucl	0.591	0.623	0.661	0.403	0.501	0.250	0.388	0.338	0.411	0.641	0.599	0.643
Gauss-D-prod	0.588	0.618	0.658	0.399	0.498	0.213	0.356	0.326	0.409	0.631	0.588	0.633
Gauss-D-JS	0.598	0.619	0.665	0.403	0.532	0.288	0.381	0.339	0.410	0.643	0.599	0.644
Gauss-S-cos	0.593	0.632	0.681	0.409	0.506	0.256	0.392	0.337	0.416	0.649	0.601	0.644
Gauss-S-eucl	0.593	0.632	0.681	0.409	0.507	0.356	0.393	0.337	0.416	0.649	0.603	0.644
Gauss-S-prod	0.591	0.619	0.659	0.403	0.505	0.312	0.389	0.328	0.412	0.633	0.591	0.633
Gauss-S-JS	0.598	0.622	0.667	0.405	0.533	0.288	0.385	0.349	0.410	0.643	0.601	0.644
Reg-li-cos	0.666	0.764	0.821	0.652	0.713	0.489	0.469	0.354	0.361	0.734	0.642	0.739
Reg-li-eucl	0.668	0.766	0.821	0.654	0.715	0.489	0.469	0.359	0.361	0.734	0.643	0.739
Reg-li-prod	0.661	0.759	0.818	0.634	0.710	0.481	0.445	0.358	0.360	0.724	0.641	0.729
Reg-li-wprod	0.663	0.761	0.819	0.638	0.711	0.482	0.446	0.359	0.361	0.725	0.642	0.731
Reg-li-JS	0.663	0.758	0.815	0.638	0.709	0.479	0.443	0.359	0.361	0.723	0.641	0.729
Reg-li-wJS	0.665	0.760	0.816	0.638	0.710	0.481	0.445	0.359	0.361	0.725	0.641	0.731
Reg-qu-cos	0.684	0.781	**0.839**	0.662	**0.723**	0.505	0.479	0.367	0.368	0.777	**0.656**	0.744
Reg-qu-eucl	**0.685**	0.781	**0.839**	**0.664**	**0.723**	**0.509**	0.479	0.367	0.368	**0.779**	**0.656**	0.744
Reg-qu-prod	0.681	0.780	0.831	0.658	0.719	0.501	0.478	0.355	0.331	0.778	0.653	0.741
Reg-qu-wprod	0.684	**0.788**	0.831	0.663	0.721	0.501	0.475	0.370	0.365	0.778	0.653	0.739
Reg-qu-JS	0.680	0.781	0.826	0.661	0.715	0.497	0.471	0.328	0.355	0.771	0.649	0.721
Reg-qu-wJS	0.678	0.782	0.824	0.662	0.712	0.498	0.469	0.326	0.351	0.771	0.644	0.720

Table 3: Results for McRae feature norms (F1). Reg-li and Reg-qu are our models with a linear and quadratic kernel.

	Taxonomic		Attributive	
	lin	quad	lin	quad
SG	0.781	0.784	0.365	0.378
CBOW	0.775	0.781	0.361	0.371
GloVe	0.785	0.786	0.364	0.377
D-GloVe	0.743	0.749	0.342	0.364
Gauss-D	0.787	0.789	0.406	0.414
Gauss-S	0.781	0.784	0.401	0.406
Reg-li	0.791	0.796	0.399	0.406
Reg-qu	**0.795**	**0.799**	**0.411**	**0.421**

5-fold cross-validation to train a binary SVM for each property and compute the average F-score due to unbalanced class label distribution. We separately present results for SVMs with a linear and a quadratic kernel. The results indeed support the hypothesis that region-based models are well-suited for this task, as both the Gaussian embedding model and our model outperform the standard word embedding models.

4.4 Hypernym Detection

For hypernym detection, we have used the following 5 benchmark data sets[10]: H1 (Baroni et al., 2012), H2 (Baroni and Lenci, 2011), H3 (Kotlerman et al., 2010), H4 (Levy et al., 2014) and H5 (Turney and Mohammad, 2015). Each of the data sets contains positive and negative examples, i.e. word pairs that are in a hypernym relation and word pairs that are not. Rather than treating this problem as a classification task, which would require selecting a threshold in addition to producing a score, we treat it as a ranking problem. In other words, we evaluate to what extent the word pairs that are in a valid hypernym relation are the ones that receive the highest scores. We use average precision as our evaluation metric.

Apart from our model, the Gaussian embedding model is the only word embedding model that can by design support unsupervised hyperynym detection. As an additional baseline, however, we also show how Skip-gram performs when using cosine similarity. While such a symmetric measure cannot faithfully model hypernyny, it was nonetheless found to be a strong baseline for hypernymy models (Vulić et al., 2016), due to the inherent difficulty of the task. We also compare with a number of standard bag-of-words based models for detecting hypernyms: WeedsPrec (Kotlerman et al., 2010), ClarkeDE (Clarke, 2009) and invCL (Lenci and Benotto, 2012). These latter models take as input the PPMI weighted co-occurrence counts.

The results are shown in Table 4, where Reg-li-KL and Reg-qu-KL refer to variants of our model

[10] https://github.com/stephenroller/emnlp2016

Table 4: Results for hypernym detection (AP). Reg-li-* and Reg-qu-* are our models with a linear and quadratic kernel.

Model	H1	H2	H3	H4	H5
WeedsPrec	0.565	0.376	0.611	0.414	0.685
ClarkeDE	0.588	0.397	0.621	0.426	0.699
invCL	0.603	0.416	0.693	0.439	0.756
SG	0.682	0.434	0.712	0.455	0.789
Gauss-D-KL	0.865	0.505	0.806	0.515	0.815
Gauss-S-KL	0.823	0.498	0.801	0.507	0.789
Gauss-D-Cos	0.846	0.499	0.801	0.509	0.811
Gauss-S-Cos	0.813	0.484	0.799	0.501	0.778
Gauss-D-KLC	0.868	0.511	0.809	0.519	0.815
Gauss-S-KLC	0.835	0.501	0.804	0.511	0.795
Reg-li-KL	0.867	0.501	0.805	0.505	0.801
Reg-qu-KL	0.871	0.512	0.811	0.521	0.814
Reg-li-Cos	0.871	0.502	0.807	0.508	0.804
Reg-qu-Cos	0.873	0.513	0.818	0.525	0.819
Reg-li-KLC	0.874	0.509	0.812	0.511	0.806
Reg-qu-KLC	**0.878**	**0.519**	**0.825**	**0.531**	**0.823**

Table 5: Results for HyperLex (Spearman ρ). Reg-li-* and Reg-qu-* are our models with a linear and quadratic kernel.

Model	All	Nouns	Verbs
WeedsPrec	0.166	0.153	0.201
ClarkeDE	0.165	0.151	0.189
invCL	0.168	0.154	0.198
SG	0.158	0.164	**0.297**
Gauss-D-KL	0.185	0.171	0.198
Gauss-S-KL	0.181	0.168	0.184
Gauss-D-Cos	0.179	0.158	0.161
Gauss-S-Cos	0.166	0.151	0.158
Gauss-D-KLC	0.191	0.177	0.199
Gauss-S-KLC	0.189	0.171	0.189
Reg-li-KL	0.181	0.165	0.179
Reg-qu-KL	0.188	0.169	0.191
Reg-li-Cos	0.184	0.168	0.181
Reg-qu-Cos	0.190	0.180	0.196
Reg-li-KLC	0.189	0.171	0.185
Reg-qu-KLC	**0.208**	**0.188**	0.201

in which Kullback-Leibler divergence is used to compare word regions. Surprisingly, both for our model and for the Gaussian embedding model, we find that using cosine similarity between the word vectors outperforms using the word regions with KL-divergence. In general, our model outperforms the Gaussian embedding model and the other baselines. Given the effectiveness of the cosine similarity, we have also experimented with the following metric:

$$hyp(w_1, w_2) = (1 - \cos(w_1, w_2)) \cdot KL(f_{w_1} || f_{w_2})$$

The results are referred to as Reg-li-KLC and Reg-qu-KLC in Table 4. These results suggest that the word regions can indeed be useful for detecting hypernymy, when used in combination with cosine similarity. Intuitively, for w_2 to be a hypernym of w_1, both words need to be similar and w_2 needs to be more general than w_1. While word regions are not needed for measuring similarity, they seem essential for modeling generality (in an unsupervised setting).

The datasets considered so far all treat hypernyms as a binary notion. In (Vulić et al., 2016) a evaluation set was introduced which contains graded hypernym pairs. The underlying intuition is that e.g. cat and dog are more typical/natural hyponyms of animal than dinosaur or amoeba. The results for this data set are shown in Table 5. In this case, we use Spearman ρ as an evaluation metric, measuring how well the rankings induced by different models correlate with the ground truth. Following (Vulić et al., 2016), we separately mention results for nouns and verbs. In the case of nouns, our findings here are broadly in agreement with those from Table 4 Interesting, for verbs we find that Skip-gram substantially outperforms the region based models, which is in accordance with our findings in the word similarity experiments.

5 Conclusions

We have proposed a new word embedding model, which is based on ordinal regression. The input to our model consists of a number of rankings, capturing how strongly each word is related to each context word in a purely ordinal way. Word vectors are then obtained by embedding these rankings in a low-dimensional vector space. Despite the fact that all quantitative information is disregarded by our model (except for constructing the rankings), it is competitive with standard methods such as Skip-gram, and in fact outperforms them in several tasks. An important advantage of our model is that it can be used to learn region representations for words, by using a quadratic kernel. Our experimental results suggest that these regions can be useful for modeling hypernymy.

Acknowledgments

This work was supported by ERC Starting Grant 637277. This work was performed using the computational facilities of the Advanced Research Computing@Cardiff (ARCCA) Division, Cardiff University. The authors would like to thank the anonymous reviewers for their insightful comments.

References

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

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*. Association for Computational Linguistics, pages 1–10.

Wei Chu and S Sathiya Keerthi. 2005. New approaches to support vector ordinal regression. In *ICML*. pages 145–152.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*. pages 112–119.

Katrin Erk. 2009. Representing words as regions in vector space. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. pages 57–65.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 462–471.

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Proc. NIPS*. pages 2121–2129.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* 57:345–420.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* .

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. Distributional vectors encode referential attributes. In *Proc. EMNLP*. pages 12–21.

Shoaib Jameel and Steven Schockaert. 2016. D-glove: A feasible least squares model for estimating word embedding densities. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 1849–1860.

Joo-Kyung Kim and Marie-Catherine de Marneffe. 2013. Deriving adjectival scales from continuous space word representations. In *Proc. EMNLP*. pages 1625–1630.

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

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of *SEM*. pages 75–79.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. In *Proc. CoNLL*. pages 171–180.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3:225–331.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods* 37:547–559.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. pages 3111–3119.

Neha Nayak. 2015. In learning hyperonyms over word embeddings. Technical report, Student technical report.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proc. EMNLP*. pages 1532–1543.

Sascha Rothe and Hinrich Schütze. 2016. Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 512–517.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. pages 726–730.

Steven Schockaert and Jae Hee Lee. 2015. Qualitative reasoning about directions in semantic spaces. In *Proceedings of the International Joint Conference on Artificial Intelligence*. pages 3207–3213.

Amnon Shashua and Anat Levin. 2002. Ranking with large margin principle: Two approaches. In *NIPS*. pages 937–944.

P. D. Turney and P. Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research* 37:141–188.

Peter D Turney and Saif M Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering* 21(03):437–476.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *International Conference on Learning Representations*.

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

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

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proc. EMNLP*. pages 1393–1398.