Word Semantic Representations using Bayesian Probabilistic Tensor Factorization

Jingwei Zhang and **Jeremy Salwen**
Columbia University
Computer Science
New York, NY 10027, USA
{jz2541,jas2312}@columbia.edu

Michael Glass and **Alfio Gliozzo**
IBM T.J. Waston Research
Yorktown Heights, NY 10598, USA
{mrglass,gliozzo}@us.ibm.com

Abstract

Many forms of word relatedness have been developed, providing different perspectives on word similarity. We introduce a Bayesian probabilistic tensor factorization model for synthesizing a single word vector representation and per-perspective linear transformations from any number of word similarity matrices. The resulting word vectors, when combined with the per-perspective linear transformation, approximately recreate while also regularizing and generalizing, each word similarity perspective.

Our method can combine manually created semantic resources with neural word embeddings to separate synonyms and antonyms, and is capable of generalizing to words outside the vocabulary of any particular perspective. We evaluated the word embeddings with GRE antonym questions, the result achieves the state-of-the-art performance.

1 Introduction

In recent years, vector space models (VSMs) have been proved successful in solving various NLP tasks including named entity recognition, part-of-speech tagging, parsing, semantic role-labeling and answering synonym or analogy questions (Turney et al., 2010; Collobert et al., 2011). Also, VSMs are reported performing well on tasks involving the measurement of word relatedness (Turney et al., 2010). Many existing works are distributional models, based on the *Distributional Hypothesis*, that words occurring in similar contexts tend to have similar meanings (Harris, 1954). The limitation is that word vectors developed from distributional models cannot reveal word relatedness if its information does not lie in word distributions. For instance, they are believed to have difficulty distinguishing antonyms from synonyms, because the distribution of antonymous words are close, since the context of antonymous words are always similar to each other (Mohammad et al., 2013). Although some research claims that in certain conditions there do exist differences between the contexts of different antonymous words (Scheible et al., 2013), the differences are subtle enough that it can hardly be detected by such language models, especially for rare words.

Another important class of lexical resource for word relatedness is a lexicon, such as WordNet (Miller, 1995) or Roget's Thesaurus (Kipfer, 2009). Manually producing or extending lexicons is much more labor intensive than generating VSM word vectors using a corpus. Thus, lexicons are sparse with missing words and multi-word terms as well as missing relationships between words. Considering the synonym / antonym perspective as an example, WordNet answers less than 40% percent of the the GRE antonym questions provided by Mohammad et al. (2008) directly. Moreover, binary entries in lexicons do not indicate the degree of relatedness, such as the degree of lexical contrast between *happy* and *sad* or *happy* and *depressed*. The lack of such information makes it less fruitful when adopted in NLP applications.

In this work, we propose a Bayesian tensor factorization model (BPTF) for synthesizing a composite word vector representation by combining multiple different sources of word relatedness. The input is a set of word by word matrices, which may be sparse, providing a number indicating the presence or degree of relatedness. We treat word relatedness matrices from different perspectives as slices, forming a word relatedness tensor. Then the composite word vectors can be efficiently obtained by performing BPTF. Furthermore, given any two words and any trained relatedness perspective, we can create or recreate the pair-wise word relatedness with regularization via per-perspective linear transformation.

This method allows one set of word vectors to represent word relatednesses from many different perspectives (e.g. LSA for topic relatedness / corpus occurrences, ISA relation and YAGO type) It is able to bring the advantages from both word relatedness calculated by distributional models, and manually created lexicons, since the former have much more vocabulary coverage and many variations, while the latter covers word relatedness that is hard to detect by distributional models. We can use information from distributional perspectives to create (if does not exist) or re-create (with regularization) word relatedness from the lexicon's perspective.

We evaluate our model on distinguishing synonyms and antonyms. There are a number of related works (Lin and Zhao, 2003; Turney, 2008; Mohammad et al., 2008; Mohammad et al., 2013; Yih et al., 2012; Chang et al., 2013). A number of sophisticated methods have been applied, producing competitive results using diverse approaches. We use the GRE antonym questions (Mohammad et al., 2008) as a benchmark, and answer these questions by finding the most contrasting choice according to the created or recreated synonym / antonym word relatedness. The result achieves state-of-the-art performance.

The rest of this paper is organized as follows. Section 2 describes the related work of word vector representations, the BPTF model and antonymy detection. Section 3 presents our BPTF model and the sampling method. Section 4 shows the experimental evaluation and results with Section 5 providing conclusion and future work.

2 Related Work

2.1 Word Vector Representations

Vector space models of semantics have a long history as part of NLP technologies. One widely-used method is deriving word vectors using latent semantic analysis (LSA) (Deerwester et al., 1990), for measuring word similarities. This provides a topic based perspective on word similarity. In recent years, neural word embeddings have proved very effective in improving various NLP tasks (e.g. part-of-speech tagging, chunking, named entity recognition and semantic role labeling) (Collobert et al., 2011). The proposed neural models have a large number of variations, such as feed-forward networks (Bengio et al., 2003), hierarchical models (Mnih and Hinton, 2008), recurrent neural networks (Mikolov, 2012), and recursive neural networks (Socher et al., 2011). Mikolov et al. (2013) reported their vector-space word representation is able to reveal linguistic regularities and composite semantics using simple vector addition and subtraction. For example, "King$-$Man$+$Woman" results in a vector very close to "Queen". Luong et al. (2013) proposed a recursive neural networks model incorporating morphological structure, and has better performance for rare words.

Some non-VSM models[1] also generate word vector representations. Yih et al. (2012) apply polarity inducing latent semantic analysis (PILSA) to a thesaurus to derive the embedding of words. They treat each entry of a thesaurus as a document giving synonyms positive term counts, and antonyms negative term counts, and preform LSA on the signed TF-IDF matrix In this way, synonyms will have cosine similarities close to one and antonyms close to minus one.

Chang et al. (2013) further introduced Multi-Relational LSA (MRLSA), as as extension of LSA, that performs Tucker decomposition over a three-way tensor consisting of multiple relations (document-term like matrix) between words as slices, to capture lexical semantics. The purposes of MRLSA and our model are similar, but the different factorization techniques offer different advantages. In MRLSA, the k-th slice of tensor $\mathcal{W}$ is approximated by

$$\mathcal{W}_{:,:,k} \approx \mathcal{X}_{:,:,k} = \mathbf{U}\mathcal{S}_{:,:,k}\mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ are both for the same word list but are not guaranteed (or necessarily desired) to be the same. Thus, this model has the ability to capture asymmetric relations, but this flexibility is a detriment for symmetric relatedness. In order to expand word relatedness coverage, MRLSA needs to choose a pivot slice (e.g. the synonym slice), thus there always must existence such a slice, and the model performance depends on the quality of this pivot slice. Also, while non-completeness is a pervasive issue in manually created lexicons, MRLSA is not flexible enough to treat the unknown entries as missing. Instead it just sets them

[1]As defined by Turney et al. (2010), VSM must be derived from event frequencies.

to zero at the beginning and uses the pivot slice to re-calculate them. In contrast, our method of BPTF is well suited to symmetric relations with many unknown relatedness entries.

2.2 BPTF Model

Salakhutdinov and Mnih (2008) introduced a Bayesian Probabilistic Matrix Factorization (BPMF) model as a collaborative filtering algorithm. Xiong et al. (2010) proposed a Bayesian Probabilistic Tensor Factorization (BPTF) model which further extended the original model to incorporate temporal factors. They modeled latent feature vector for users and items, both can be trained efficiently using Markov chain Monte Carlo methods, and they obtained competitive results when applying their models on real-world recommendation data sets.

2.3 Antonomy Detection

There are a number of previous works in detecting antonymy. Lin and Zhao (2003) identifies antonyms by looking for pre-identified phrases in corpus datasets. Turney (2008) proposed a supervised classification method for handling analogies, then apply it to antonyms by transforming antonym pairs into analogy relations. Mohammad et al. (Mohammad et al., 2008; Mohammad et al., 2013) proposed empirical approaches considering corpus co-occurrence statistics and the structure of a published thesaurus. Based on the assumption that the strongly related words of two words in a contrasting pair are also often antonymous, they use affix patterns (e.g. "un-", "in-" and "im-") and a thesaurus as seed sets to add contrast links between word categories. Their best performance is achieved by further manually annotating contrasting adjacent categories. This approach relies on the *Contrast Hypothesis*, which will increase false positives even with a carefully designed methodology. Furthermore, while this approach can expand contrast relationships in a lexicon, out-of-vocabulary words still pose a substancial challenge.

Yih et al. (2012) and Chang et al. (2013) also applied their vectors on antonymy detection, and Yih et al. achieves the state-of-the-art performance in answering GRE antonym questions. In addition to the word vectors generated from PILSA, they use morphology and k-nearest neighbors from distributional word vector spaces to derive the embeddings for out-of-vocabulary words. The latter is problematic since both synonyms and antonyms are distributionally similar. Their approach is two stage: polarity inducing LSA from a manually created thesaurus, then falling back to morphology and distributional similarity when the lexicon lacks coverage. In contrast, we focus on fusing the information from thesauruses and automatically induced word relatedness measures during the word vector space creation. Then prediction is done in a single stage, from the latent vectors capturing all word relatedness perspectives and the appropriate per-perspective transformation vector.

3 Methods

3.1 The Bayesian Probabilistic Tensor Factorization Model

Our model is a variation of the BPMF model (Salakhutdinov and Mnih, 2008), and is similar to the temporal BPTF model (Xiong et al., 2010). To model word relatedness from multiple perspectives, we denote the relatedness between word i and word j from perspective k as R_{ij}^k. Then we can organize these similarities to form a three-way tensor $\mathbf{R} \in \mathbb{R}^{N \times N \times K}$.

Table 1 shows an example, the first slice of the tensor is a $N \times N$ matrix consists of 1/-1 corresponding to the synonym/antonym entries in the Roget's thesaurus, and the second slice is a $N \times N$ matrix consists of the cosine similarity from neural word embeddings created by Luong et al. (2013), where N is the number of words in the vocabulary. Note that in our model the entries missing in Table 1a do not necessarily need to be treated as zero. Here we use the indicator variable I_{ij}^k to denote if the entry R_{ij}^k exists ($I_{ij}^k = 1$) or not ($I_{ij}^k = 0$). If $K = 1$, the BPTF model becomes to BPMF. Hence the key difference between BPTF and BPMF is that the former combines multiple complementary word relatedness perspectives, while the later only smooths and generalizes over one.

We assume the relatedness R_{ij}^k to be Gaussian, and can be expressed as the inner-product of three D-dimensional latent vectors:

$$R_{ij}^k | V_i, V_j, P_k \sim \mathcal{N}(< V_i, V_j, P_k >, \alpha^{-1}),$$

where $< \cdot, \cdot, \cdot >$ is a generalization of dot product:

$$< V_i, V_j, P_k > \equiv \sum_{d=1}^{D} V_i^{(d)} V_j^{(d)} P_k^{(d)},$$

	happy	joyful	lucky	sad	depressed
happy		1	1	-1	-1
joyful	1			-1	
lucky	1			-1	
sad	-1	-1	-1		1
depressed	-1			1	

(a) The first slice: synonym & antonym relatedness

	happy	joyful	lucky	sad	depressed
happy		.03	.61	.65	.13
joyful	.03		.25	.18	.23
lucky	.61	.25		.56	.31
sad	.65	.18	.56		-.01
depressed	.13	.23	.31	-.01	

(b) The second slice: distributional similarity

Table 1: Word Relatedness Tensor

and α is the precision, the reciprocal of the variance. V_i and V_j are the latent vectors of word i and word j, and P_k is the latent vector for perspective k.

We follow a Bayesian approach, adding Gaussian priors to the variables:

$$V_i \sim \mathcal{N}(\mu_V, \Lambda_V^{-1}),$$

$$P_i \sim \mathcal{N}(\mu_P, \Lambda_P^{-1}),$$

where μ_V and μ_P are D dimensional vectors and Λ_V and Λ_P are D-by-D precision matrices.

Furthermore, we model the prior distribution of hyper-parameters as conjugate priors (following the model by (Xiong et al., 2010)):

$$p(\alpha) = \mathcal{W}(\alpha | \hat{W}_0, \nu_0),$$

$$p(\mu_V, \Lambda_V) = \mathcal{N}(\mu_V | \mu_0, (\beta_0 \Lambda_V)^{-1}) \mathcal{W}(\Lambda_V | W_0, \nu_0),$$

$$p(\mu_P, \Lambda_P) = \mathcal{N}(\mu_P | \mu_0, (\beta_0 \Lambda_P)^{-1}) \mathcal{W}(\Lambda_P | W_0, \nu_0),$$

where $\mathcal{W}(W_0, \nu_0)$ is the Wishart distribution of degree of freedom ν and a D-by-D scale matrix W, and $\hat{W}_0$ is a 1-by-1 scale matrix for α. The graphical model is shown in Figure 1 (with β_0 set to 1). After choosing the hyper-priors, the only remaining parameter to tune is the dimension of the latent vectors.

Due to the existence of prior distributions, our model can capture the correlation between different perspectives during the factorization stage, then create or re-create word relatedness using this correlation for regularization and generalization. This advantage is especially useful when such correlation is too subtle to be captured by other methods. On the other hand, if perspectives (let's say k and l) are actually unrelated, our model can handle it as well by making P_k and P_l orthogonal to each other.

3.2 Inference

To avoid calculating intractable distributions, we use a numerical method to approximate the results. Here we use the Gibbs sampling algorithm

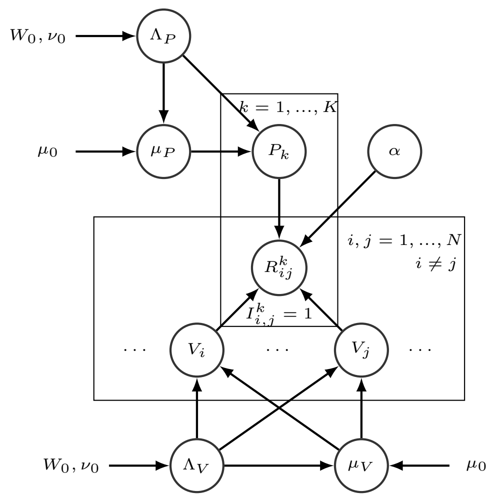

Figure 1: The graphical model for BPTF.

to perform the Markov chain Monte Carlo method. When sampling a block of parameters, all other parameters are fixed, and this procedure is repeated many times until convergence. The sampling algorithm is shown in Algorithm 1.

With conjugate priors, and assuming $I_{i,i}^k = 0, \forall i, k$ (we do not consider a word's relatedness to itself), the posterior distributions for each block of parameters are:

$$p(\alpha | \mathbf{R}, \mathbf{V}, \mathbf{P}) = \mathcal{W}(\hat{W_0}^*, \hat{\nu_0}^*) \qquad (1)$$

Where:

$$\hat{\nu}_0^* = \hat{\nu}_0 + \sum_{k=1}^{2} \sum_{i,j=1}^{N} I_{ij}^k,$$

$$(\hat{W}_0^*)^{-1} = \hat{W}_0^{-1} + \sum_{k=1}^{2} \sum_{i,j=1}^{N} I_{ij}^k (R_{ij}^k - < V_i, V_j, P_k >)^2$$

$$p(\mu_V, \Lambda_V | \mathbf{V}) = \mathcal{N}(\mu_V | \mu_0^*, (\beta_0^* \Lambda_V)^{-1}) \mathcal{W}(\Lambda_V | W_0^*, \nu_0^*) \quad (2)$$

Where:

$$\mu_0^* = \frac{\beta_0 \mu_0 + N\bar{V}}{\beta_0 + N}, \beta_0^* = \beta_0 + N, \nu_0^* = \nu_0 + N,$$
$$(W_0^*)^{-1} = W_0^{-1} + N\bar{S} + \frac{\beta_0 N}{\beta_0 + N}(\mu_0 - \bar{V})(\mu_0 - \bar{V})^T,$$
$$\bar{V} = \frac{1}{N}\sum_{i=1}^{N} V_i, \bar{S} = \frac{1}{N}\sum_{i=1}^{N}(V_i - \bar{V})(V_i - \bar{V})^T$$

$$p(\mu_P, \Lambda_P | \mathbf{P}) = \mathcal{N}(\mu_P | \mu_0^*, (\beta_0^* \Lambda_P)^{-1}) \mathcal{W}(\Lambda_P | W_0^*, \nu_0^*) \quad (3)$$

Which has the same form as $p(\mu_V, \Lambda_V | \mathbf{V})$.

$$p(V_i | \mathbf{R}, \mathbf{V}_{\neg i}, \mathbf{P}, \mu_V, \Lambda_V, \alpha) = \mathcal{N}(\mu_i^*, (\Lambda_i^*)^{-1}) \quad (4)$$

Where:

$$\mu_i^* = (\Lambda_i^*)^{-1}(\Lambda_V \mu_V + \alpha \sum_{k=1}^{2}\sum_{j=1}^{N} I_{ij}^k R_{ij}^k Q_{jk}),$$
$$\Lambda_i^* = \Lambda_V + \alpha \sum_{k=1}^{2}\sum_{j=1}^{N} I_{ij}^k Q_{jk} Q_{jk}^T,$$
$$Q_{jk} = V_j \odot P_k$$

$\odot$ is the element-wise product.

$$p(P_i | \mathbf{R}, \mathbf{V}, \mathbf{P}_{\neg i}, \mu_P, \Lambda_P, \alpha) = \mathcal{N}(\mu_i^*, (\Lambda_i^*)^{-1}) \quad (5)$$

Where:

$$\mu_k^* = (\Lambda_k^*)^{-1}(\Lambda_P \mu_P + \alpha \sum_{i,j=1}^{N} I_{ij}^k R_{ij}^k X_{ij}),$$
$$\Lambda_k^* = \Lambda_P + \alpha \sum_{i,j=1}^{N} I_{ij}^k X_{ij} X_{ij}^T,$$
$$X_{ij} = V_i \odot V_j$$

The influence each perspective k has on the latent word vectors is roughly propotional to the number of non-empty entries $n_k = \sum_{i,j} I_{i,j}^k$. If one wants to adjust the weight of each slices, this can easily achieved by adjusting (e.g. down sampling) the number of entries of each slice sampled at each iteration.

3.2.1 Out-of-Vocabulary words

It often occurs that some of the perspectives have greater word coverage than the others. For example, hand-labeled word relatedness usually has much less coverage than automatically acquired similarities. Of course, it is typically for the hand-labeled perspectives that the generalization is most desired. In this situation, our model can generalize word relatedness for the sparse perspective. For example, assume perspective k has larger vocabulary coverage N_k, while perspective l has a smaller coverage N_l.

There are two options for using the high vocabulary word relation matrix to generalize over the perspective with lower coverage. The most direct way simply considers the larger vocabulary in the BPTF $\mathbf{R} \in \mathbb{R}^{N_k \times N_k \times K}$ directly. A more efficient method trains on a tensor using the smaller vocabulary $\mathbf{R} \in \mathbb{R}^{N_l \times N_l \times K}$, then samples the $N_k - N_l$ word vectors using Equation 4.

Algorithm 1 Gibbs Sampling for BPTF

```
Initialize the parameters.
repeat
  Sample the hyper-parameters $\alpha$, $\mu_V$, $\Lambda_V$, $\mu_P$,
  $\Lambda_P$ (Equation 1, 2, 3)
  for $i = 1$ to $N$ do
    Sample $V_i$ (Equation 4)
  end for
  for $k = 1$ to $2$ do
    Sample $P_k$ (Equation 5)
  end for
until convergence
```

3.3 Predictions

With MCMC method, we can approximate the word relatedness distribution easily by averaging over a number of samples (instead of calculating intractable marginal distribution):

$$p(\hat{R}_{ij}^k | \mathbf{R}) \approx \frac{1}{M}\sum_{m=1}^{M} p(\hat{R}_{ij}^k | V_i^m, V_j^m, P_k^m, \alpha^m),$$

where m indicate parameters sampled from different sampling iterations.

3.4 Scalability

The time complexity of training our model is roughly $O(n \times D^2)$, where n is the number of observed entries in the tensor. If one is only interested in creating and re-creating word relatedness of one single slice rather than synthesizing word vectors, then entries in other slices can be downsampled at every iteration to reduce the training time. In our model, the vector length D is not sensitive and does not necessarily need to be very long. Xiong et al. (2010) reported in their collaborative filtering experiment $D = 10$ usually gives satisfactory performance.

4 Experimental Evaluation

In this section, we evaluate our model by answering antonym questions. This task is especially suitable for evaluating our model since the performance of straight-forward look-up from the thesauruses we considered is poor. There are two major limitations:

1. The thesaurus usually only contains antonym information for word pairs with a strong contrast.
2. The vocabulary of the antonym entries in the thesaurus is limited, and does not contain many words in the antonym questions.

On the other hand, distributional similarities can be trained from large corpora and hence have a large coverage for words. This implies that we can treat the thesaurus data as the first slice, and the distributional similarities as the second slice, then use our model to create / recreate word relatedness on the first slice to answer antonym questions.

4.1 The GRE Antonym Questions

There are several publicly available test datasets to measure the correctness of our word embeddings. In order to be able to compare with previous works, we follow the widely-used GRE test dataset provided by (Mohammad et al., 2008), which has a development set (consisting of 162 questions) and a test set (consisting of 950 questions). The GRE test is a good benchmark because the words are relatively rare (19% of the words in Mohammad's test are not in the top 50,000 most frequent words from Google Books (Goldberg and Orwant, 2013)), thus it is hard to lookup answers from a thesaurus directly with high recall. Below is an example of the GRE antonym question:

adulterate:	a. *renounce*	b. *forbid*
c. *purify*	d. *criticize*	e. *correct*

The goal is to choose the most opposite word from the target, here the correct answer is *purify*.

4.2 Data Resources

In our tensor model, the first slice ($k = 1$) consists of synonyms and antonyms from public thesauruses, and the second slice ($k = 2$) consists of cosine similarities from neural word embeddings (example in Table 1)

4.2.1 Thesaurus

Two popular thesauruses used in other research are *the Macquarie Thesaurus* and *the Encarta Thesaurus*. Unfortunately, their electronic versions are not publicly available. In this work we use two alternatives:

WordNet Words in WordNet (version 3.0) are grouped into sense-disambiguated synonym sets (synsets), and synsets have links between each other to express conceptual relations. Previous works reported very different look-up performance using WordNet (Mohammad et al., 2008; Yih et al., 2012), we consider this difference as different understanding of the WordNet structure. By extending "indirect antonyms" defined in WordNet to nouns, verbs and adverbs that similar words share the antonyms,we achieve a look-up performance close to Yih et al. (2012). Using this interpretation of WordNet synonym and antonym structure we obtain a thesaurus containing 54,239 single-token words. Antonym entries are present for 21,319 of them with 16.5 words per entry on average, and 52,750 of them have synonym entries with 11.7 words per entry on average.

Roget's Only considering single-token words, *the Roget's Thesaurus* (Kipfer, 2009) contains 47,282 words. Antonym entries are present for 8,802 of them with 4.2 words per entry on average, and 22,575 of them have synonym entries with 20.7 words per entry on average. Although the Roget's Thesaurus has a less coverage on both vocabulary and antonym pairs, it has better look-up precision in the GRE antonym questions.

4.2.2 Distributional Similarities

We use cosine similarity of the *morphRNN* word representations[2] provided by Luong et al. (2013) as a distributional word relatedness perspective. They used morphological structure in training recursive neural networks and the learned models outperform previous works on word similarity tasks, especially a task focused on rare words. The vector space models were initialized from existing word embeddings trained on Wikipedia. We use word embeddings adapted from Collobert et al. (2011). This advantage complements the weakness of the thesaurus perspective – that it has less coverage on rare words. The word vector data contains 138,218 words, and it covers 86.9% of the words in the GRE antonym questions. Combining the two perspectives, we can cover 99.8% of the

	Dev. Set			Test Set		
	Prec.	Rec.	F_1	Prec.	Rec.	F_1
WordNet lookup	0.40	0.40	0.40	0.42	0.41	0.42
WordNet PILSA	0.63	0.62	0.62	0.60	0.60	0.60
WordNet MRLSA	0.66	0.65	0.65	0.61	0.59	0.60
Encarta lookup	0.65	0.61	0.63	0.61	0.56	0.59
Encarta PILSA	0.86	0.81	0.84	0.81	0.74	0.77
Encarta MRLSA	0.87	0.82	0.84	**0.82**	0.74	0.78
Encarta PILSA + S2Net + Emebed	**0.88**	**0.87**	**0.87**	0.81	**0.80**	**0.81**
W&E MRLSA	0.88	0.85	0.87	0.81	0.77	0.79
WordNet lookup*	0.93	0.32	0.48	0.95	0.33	0.49
WordNet lookup	0.48	0.44	0.46	0.46	0.43	0.44
WordNet BPTF	0.63	0.63	0.63	0.63	0.62	0.62
Roget lookup*	1.00	0.35	0.52	0.99	0.31	0.47
Roget lookup	0.61	0.44	0.51	0.55	0.39	0.45
Roget BPTF	0.80	0.80	0.80	0.76	0.75	0.76
W&R lookup*	1.00	0.48	0.64	0.98	0.45	0.62
W&R lookup	0.62	0.54	0.58	0.59	0.51	0.55
W&R BPMF	0.59	0.59	0.59	0.52	0.52	0.52
W&R BPTF	**0.88**	**0.88**	**0.88**	**0.82**	**0.82**	**0.82**

Table 2: Development and test results on the GRE antonym questions. *Note: to allow comparison, in look-up we follow the approach used by (Yih et al., 2012): randomly guess an answer if the target word is in the vocabulary while none of the choices are. Asterisk indicates the look-up results without random guessing.

GRE antonym question words. Further using morphology information from WordNet, the coverage achieves 99.9%.

4.3 Tests

To answer the GRE questions, we calculate R^1_{ij} for word pair (i, j), where i is the target word and j is one of the question's candidates. The candidate with the smallest similarity is then the predicted answer. If a target word is missing in the vocabulary, that question will not be answered, while if a choice is missing, that choice will be ignored.

We first train on a tensor from a subset consisting of words with antonym entries, then add all other words using the out-of-vocabulary method described in Section 3. During each iteration, zeros are randomly added into the first slice to keep the model from overfitting. In the meantime, the second slice entries is randomly downsampled to match the number of non-empty entries in the first slice. This ensures each perspective has approximately equal influence on the latent word vectors.

We sample the parameters iteratively, and choose the burn-in period and vector length D according to the development set. We choose the vector length $D = 40$, the burn-in period starting from the 30th iterations, then averaging the relatedness over 200 runs. The hyper-priors used are $\mu_0 = 0$, $\nu_0 = \hat{\nu}_0 = D$, $\beta_0 = 1$ and $W_0 = \hat{W}_0 = \mathbf{I}$ (not tuned). Note that Yih et al. (2012) use a vector length of 300, which means our embeddings save considerable storage space and running time. Our model usually takes less than 30 minutes to meet the convergence criteria (on a machine with an Intel Xeon E3-1230V2 @ 3.3GHz CPU). In contrast, the MRLSA requires about 3 hours for tensor decomposition (Chang et al., 2013).

4.4 Results

The results are summarized in Table 2. We list the results of previous works (Yih et al., 2012; Chang et al., 2013) at the top of the table, where the best performance is achieved by PILSA on Encarta with further discriminative training and embedding. For comparison, we adopt the standard first used by (Mohammad et al., 2008), where *precision* is the number of questions answered correctly

[2]http://www-nlp.stanford.edu/ lmthang/morphoNLM/

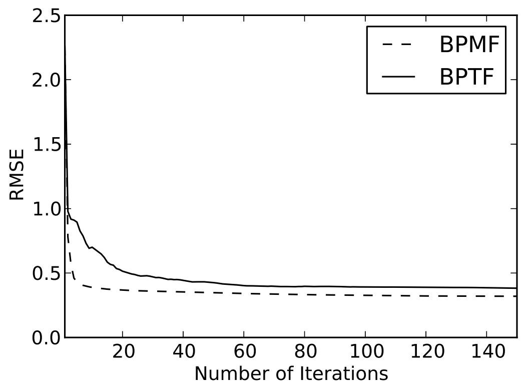

Figure 2: Convergence curves of BPMF and BPTF in training the W&R dataset. MAE is the mean absolute error over the synonym & antonym slice in the training tensor.

divided by the number of questions answered. *Recall* is the number of questions answered correctly divided by the total number of questions. BPMF (Bayesian Probabilistic Matrix Factorization) result is derived by only keeping the synonym & antonym slice in our BPTF model.

By using Roget's and WordNet together, our method increases the baseline look-up recall from 51% to 82% on the test set, while Yih's method increases the recall of Encarta from 56% to 80%. This state-of-the-art performance is achieved with the help of a neural network for fine tuning and multiple schemes of out-of-vocabulary embedding, while our method has inherent and straightforward "out-of-vocabulary embedding". While MRLSA, which has this character as well, only has a recall 77% when combining WordNet and Encarta together.

WordNet records less antonym relations for nouns, verbs and adverbs, while the GRE antonym questions has a large coverage of them. Although by extending these antonym relations using the "indirect antonym" concept achieves better look-up performance than Roget's, in contrast, the BPTF performance is actually much lower. This implies Roget's has better recording of antonym relations. Mohammad et al. (2008) reproted a 23% F-score look-up performance of WordNet which support this claim as well. Combining WordNet and Roget's together can improve the look-up performance further to 59% precision and 51% recall (still not as good as Encarta look-up).

Notably, if we strictly follow our BPTF approach but only use the synonym & antonym slice (i.e. a matrix factorization model instead of tensor factorization model), this single-slice model BPMF has performance that is only slightly better than look-up. Meanwhile Figure 1 shows the convergence curves of BPMF and BPTF. BPMF actually has lower MAE after convergence. Such behavior is caused by overfitting of BPMF on the training data. While known entries were recreated well, empty entries were not filled correctly. On the other hand, note that although our BPTF model has a higher MAE, it has much better performance in answering the GRE antonym questions. We interpret this as the regularization and generalization effect from other slice(s). Instead of focusing on one-slice training data, our model fills the missing entries with the help of inter-slice relations.

We also experimented with a linear metric learning method over the generated word vectors (to learn a metric matrix A to measure the word relatedness via $V_i^T A V_j$) using L-BFGS. By optimizing the mean square error on the synonym & antonym slice, we can reduce 8% of the mean square error on a held out test set, and improve the F-score by roughly 0.5% (of a single iteration). Although this method doesn't give a significant improvement, it is general and has the potential to boost the performance in other scenarios.

5 Conclusion

In this work, we propose a method to map words into a metric space automatically using thesaurus data, previous vector space models, or other word relatedness matrices as input, which is capable of handling out-of-vocabulary words of any particular perspective. This allows us to derive the relatedness of any given word pair and any perspective by the embedded word vectors with perperspective linear transformation. We evaluated the word embeddings with GRE antonym questions, and the result achieves the state-of-the-art performance.

For future works, we will extend the model and its applications in three main directions. First, in this model we only use a three-way tensor with two slices, while more relations may be able to add into it directly. Possible additional perspective slices include LSA for topic relatedness, and corpus occurrences in engineered or induced semantic patterns.

Second, we will apply the method to other tasks that require completing a word relatedness matrix. We evaluated the performance of our model on creating / recreating one perspective of word relatedness: antonymy. Perhaps using vectors generated from many kinds of perspectives would improve the performance on other NLP tasks, such as term matching employed by textual entailment and machine translation metrics.

Third, if our model does learn the relation between semantic similarities and distributional similarities, there may be fruitful information contained in the vectors V_i and P_k that can be explored. One straight-forward idea is that the dot product of perspective vectors $P_k \cdot P_l$ should be a measurement of correlation between perspectives.

Also, a straightforward adaptation of our model has the potential ability to capture asymmetric word relatedness as well, by using a per-perspective matrix instead of vector for the asymmetric slices (i.e. use $V_i^T A_k V_j$ instead of $\sum_{d=1}^{D} V_i^{(d)} P_k^{(d)} V_j^{(d)}$ for calculating word relatedness, where A_k is a square matrix).

Acknowledgments

We thank Christopher Kedzie for assisting the *Semantic Technologies in IBM Watson* seminar course in which this work has been carried out, and Kai-Wei Chang for giving detailed explanation of the evaluation method in his work.

References

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

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Scott C. Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 1, pages 241–247.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Barbara Ann Kipfer. 2009. *Roget's 21st Century Thesaurus, Third Edition*. Philip Lief Group.

Dekang Lin and Shaojun Zhao. 2003. Identifying synonyms among distributionally similar words. In *In Proceedings of IJCAI-03*, page 14921493.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, Sofia, Bulgaria.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751. The Association for Computational Linguistics.

Tom Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Ph. D. thesis, Brno University of Technology.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *NIPS*, pages 1081–1088.

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word-pair antonymy. In *EMNLP*, pages 982–991. Association for Computational Linguistics.

Saif Mohammad, Bonnie Dorr, Graeme Hirst, and Peter Turney. 2013. Computing lexical contrast. *Computational Linguistics*, 39(3):555–590.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887. ACM.

Silke Scheible, Sabine Schulte im Walde, and Sylvia Springorum. 2013. Uncovering distributional differences between synonyms and antonyms in a word space model. *International Joint Conference on Natural Language Processing*, pages 489–497.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136.

Peter D. Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Peter D. Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. *Coling*, pages 905–912, August.

Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222. SIAM.

Wen-tau Yih, Geoffrey Zweig, and John C. Platt. 2012. Polarity inducing latent semantic analysis. In *EMNLP-CoNLL*, pages 1212–1222. Association for Computational Linguistics.