Morphological Priors for Probabilistic Neural Word Embeddings

Parminder Bhatia[*]
Yik Yak, Inc.
3525 Piedmont Rd NE, Building 6, Suite 500
Atlanta, GA
parminder@yikyakapp.com

Robert Guthrie[*] and **Jacob Eisenstein**
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30312 USA
{rguthrie3 + jacobe}@gatech.edu

Abstract

Word embeddings allow natural language processing systems to share statistical information across related words. These embeddings are typically based on distributional statistics, making it difficult for them to generalize to rare or unseen words. We propose to improve word embeddings by incorporating morphological information, capturing shared sub-word features. Unlike previous work that constructs word embeddings directly from morphemes, we combine morphological and distributional information in a unified probabilistic framework, in which the word embedding is a latent variable. The morphological information provides a prior distribution on the latent word embeddings, which in turn condition a likelihood function over an observed corpus. This approach yields improvements on intrinsic word similarity evaluations, and also in the downstream task of part-of-speech tagging.

1 Introduction

Word embeddings have been shown to improve many natural language processing applications, from language models (Mikolov et al., 2010) to information extraction (Collobert and Weston, 2008), and from parsing (Chen and Manning, 2014) to machine translation (Cho et al., 2014). Word embeddings leverage a classical idea in natural language processing: use distributional statistics from large amounts of unlabeled data to learn representations that allow sharing

---

[*]The first two authors contributed equally. Code is available at https://github.com/rguthrie3/MorphologicalPriorsForWordEmbeddings.

across related words (Brown et al., 1992). While this approach is undeniably effective, the long-tail nature of linguistic data ensures that there will always be words that are not observed in even the largest corpus (Zipf, 1949). There will be many other words which are observed only a handful of times, making the distributional statistics too sparse to accurately estimate the 100- or 1000-dimensional dense vectors that are typically used for word embeddings. These problems are particularly acute in morphologically rich languages like German and Turkish, where each word may have dozens of possible inflections.

Recent work has proposed to address this issue by replacing word-level embeddings with embeddings based on subword units: morphemes (Luong et al., 2013; Botha and Blunsom, 2014) or individual characters (Santos and Zadrozny, 2014; Ling et al., 2015; Kim et al., 2016). Such models leverage the fact that word meaning is often *compositional*, arising from subword components. By learning representations of subword units, it is possible to generalize to rare and unseen words.

But while morphology and orthography are sometimes a signal of semantics, there are also many cases similar spellings do *not* imply similar meanings: *better-batter*, *melon-felon*, *dessert-desert*, etc. If each word's embedding is constrained to be a deterministic function of its characters, as in prior work, then it will be difficult to learn appropriately distinct embeddings for such pairs. Automated morphological analysis may be incorrect: for example, *really* may be segmented into *re+ally*, incorrectly suggesting a similarity to *revise* and *review*. Even correct morphological segmentation may be misleading. Consider

that *incredible* and *inflammable* share a prefix *in-*, which exerts the opposite effect in these two cases.[1] Overall, a word's observed internal structure gives evidence about its meaning, but it must be possible to override this evidence when the distributional facts point in another direction.

We formalize this idea using the machinery of probabilistic graphical models. We treat word embeddings as *latent variables* (Vilnis and McCallum, 2014), which are conditioned on a prior distribution that is based on word morphology. We then maximize a variational approximation to the expected likelihood of an observed corpus of text, fitting variational parameters over latent binary word embeddings. For common words, the expected word embeddings are largely determined by the expected corpus likelihood, and thus, by the distributional statistics. For rare words, the prior plays a larger role. Since the prior distribution is a function of the morphology, it is possible to impute embeddings for unseen words after training the model.

We model word embeddings as latent binary vectors. This choice is based on linguistic theories of lexical semantics and morphology. Morphemes are viewed as adding morphosyntactic *features* to words: for example, in English, *un-* adds a negation feature (*unbelievable*), *-s* adds a plural feature, and *-ed* adds a past tense feature (Halle and Marantz, 1993). Similarly, the lexicon is often viewed as organized in terms of features: for example, the word *bachelor* carries the features HUMAN, MALE, and UNMARRIED (Katz and Fodor, 1963). Each word's semantic role within a sentence can also be characterized in terms of binary features (Dowty, 1991; Reisinger et al., 2015). Our approach is more amenable to such theoretical models than traditional distributed word embeddings. However, we can also work with the *expected word embeddings*, which are vectors of probabilities, and can therefore be expected to hold the advantages of dense distributed representations (Bengio et al., 2013).

2 Model

The modeling framework is illustrated in Figure 1, focusing on the word *sesquipedalianism*. This word is rare, but its morphology indicates several of its properties: the *-ism* suffix suggests that the word is a noun, likely describing some abstract property; the *sesqui-* prefix refers to one and a half, and so on. If the word is unknown, we must lean heavily on these intuitions, but if the word is well attested then we can rely instead on its examples in use.

It is this reasoning that our modeling framework aims to formalize. We treat word embeddings as latent variables in a joint probabilistic model. The prior distribution over a word's embedding is conditioned on its morphological structure. The embedding itself then participates, as a latent variable, in a neural sequence model over a corpus, contributing to the overall corpus likelihood. If the word appears frequently, then the corpus likelihood dominates the prior — which is equivalent to relying on the word's distributional properties. If the word appears rarely, then the prior distribution steps in, and gives a best guess as to the word's meaning.

Before describing these component pieces in detail, we first introduce some notation. The representation of word w is a latent binary vector $\boldsymbol{b}_w \in \{0,1\}^k$, where k is the size of each word embedding. As noted in the introduction, this binary representation is motivated by feature-based theories of lexical semantics (Katz and Fodor, 1963). Each word w is constructed from a set of M_w observed morphemes, $\mathcal{M}_w = (m_{w,1}, m_{w,2}, \ldots, m_{w,M_w})$. Each morpheme is in turn drawn from a finite vocabulary of size v_m, so that $m_{w,i} \in \{1, 2, \ldots, v_m\}$. Morphemes are obtained from an unsupervised morphological segmenter, which is treated as a black box. Finally, we are given a corpus, which is a sequence of words, $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, where each word $x_t \in \{1, 2, \ldots, v_w\}$, with v_w equal to the size of the vocabulary, including the token ⟨UNK⟩ for unknown words.

2.1 Prior distribution

The key differentiating property of this model is that rather than estimating word embeddings directly, we treat them as a latent variable, with a prior distribution reflecting the word's morphological proper-
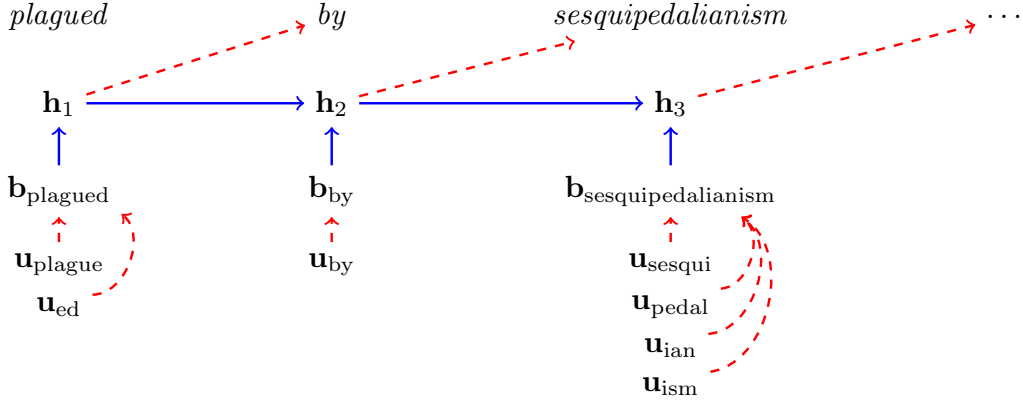
Figure 1: Model architecture, applied to the example sequence . . . *plagued by sesquipedalianism* . . . . Blue solid arrows indicate direct computation, red dashed arrows indicate probabilistic dependency. For simplicity, we present our models as recurrent neural networks rather than long short-term memories (LSTMs).

ties. To characterize this prior distribution, each morpheme m is associated with an embedding of its own, $\boldsymbol{u}_m \in \mathbb{R}^k$, where k is again the embedding size. Then for position i of the word embedding $\boldsymbol{b}_w$, we have the following prior,

$$b_{w,i} \sim \text{Bernoulli}\left(\sigma(\sum_{m\in\mathcal{M}_w} u_{m,i})\right), \quad (1)$$

where $\sigma(\cdot)$ indicates the sigmoid function. The prior log-likelihood for a set of word embeddings is,

$$\log P(\boldsymbol{b}; \mathcal{M}, \boldsymbol{u}) \quad (2)$$

$$= \sum_{w}^{V_w} \log P(\boldsymbol{b}_w; \mathcal{M}_w, \boldsymbol{u}) \quad (3)$$

$$= \sum_{w}^{V_w} \sum_{i}^{k} \log P(b_{w,i}; \mathcal{M}_w, \boldsymbol{u}) \quad (4)$$

$$= \sum_{w}^{V_w} \sum_{i}^{k} b_{w,i} \log \sigma\left(\sum_{m\in\mathcal{M}_w} u_{m,i}\right) \quad (5)$$

$$+ (1 - b_{w,i}) \log \left(1 - \sigma\left(\sum_{m\in\mathcal{M}_w} u_{m,i}\right)\right).$$

2.2 Expected likelihood

The corpus likelihood is computed via a recurrent neural network language model (Mikolov et al., 2010, RNNLM), which is a generative model of sequences of tokens. In the RNNLM, the probability of each word is conditioned on all preceding words through a recurrently updated state vector. This state vector in turn depends on the embeddings of the previous words, through the following update equations:

$$\boldsymbol{h}_t = f(\boldsymbol{b}_{x_t}, \boldsymbol{h}_{t-1}) \quad (6)$$

$$x_{t+1} \sim \text{Multinomial}\left(\text{Softmax}\left[\mathbf{V}\boldsymbol{h}_t\right]\right). \quad (7)$$

The function $f(\cdot)$ is a recurrent update equation; in the RNN, it corresponds to $\sigma(\Theta \boldsymbol{h}_{t-1} + \boldsymbol{b}_{x_t})$, where $\sigma(\cdot)$ is the elementwise sigmoid function. The matrix $\mathbf{V} \in \mathbb{R}^{v\times k}$ contains the "output embeddings" of each word in the vocabulary. We can then define the conditional log-likelihood of a corpus $\boldsymbol{x} = (x_1, x_2, \ldots x_N)$ as,

$$\log P(\boldsymbol{x} \mid \boldsymbol{b}) = \sum_{t}^{N} \log P(x_t \mid \boldsymbol{h}_{t-1}, \boldsymbol{b}). \quad (8)$$

Since $\boldsymbol{h}_{t-1}$ is deterministically computed from $\boldsymbol{x}_{1:t-1}$ (conditioned on $\boldsymbol{b}$), we can equivalently write the log-likelihood as,

$$\log P(\boldsymbol{x} \mid \boldsymbol{b}) = \sum_{t} \log P(x_t \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{b}). \quad (9)$$

This same notation can be applied to compute the likelihood under a long-short term memory (LSTM) language model (Sundermeyer et al., 2012). The only difference is that the recurrence function $f(\cdot)$ from Equation 6 now becomes more complex, including the input, output, and forget gates, and the recurrent state $\boldsymbol{h}_t$ now includes the memory cell. As the LSTM

update equations are well known, we focus on the more concise RNN notation, but we employ LSTMs in all experiments due to their better ability to capture long-range dependencies.

2.3 Variational approximation

Inference on the marginal likelihood $P(\boldsymbol{x}_{1:N}) = \int P(\boldsymbol{x}_{1:N}, \boldsymbol{b}) d\boldsymbol{b}$ is intractable. We address this issue by making a variational approximation,

$$\log P(\boldsymbol{x}) = \log \sum_{\boldsymbol{b}} P(\boldsymbol{x} \mid \boldsymbol{b}) P(\boldsymbol{b}) \qquad (10)$$

$$= \log \sum_{\boldsymbol{b}} \frac{Q(\boldsymbol{b})}{Q(\boldsymbol{b})} P(\boldsymbol{x} \mid \boldsymbol{b}) P(\boldsymbol{b}) \qquad (11)$$

$$= \log E_q \left[P(\boldsymbol{x} \mid \boldsymbol{b}) \frac{P(\boldsymbol{b})}{Q(\boldsymbol{b})} \right] \qquad (12)$$

$$\geq E_q [\log P(\boldsymbol{x} \mid \boldsymbol{b}) + \log P(\boldsymbol{b}) - \log Q(\boldsymbol{b})] \qquad (13)$$

The variational distribution $Q(\boldsymbol{b})$ is defined using a fully factorized mean field approximation,

$$Q(\boldsymbol{b}; \boldsymbol{\gamma}) = \prod_w^{v_w} \prod_i^k q(b_{w,i}; \gamma_{w,i}). \qquad (14)$$

The variational distribution is a product of Bernoullis, with parameters $\gamma_{w,j} \in [0, 1]$. In the evaluations that follow, we use the expected word embeddings $q(\boldsymbol{b}_w)$, which are dense vectors in $[0, 1]^k$. We can then use $Q(\cdot)$ to place a variational lower bound on the expected conditional likelihood,

Even with this variational approximation, the expected log-likelihood is still intractable to compute. In recurrent neural network language models, each word x_t is conditioned on the entire prior history, $\boldsymbol{x}_{1:t-1}$ — indeed, this is one of the key advantages over fixed-length n-gram models. However, this means that the individual expected log probabilities involve not just the word embedding of x_t and its immediate predecessor, but rather, the embeddings

of *all* words in the sequence $\boldsymbol{x}_{1:t}$:

$$E_q [\log P(\boldsymbol{x} \mid \boldsymbol{b})] \qquad (15)$$

$$= \sum_t^N E_q [\log P(x_t \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{b})] \qquad (16)$$

$$= \sum_t^N \sum_{\{\boldsymbol{b}_w : w \in \boldsymbol{x}_{1:t}\}} Q(\{\boldsymbol{b}_w : w \in \boldsymbol{x}_{1:t}\})$$
$$\times \log P(x_t \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{b}). \qquad (17)$$

We therefore make a further approximation by taking a local expectation over the recurrent state,

$$E_q [\boldsymbol{h}_t] \approx f(E_q [\boldsymbol{b}_{x_t}], E_q [\boldsymbol{h}_{t-1}]) \qquad (18)$$

$$E_q [\log P(x_t \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{b})] \approx \log \text{Softmax} (\mathbf{V} E_q [\boldsymbol{h}_t]). \qquad (19)$$

This approximation means that we do not propagate uncertainty about $\boldsymbol{h}_t$ through the recurrent update or through the likelihood function, but rather, we use local point estimates. Alternative methods such as variational autoencoders (Chung et al., 2015) or sequential Monte Carlo (de Freitas et al., 2000) might provide better and more principled approximations, but this direction is left for future work.

Variational bounds in the form of Equation 13 can generally be expressed as a difference between an expected log-likelihood term and a term for the Kullback-Leibler (KL) divergence between the prior distribution $P(\boldsymbol{b})$ and the variational distribution $Q(\boldsymbol{b})$ (Wainwright and Jordan, 2008). Incorporating the approximation in Equation 19, the resulting objective is,

$$\mathcal{L} = \sum_t^N \log P(x_t \mid \boldsymbol{x}_{1:t-1}; E_q[\boldsymbol{b}])$$
$$- D_{KL}(Q(\boldsymbol{b}) \parallel P(\boldsymbol{b})). \qquad (20)$$

The KL divergence is equal to,

$$D_{KL}(Q(\boldsymbol{b}) \parallel P(\boldsymbol{b})) \tag{21}$$

$$= \sum_w^{v_w} \sum_i^k D_{KL}(q(b_{w,i}) \parallel P(b_{w,i})) \tag{22}$$

$$= \sum_w^{v_w} \sum_i^k \gamma_{w,i} \log \sigma(\sum_{m \in \mathcal{M}_w} u_{m,i})$$
$$+ (1 - \gamma_{w,i}) \log(1 - \sigma(\sum_{m \in \mathcal{M}_w} u_{m,i}))$$
$$- \gamma_{w,i} \log \gamma_{w,i} - (1 - \gamma_{w,i}) \log(1 - \gamma_{w,i}). \tag{23}$$

Each term in the variational bound can be easily constructed in a computation graph, enabling automatic differentiation and the application of standard stochastic optimization techniques.

3 Implementation

The objective function is given by the variational lower bound in Equation 20, using the approximation to the conditional likelihood described in Equation 19. This function is optimized in terms of several parameters:

- the morpheme embeddings, $\{\boldsymbol{u}_m\}_{m \in 1 \ldots v_m}$;

- the variational parameters on the word embeddings, $\{\boldsymbol{\gamma}\}_{w \in 1 \ldots v_w}$;

- the output word embeddings $\mathbf{V}$;

- the parameter of the recurrence function, Θ.

Each of these parameters is updated via the `RMSProp` online learning algorithm (Tieleman and Hinton, 2012). The model and baseline (described below) are implemented in `blocks` (van Merriënboer et al., 2015). In the remainder of the paper, we refer to our model as VAREMBED.

3.1 Data and preprocessing

All embeddings are trained on 22 million tokens from the the North American News Text (NANT) corpus (Graff, 1995). We use an initial vocabulary of 50,000 words, with a special ⟨UNK⟩ token for words that are not among the 50,000 most common. We then perform downcasing and convert all numeric tokens to a special ⟨NUM⟩ token. After these

steps, the vocabulary size decreases to 48,986. Note that the method can impute word embeddings for out-of-vocabulary words under the prior distribution $P(\boldsymbol{b}; \mathcal{M}, \boldsymbol{u})$; however, it is still necessary to decide on a vocabulary size to determine the number of variational parameters $\boldsymbol{\gamma}$ and output embeddings to estimate.

Unsupervised morphological segmentation is performed using Morfessor (Creutz and Lagus, 2002), with a maximum of sixteen morphemes per word. This results in a total of 14,000 morphemes, which includes stems for monomorphemic words. We do not rely on any labeled information about morphological structure, although the incorporation of gold morphological analysis is a promising topic for future work.

3.2 Learning details

The LSTM parameters are initialized uniformly in the range $[-0.08, 0.08]$. The word embeddings are initialized using pre-trained `word2vec` embeddings. We train the model for 15 epochs, with an initial learning rate of 0.01, a decay of 0.97 per epoch, and minibatches of size 25. We clip the norm of the gradients (normalized by minibatch size) at 1, using the default settings in the `RMSprop` implementation in `blocks`. These choices are motivated by prior work (Zaremba et al., 2014). After each iteration, we compute the objective function on the development set; when the objective does not improve beyond a small threshold, we halve the learning rate.

Training takes roughly one hour per iteration using an NVIDIA 670 GTX, which is a commodity graphics processing unit (GPU) for gaming. This is nearly identical to the training time required for our reimplementation of the algorithm of Botha and Blunsom (2014), described below.

3.3 Baseline

The most comparable approach is that of Botha and Blunsom (2014). In their work, embeddings are estimated for each morpheme, as well as for each in-vocabulary word. The final embedding for a word is then the sum of these embeddings, e.g.,

$$\overline{\text{greenhouse}} = \overline{greenhouse} + \overline{green} + \overline{house}, \tag{24}$$

where the italicized elements represent learned embeddings.

We build a baseline that is closely inspired by this approach, which we call SUMEMBED. The key difference is that while Botha and Blunsom (2014) build on the log-bilinear language model (Mnih and Hinton, 2007), we use the same LSTM-based architecture as in our own model implementation. This enables our evaluation to focus on the critical difference between the two approaches: the use of latent variables rather than summation to model the word embeddings. As with our method, we used pre-trained `word2vec` embeddings to initialize the model.

3.4 Number of parameters

The dominant terms in the overall number of parameters are the (expected) word embeddings themselves. The variational parameters of the input word embeddings, γ, are of size $k \times v_w$. The output word embeddings are of size $\#|\boldsymbol{h}| \times v_w$. The morpheme embeddings are of size $k \times v_m$, with $v_m \ll v_w$. In our main experiments, we set $v_w = 48,896$ (see above), $k = 128$, and $\#|\boldsymbol{h}| = 128$. After including the character embeddings and the parameters of the recurrent models, the total number of parameters is roughly 12.8 million. This is identical to number of parameters in the SUMEMBED baseline.

4 Evaluation

Our evaluation compares the following embeddings:

WORD2VEC We train the popular `word2vec` CBOW (continuous bag of words) model (Mikolov et al., 2013), using the `gensim` implementation.

SUMEMBED We compare against the baseline described in § 3.3, which can be viewed as a reimplementation of the compositional model of Botha and Blunsom (2014).

VAREMBED For our model, we take the *expected* embeddings $E_q[\boldsymbol{b}]$, and then pass them through an inverse sigmoid function to obtain values over the entire real line.

4.1 Word similarity

Our first evaluation is based on two classical word similarity datasets: Wordsim353 (Finkelstein et al.,

2001) and the Stanford "rare words" (rw) dataset (Luong et al., 2013). We report Spearmann's ρ, a measure of rank correlation, evaluating on both the entire vocabulary as well as the subset of in-vocabulary words.

As shown in Table 1, VAREMBED consistently outperforms SUMEMBED on both datasets. On the subset of in-vocabulary words, WORD2VEC gives slightly better results on the wordsim words that are in the NANT vocabulary, but is not applicable to the complete dataset. On the rare words dataset, WORD2VEC performs considerably worse than both morphology-based models, matching the findings of Luong et al. (2013) and Botha and Blunsom (2014) regarding the importance of morphology for doing well on this dataset.

4.2 Alignment with lexical semantic features

Recent work questions whether these word similarity metrics are predictive of performance on downstream tasks (Faruqui et al., 2016). The QVEC statistic is another intrinsic evaluation method, which has been shown to be better correlated with downstream tasks (Tsvetkov et al., 2015). This metric measures the alignment between word embeddings and a set of lexical semantic features. Specifically, we use the semcor noun verb supersenses oracle provided at the qvec github repository.[2]

As shown in Table 2, VAREMBED outperforms SUMEMBED on the full lexicon, and gives similar performance to WORD2VEC on the set of in-vocabulary words. We also consider the morpheme embeddings alone. For SUMEMBED, this means that we construct the word embedding from the sum of the embeddings for its morphemes, *without* the additional embedding per word. For VAREMBED, we use the expected embedding under the prior distribution $E[\boldsymbol{b} \mid \boldsymbol{c}]$. The results for these representations are shown in the bottom half of Table 2, revealing that VAREMBED learns much more meaningful embeddings at the morpheme level, while much of the power of SUMEMBED seems to come from the word embeddings.

---

[2]`https://github.com/ytsvetko/qvec`

	WORD2VEC	SUMEMBED	VAREMBED
Wordsim353			
all words (353)	n/a	42.9	**48.8**
in-vocab (348)	**51.4**	45.9	51.3
rare words (rw)			
all words (2034)	n/a	23.0	**24.0**
in-vocab (715)	33.6	37.3	**44.1**

Table 1: Word similarity evaluation results, as measured by Spearmann's $\rho \times 100$. WORD2VEC cannot be evaluated on all words, because embeddings are not available for out-of-vocabulary words. The total number of words in each dataset is indicated in parentheses.

	all words (4199)	in vocab (3997)
WORD2VEC	n/a	**34.8**
SUMEMBED	32.8	33.5
VAREMBED	**33.6**	34.7
morphemes only		
SUMEMBED	24.7	25.1
VAREMBED	**30.2**	**31.0**

Table 2: Alignment with lexical semantic features, as measured by QVEC. Higher scores are better, with a maximum possible score of 100.

	dev	test
WORD2VEC	92.42	92.40
SUMEMBED	**93.26**	**93.26**
VAREMBED	93.05	93.09

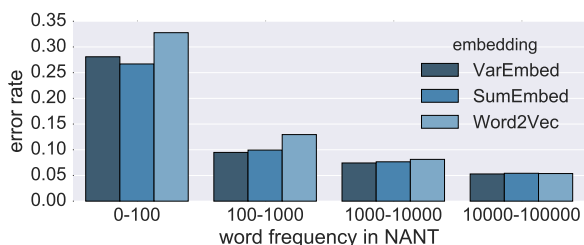**Table 3:** Part-of-speech tagging accuracies

Figure 2: Error rates by word frequency.

4.3 Part-of-speech tagging

Our final evaluation is on the downstream task of part-of-speech tagging, using the Penn Treebank. We build a simple classification-based tagger, using a feedforward neural network. (This is not intended as an alternative to state-of-the-art tagging algorithms, but as a comparison of the syntactic utility of the information encoded in the word embeddings.) The inputs to the network are the concatenated embeddings of the five word neighborhood $(x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2})$; as in all evaluations, 128-dimensional embeddings are used, so the total size of the input is 640. This input is fed into a network with two hidden layers of size 625, and a softmax output layer over all tags. We train using RMSProp (Tieleman and Hinton, 2012).

Results are shown in Table 3. Both morphologically-informed embeddings are significantly better to WORD2VEC ($p < .01$, two-tailed binomial test), but the difference between SUMEMBED and VAREMBED is not significant

at $p < .05$. Figure 2 breaks down the errors by word frequency. As shown in the figure, the tagger based on WORD2VEC performs poorly for rare words, which is expected because these embeddings are estimated from sparse distributional statistics. SUMEMBED is slightly better on the rarest words (the $0 - 100$ group accounts for roughly 10% of all tokens). In this case, it appears that this simple additive model is better, since the distributional statistics are too sparse to offer much improvement. The probabilistic VAREMBED embeddings are best for all other frequency groups, showing that it effectively combines morphology and distributional statistics.

5 Related work

Adding side information to word embeddings An alternative approach to incorporating additional

information into word embeddings is to constrain the embeddings of semantically-related words to be similar. Such work typically draws on existing lexical semantic resources such as WordNet. For example, Yu and Dredze (2014) define a joint training objective, in which the word embedding must predict not only neighboring word tokens in a corpus, but also related word *types* in a semantic resource; a similar approach is taken by Bian et al. (2014). Alternatively, Faruqui et al. (2015) propose to "retrofit" pre-trained word embeddings over a semantic network. Both retrofitting and our own approach treat the true word embeddings as latent variables, from which the pre-trained word embeddings are stochastically emitted. However, a key difference from our approach is that the underlying representation in these prior works is relational, and not generative. These methods can capture similarity between words in a relational lexicon such as WordNet, but they do not offer a generative account of how (approximate) meaning is constructed from orthography or morphology.

Word embeddings and morphology The SUMEMBED baseline is based on the work of Botha and Blunsom (2014), in which words are segmented into morphemes using MORFESSOR (Creutz and Lagus, 2002), and then word representations are computed through addition of morpheme representations. A key modeling difference from this prior work is that rather than computing word embeddings directly and deterministically from subcomponent embeddings (morphemes or characters, as in (Ling et al., 2015; Kim et al., 2016)), we use these subcomponents to define a prior distribution, which can be overridden by distributional statistics for common words. Other work exploits morphology by training word embeddings to optimize a joint objective over distributional statistics and rich, morphologically-augmented part of speech tags (Cotterell and Schütze, 2015). This can yield better word embeddings, but does not provide a way to compute embeddings for unseen words, as our approach does.

Recent work by Cotterell et al. (2016) extends the idea of retrofitting, which was based on semantic similarity, to a morphological framework. In this model, embeddings are learned for morphemes as well as for words, and each word embedding is conditioned on the sum of the morpheme embeddings,

using a multivariate Gaussian. The covariance of this Gaussian prior is set to the inverse of the number of examples in the training corpus, which has the effect of letting the morphology play a larger role for rare or unseen words. Like all retrofitting approaches, this method is applied in a pipeline fashion after training word embeddings on a large corpus; in contrast, our approach is a joint model over the morphology and corpus. Another practical difference is that Cotterell et al. (2016) use gold morphological features, while we use an automated morphological segmentation.

Latent word embeddings Word embeddings are typically treated as a parameter, and are optimized through point estimation (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2010). Current models use word embeddings with hundreds or even thousands of parameters per word, yet many words are observed only a handful of times. It is therefore natural to consider whether it might be beneficial to model uncertainty over word embeddings. Vilnis and McCallum (2014) propose to model Gaussian densities over dense vector word embeddings. They estimate the parameters of the Gaussian directly, and, unlike our work, do not consider using orthographic information as a prior distribution. This is easy to do in the latent binary framework proposed here, which is also a better fit for some theoretical models of lexical semantics (Katz and Fodor, 1963; Reisinger et al., 2015). This view is shared by Kruszewski et al. (2015), who induce binary word representations using labeled data of lexical semantic entailment relations, and by Henderson and Popa (2016), who take a mean field approximation over binary representations of lexical semantic features to induce hyponymy relations.

More broadly, our work is inspired by recent efforts to combine directed graphical models with discriminatively trained "deep learning" architectures. The variational autoencoder (Kingma and Welling, 2014), neural variational inference (Mnih and Gregor, 2014; Miao et al., 2016), and black box variational inference (Ranganath et al., 2014) all propose to use a neural network to compute the variational approximation. These ideas are employed by Chung et al. (2015) in the variational recurrent neural network, which places a latent continuous variable at each time step. In contrast, we have a dictionary of latent vari-

ables — the word embeddings — which introduce uncertainty over the hidden state h_t in a standard recurrent neural network or LSTM. We train this model by employing a mean field approximation, but these more recent techniques for neural variational inference may also be applicable. We plan to explore this possibility in future work.

6 Conclusion and future work

We present a model that unifies compositional and distributional perspectives on lexical semantics, through the machinery of Bayesian latent variable models. In this framework, our prior expectations of word meaning are based on internal structure, but these expectations can be overridden by distributional statistics. The model is based on the very successful long-short term memory (LSTM) for sequence modeling, and while it employs a Bayesian justification, its inference and estimation are little more complicated than a standard LSTM. This demonstrates the advantages of reasoning about uncertainty even when working in a "neural" paradigm.

This work represents a first step, and we see many possibilities for improving performance by extending it. Clearly we would expect this model to be more effective in languages with richer morphological structure than English, and we plan to explore this possibility in future work. From a modeling perspective, our prior distribution merely sums the morpheme embeddings, but a more accurate model might account for sequential or combinatorial structure, through a recurrent (Ling et al., 2015), recursive (Luong et al., 2013), or convolutional architecture (Kim et al., 2016). There appears to be no technical obstacle to imposing such structure in the prior distribution. Furthermore, while we build the prior distribution from morphemes, it is natural to ask whether characters might be a better underlying representation: character-based models may generalize well to non-word tokens such as names and abbreviations, they do not require morphological segmentation, and they require a much smaller number of underlying embeddings. On the other hand, morphemes encode rich regularities across words, which may make a morphologically-informed prior easier to learn than a prior which works directly at the character level. It is possible that this tradeoff could be transcended

by combining characters and morphemes in a single model.

Another advantage of latent variable models is that they admit partial supervision. If we follow Tsvetkov et al. (2015) in the argument that word embeddings should correspond to lexical semantic features, then an inventory of such features could be used as a source of partial supervision, thus locking dimensions of the word embeddings to specific semantic properties. This would complement the graph-based "retrofitting" supervision proposed by Faruqui et al. (2015), by instead placing supervision at the level of individual words.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Machine Learning and Knowledge Discovery in Databases*, pages 132–148. Springer.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 740–750.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk,

and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Neural Information Processing Systems (NIPS)*, Montréal.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 160–167.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, CO, May.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the Association for Computational Linguistics (ACL)*, Berlin, August.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 21–30. Association for Computational Linguistics.

João FG de Freitas, Mahesan Niranjan, Andrew H. Gee, and Arnaud Doucet. 2000. Sequential monte carlo methods to train neural network models. *Neural computation*, 12(4):955–993.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, pages 547–619.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, CO, May.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arxiv*, 1605.02276.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*, pages 406–414. ACM.

David Graff. 1995. North american news text corpus.

Morris Halle and Alec Marantz. 1993. Distributed morphology and the pieces of inflection. In Kenneth L. Hale and Samuel J. Keyser, editors, *The view from building 20*. MIT Press, Cambridge, MA.

James Henderson and Diana Nicoleta Popa. 2016. A vector space for distributional semantics for entailment.

In *Proceedings of the Association for Computational Linguistics (ACL)*, Berlin, August.

Jerrold J Katz and Jerry A Fodor. 1963. The structure of a semantic theory. *Language*, pages 170–210.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, Lisbon, September.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

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

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1791–1799.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Rajesh Ranganath, Sean Gerrish, and David Blei. 2014. Black box variational inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 814–822.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1818–1826.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Proceedings of INTERSPEECH*.

Tijman Tieleman and Geoffrey Hinton. 2012. Lecture 6.5: Rmsprop. Technical report, Coursera Neural Networks for Machine Learning.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, Lisbon, September.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *CoRR*, abs/1412.6623.

Martin J. Wainwright and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 545–550, Baltimore, MD.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

George Kingsley Zipf. 1949. *Human behavior and the principle of least effort*. Addison-Wesley.