A Probabilistic Model for Semantic Word Vectors

Andrew L. Maas and Andrew Y. Ng
Computer Science Department
Stanford University
Stanford, CA 94305
[amaas, ang]@cs.stanford.edu

Abstract

Vector representations of words capture relationships in words' functions and meanings. Many existing techniques for inducing such representations from data use a pipeline of hand-coded processing techniques. Neural language models offer principled techniques to learn word vectors using a probabilistic modeling approach. However, learning word vectors via language modeling produces representations with a *syntactic* focus, where word similarity is based upon how words are used in sentences. In this work we wish to learn word representations to encode word meaning – *semantics*. We introduce a model which learns semantically focused word vectors using a probabilistic model of documents. We evaluate the model's word vectors in two tasks of sentiment analysis.

1 Introduction

Word representations are a critical component of many natural language processing systems. Representing words as indices in a vocabulary fails to capture the rich structure of synonymy and antonymy among words. Vector representations encode continuous similarities between words as distance or angle between word vectors in a high-dimensional space. Word representation vectors have proved useful in tasks such as named entity recognition, part of speech tagging, and document retrieval [23, 6, 21].

Neural language models [2, 6, 14, 15] induce word vectors by back-propagating errors in a language modeling task through nonlinear neural networks, or linear transform matrices. Language modeling, predicting the next word in a sentence given a few preceding words, is primarily a syntactic task. Issues of *syntax* concern word function and the structural arrangement of words in a sentence, while issues of *semantics* concern word meaning. Learning word vectors using the syntactic task of language modeling produces representations which are syntactically focused. Word similarities with syntactic focus would pair "wonderful" with other highly polarized adjectives such as "terrible" or "awful." These similarities result from the fact that these words have similar syntactic properties – they are likely to occur at the same location in sentences like "the food was absolutely ____." In contrast, word representations capturing semantic similarity would associate "wonderful" with words of similar meaning such as "fantastic" and "prize-winner" because they have similar meaning despite possible differences in syntactic function. The construction of neural language models makes them unable to learn word representations which are primarily semantic.

Neural language models are instances of vector space models, which broadly refers to any method for inducing vector representations of words. Turney and Pantel [23] give a recent review of both syntactic and semantic vector space models. Most VSMs implement some combination of weighting, smoothing, and dimension reducing a word association matrix (e.g. TF-IDF weighting). For semantic or syntactic word representations VSMs use a term-document or word-context matrix respectively for word association. For each VSM processing stage there are dozens of possibilities, making the design space of VSMs overwhelming. Furthermore, many methods have little theo-

retical foundation and a particular weighting or dimension reduction technique is selected simply because it has been shown to work in practice. Neural language models offer a VSM for syntactic word vectors which has a complete probabilistic foundation. The present work offers a similarly well-founded probabilistic model which learns semantic, as opposed to syntactic, word vectors.

This work develops a model which learns semantically oriented word vectors using unsupervised learning. Word vectors are discovered from data as part of a probabilistic model of word occurrence in documents similar to a probabilistic topic model. Learning vectors from document-level word co-occurrence allows our model to learn word representations based on the topical information conveyed by words. Building a VSM with probabilistic foundation allows us to offer a principled solution to word vector learning in place of the hand-designed processing pipelines typically used. Our experiments show that our model learns vectors more suitable for document-level tasks when compared with other VSMs.

2 Log-Bilinear Language Model

Prior work introduced neural probabilistic language models [2], which predict the n^{th} word in a sequence given the $n-1$ preceding context words. More formally, a model defines a distribution $P(w_n|w_{1:n-1})$ where the number of context words is often small ($n \leq 6$). Neural language models encode this distribution using word vectors. Let ϕ_w be the vector representation of word w, a neural language model uses $P(w_n|w_{1:n-1}) = P(w_n|\phi_{1:n-1})$

Mnih and Hinton [14] introduce a neural language model which uses a log-bilinear energy function (lblLm). The model parametrizes the log probability of a word occurring in a given context using an inner product of the form

$$\log p(w_n|w_{1:n-1}) = \log p(w_n|\phi_{1:n-1}) \propto \left\langle \phi_n, \sum_{i=1}^{n-1} \phi_i^T C_i \right\rangle. \tag{1}$$

This is an inner product between the query word's representation ϕ_n and a sum of the context words' representations after each is transformed by a position specific matrix C_i. The ϕ vectors learned as part of the language modeling task are useful features for syntactic natural language processing tasks such as named-entity recognition and chunking [21]. As a VSM, the lblLm is a theoretically well-founded approach to learning syntactic word representations from word-context information.

The lblLm method does not provide a tractable solution for inducing word vectors from term-document data. The model introduces a transform matrix C_i for each context word, which causes the number of model parameters to grow linearly as the number of context words increases. For 100-dimensional word representation vectors, each C_i contains 10^4 parameters, which makes for an unreasonably large number of parameters when trying to learn representations from documents containing hundreds or thousands of words. Furthermore it is unclear how the model could handle documents of variable length, or if predicting a single word given all other words in the document is a good objective for training semantic word representations. Though the details of other neural language models differ, they face similar challenges in learning semantic word vectors because of their parametrization and language modeling objective.

3 Log-Bilinear Document Model

We now introduce a model which learns word representations from term-document information using principles similar to those used in the lblLm and other neural language models. However unlike previous work in the neural language model literature our model naturally handles term-document data to learn semantic word vectors. We derive a probabilistic model with log-bilinear energy function to model the bag of words distribution of a document. This approach naturally handles long, variable length documents, and learns representations sensitive to long-range word correlations. Maximum likelihood learning can then be efficiently performed with coordinate ascent optimization.

3.1 Model

Starting with the broad goal of matching the empirical distribution of words in a document, we model a document using a continuous mixture distribution over words indexed by a random variable θ. We assume words in a document are conditionally independent given the mixture variable θ. We assign a probability to a document d using a joint distribution over the document and a random variable θ. The model assumes each word $w_i \in d$ is conditionally independent of the other words given θ. The probability of a document is thus,

$$p(d) = \int p(d, \theta) d\theta = \int p(\theta) \prod_{i=1}^{N} p(w_i|\theta) d\theta. \tag{2}$$

Where N is the number of words in d and w_i is the i^{th} word in d. We use a Gaussian prior on θ.

We define the conditional distribution $p(w_i|\theta)$ using a log-linear model with parameters R and b. The energy function uses a word representation matrix $R \in \mathbb{R}^{(\beta \times |V|)}$ where each word w (represented as a one-hot vector) in the vocabulary V has a β-dimensional vector representation $\phi_w = Rw$ corresponding to that word's column in R. The random variable θ is also a β-dimensional vector, $\theta \in \mathbb{R}^{(\beta)}$ which weights each of the β dimensions of words' representation vectors. We additionally introduce a bias b_w for each word to capture differences in overall word frequencies. The energy assigned to a word w given these model parameters is,

$$E(w; \theta, \phi_w, b_w) = -\theta^T \phi_w - b_w. \tag{3}$$

To obtain the final distribution $p(w|\theta)$ we use a softmax,

$$p(w|\theta; R, b) = \frac{\exp(-E(w; \theta, \phi_w, b_w))}{\sum_{w' \in V} \exp(-E(w'; \theta, \phi_{w'}, b_{w'}))} = \frac{\exp(\theta^T \phi_w + b_w)}{\sum_{w' \in V} \exp(\theta^T \phi_{w'} + b_{w'})}. \tag{4}$$

The number of terms in the denominator's summation grows linearly in $|V|$, making exact computation of the distribution possible. For a given θ, a word w's occurrence probability is proportional to how closely its representation vector ϕ_w matches the scaling direction of θ This idea is similar to the word vector inner product used in the lblLm model.

Equation 2 resembles the probabilistic model of latent Dirichlet allocation (LDA) [3], which models documents as mixtures of latent topics. One could view the entries of a word vector ϕ as that word's association strength with respect to each latent topic dimension. The random variable θ then defines a weighting over topics. However, our model does not attempt to model individual topics, but instead directly models word probabilities conditioned on the topic weighting variable θ. Because of the log-linear formulation of the conditional distribution, θ is a vector in $\mathbb{R}^{\beta}$ and not restricted to the unit simplex as it is in LDA.

3.2 Learning

Given a document collection D, we assume documents are i.i.d samples and denote the k^{th} document as d_k. We wish to learn model parameters R and b to maximize,

$$\max_{R,b} p(D; R, b) = \prod_{d_k \in D} \int p(\theta) \prod_{i=1}^{N_k} p(w_i|\theta; R, b) d\theta. \tag{5}$$

Using MAP estimates for θ, we approximate this learning problem as,

$$\max_{R,b} \prod_{d_k \in D} p(\hat{\theta}_k) \prod_{i=1}^{N_k} p(w_i|\hat{\theta}_k; R, b), \tag{6}$$

where $\hat{\theta}_k$ denotes the MAP estimate of θ for d_k. We introduce a regularization term for the word representation matrix R. The word biases b are not regularized reflecting the fact that we want the biases to capture whatever overall word frequency statistics are present in the data. By taking the logarithm and simplifying we obtain the final learning problem,

$$\max_{R,b} \lambda ||R||_F^2 + \sum_{d_k \in D} \lambda ||\hat{\theta}_k||_2^2 + \sum_{i=1}^{N_k} \log p(w_i|\hat{\theta}_k; R, b). \tag{7}$$

The free parameters in the model are the regularization weight λ and the word vector dimensionality β. We use a single regularization weight λ for R and θ because the two are linearly linked in the conditional distribution $p(w|\theta; R, b)$.

The problem of finding optimal values for R and b requires optimization of the non-convex objective function. We use coordinate ascent, which first optimizes the word representations (R and b) while leaving the MAP estimates ($\hat{\theta}$) fixed. Then we find the new MAP estimate for each document while leaving the word representations fixed, and continue this process until convergence. The optimization algorithm quickly finds a global solution for each $\hat{\theta}_k$ because we have a low-dimensional, convex problem in each $\hat{\theta}_k$. Because the MAP estimation problems for different documents are independent, we can solve them on separate machines in parallel. This facilitates scaling the model to document collections with hundreds of thousands of documents.

4 Experiments

We evaluate our model with document-level and sentence-level categorization tasks in the domain of online movie reviews. These are sub-tasks of *sentiment analysis* which has recently received much attention as a challenging set of problems in natural language processing [4, 18, 22]. In both tasks we compare our model with several existing methods for word vector induction, and previously reported results from the literature. We also qualitatively evaluate the model's word representations by visualizing word similarities.

4.1 Word Representation Learning

We induce word representations with our model using 50,000 movie reviews from The Internet Movie Database (IMDB). Because some movies receive substantially more reviews than others, we limited ourselves to including at most 30 reviews from any movie in the collection. Previous work [5] shows function and negating words usually treated as stop words are in fact indicative of sentiment, so we build our dictionary by keeping the 20,000 most frequent unigram tokens without stop word removal. Additionally, because certain non-word tokens (e.g. "!" and ":-)") are indicative of sentiment, we allow them in our vocabulary.

As a qualitative assessment of word representations, we visualize the words most similar to a query word using vector similarity of the learned representations. Given a query word w and another word w' we obtain their vector representations ϕ_w and $\phi_{w'}$, and evaluate their cosine similarity as $Similarity(\phi_w, \phi_{w'}) = \frac{\phi_w^T \phi_{w'}}{||\phi_w|| \cdot ||\phi_{w'}||}$. By assessing the similarity of w with all other words w' in the vocabulary we can find the words deemed most similar by the model. Cosine similarity is often used with word vectors because it ignores differences in magnitude.

Table 1 shows the most similar words to given query words using our model's word representations. The vector similarities capture our intuitive notions of semantic similarity. The most similar words have a broad range of part-of-speech and functionality, but adhere to the theme suggested by the query word. Previous work on term-document VSMs demonstrated similar results, and compared the recovered word similarities to human concept organization [12, 20]. Table 1 also shows the most similar words to query words using word vectors trained via the lblLm on news articles (obtained already trained from [21]). Word similarities captured by the neural language model are primarily syntactic where part of speech similarity dominates semantic similarity. Word vectors obtained from LDA perform poorly on this task (not shown), presumably because LDA word/topic distributions do not meaningfully embed words in a vector space.

4.2 Other Word Representations

We implemented several alternative vector space models for comparison. With the exception of the lblLm, we induce word representations for each of the models using the same training data used to induce our own word representations.

Latent Semantic Analysis (LSA) [7]. One of the most commonly used tools in information retrieval, LSA applies the singular value decomposition (SVD) algorithm to factor a term-document

Table 1: Similarity of learned word vectors. The five words most similar to the target word (top row) using cosine similarity applied to the word vectors discovered by our model and the log-bilinear language model.

	romance	**mothers**	**murder**	**comedy**	**awful**	**amazing**
Our Model	romantic	lesbian	murdered	funny	terrible	absolutely
	love	mother	crime	laughs	horrible	fantastic
	chemistry	jewish	murders	hilarious	ridiculous	truly
	relationship	mom	committed	serious	bad	incredible
	drama	tolerance	murderer	few	stupid	extremely
LblLm	colours	parents	fraud	drama	unsettling	unbelievable
	paintings	families	kidnapping	monster	vice	incredible
	joy	veterans	rape	slogan	energetic	obvious
	diet	patients	corruption	guest	hires	perfect
	craftsmanship	adults	conspiracy	mentality	unbelievable	clear

co-occurrence matrix. To obtain a k-dimensional representation for a given word, only the entries corresponding to the k largest singular values are taken from the word's basis in the factored matrix.

Latent Dirichlet Allocation (LDA) [3]. LDA is a probabilistic model of documents which assumes each document is a mixture of latent topics. This model is often used to categorize or cluster documents by topic. For each latent topic, the model learns a conditional distribution $p(word|topic)$ for the probability a word occurs within the given topic. To obtain a k-dimensional vector representation of each word w, we use each $p(w|topic)$ value in the vector after training a k-topic model on the data. We normalize this vector to unit length because more frequent words often have high probability in many topics. To train the LDA model we use code released by the authors of [3]. When training LDA we remove from our vocabulary very frequent and very rare words.

Log-Bilinear Language Model (lblLm) [15]. This is the model given in [14] and discussed in section 2, but extended to reduce training time. We obtained the word representations from this model used in [21] which were trained on roughly 37 million words from a news corpus with a context window of size five.

4.3 Sentiment Classification

Our first evaluation task is document-level sentiment classification. A classifier must predict whether a given review is positive or negative (thumbs up vs. thumbs down) given only the text of the review. As a document-level categorization task, sentiment classification is substantially more difficult than topic-based categorization [22]. We chose this task because word vectors trained using term-document matrices are most commonly used in document-level tasks such as categorization and retrieval. The evaluation dataset is the polarity dataset version 2.0 introduced by Pang and Lee[1] [17]. This dataset consists of 2,000 movie reviews, where each is associated with a binary sentiment polarity label. We report 10-fold cross validation results using the authors' published folds to make our results comparable to those previously reported in the literature. We use a linear support vector machine classifier trained with LibLinear [8] and set the SVM regularization parameter to the same value used in [18, 17].

Because we are interested in evaluating the capabilities of various word representation learners, we use as features the *mean representation vector*, an average of the word representations for all words present in the document. The number of times a word appears in a document is often used as a feature when categorizing documents by topic. However, previous work found a binary indicator of whether or not the word is present to be a more useful feature in sentiment classification [22, 18]. For this reason we used term presence for our bag-of-words features. We also evaluate performance using mean representation vectors concatenated with the original bag-of-words vector. In all cases we normalize each feature vector to unit norm, and following the technique of [21] scale word representation matrices to have unit standard deviation.

[1] http://www.cs.cornell.edu/people/pabo/movie-review-data

Table 2: Sentiment classification results on the movie review dataset from [17]. Features labeled with "mean" are arithmetic means of the word vectors for words present in the review. Our model's representation outperforms other word vector methods, and is competitive with systems specially designed for sentiment classification.

Features	Accuracy (%)
Bag of Words (BoW)	86.75
LblLm Mean	71.30
LDA Mean	66.70
LSA Mean	77.45
Our Method Mean	**88.50**
LblLm Mean + BoW	86.10
LDA Mean + BoW	86.70
LSA Mean + BoW	85.25
Our Method Mean + BoW	**89.35**
BoW SVM reported in [17]	87.15
Contextual Valence Shifters [11]	86.20
δ TF-IDF Weighting [13]	88.10
Appraisal Taxonomy [25]	**90.20**

Table 2 shows the classification performance of our method, other VSMs we implemented, and previously reported results from the literature. Our method's features clearly outperform those of other VSMs. On its own, our method's word vectors outperform bag-of-words features with two orders of magnitude fewer features. When concatenated with the bag-of-words features, our method is competitive with previously reported results which use models engineered specifically for the task of sentiment classification. To our knowledge, the only method which outperforms our model's mean vectors concatenated with bag-of-words features is the work of Whitelaw *et al* [25]. This work builds a feature set of adjective phrases expressing sentiment using hand-selected words indicative of sentiment, WordNet, and online thesauri. That such a task-specific model narrowly outperforms our method is evidence for the power of unsupervised feature learning.

4.4 Subjectivity Detection

As a second evaluation task, we performed sentence-level subjectivity classification. In this task, a classifier is trained to decide whether a given sentence is subjective, expressing the writer's opinions, or objective, expressing purely facts. We used the dataset of Pang and Lee [17] which gathered subjective sentences from movie review summaries and objective sentences from movie plot summaries. This task is substantially different from the review classification task because it uses sentences as opposed to entire documents and the target concept is subjectivity instead of opinion polarity. We randomly split the 10,000 examples into 10 folds and report 10-fold cross validation accuracy using the SVM training protocol of [17].

Table 3 shows classification accuracies from the sentence subjectivity experiment. Our model provided superior features when compared against other VSMs, and slightly outperformed the bag-of-words baseline. Further improvement over the bag-of-words baseline is obtained by concatenating the two sets of features together.

5 Related Work

Prior work has developed several models to learn word representations via a probabilistic language modeling objective. Mnih and Hinton [14, 15] introduced an energy-based log-bilinear model for word representations following earlier work on neural language models [2, 16]. Successful application of these word representation learners and other neural network models include semantic role labeling, chunking, and named entity recognition [6, 21].

In contrast to the syntactic focus of language models, probabilistic topic models aim to capture document-level correlations among words [20]. Our probabilistic model is similar to LDA [3],

Table 3: Sentence subjective/objective classification accuracies using the movie review subjectivity dataset of [17]. Features labeled with "mean" are arithmetic means of the word vectors for words present in the sentence.

Features	Accuracy (%)
Bag of Words (BoW)	90.25
LblLm Mean	78.45
LDA Mean	66.65
LSA Mean	84.11
Our Method Mean	**90.36**
LblLm Mean + BoW	87.29
LDA Mean + BoW	88.82
LSA Mean + BoW	88.75
Our Method Mean + BoW	**91.54**
BoW SVM reported in [17]	90

which is related to pLSI [10]. However, pLSI doesn't give a well-defined probabilistic model over previously unseen novel documents. The recently introduced replicated softmax model [19] uses an undirected graphical model to learn topics in a document collection.

Turney and Pantel [23] offer an extensive review of VSMs which employ a matrix factorization technique after applying some weighting or smoothing operation to the matrix entries. Several recent techniques learn word representations in a principled manner as part of an application of interest. These applications include retrieval and ranking systems [1, 9], and systems to represent images and textual tags in the same vector space [24]. Our work learns word representations via the more basic task of topic modeling as compared to these more specialized representation learners.

6 Discussion

We presented a vector space model which learns semantically sensitive word representations via a probabilistic model of word occurrence in documents. Its probabilistic foundation gives a theoretically justified technique for word vector induction as an alternative to the overwhelming number of matrix factorization-based techniques commonly used. Our model is parametrized as a log-bilinear model following recent success in using similar techniques for language models [2, 6, 14, 15]. By assuming word order independence and replacing the language modeling objective with a document modeling objective, our model captures word relations at the document level.

Our model's foundation is closely related to probabilistic latent topic models [3, 20]. However, we parametrize our topic model in a manner which aims to capture word representations instead of latent topics. In our experiments, our method performed better than LDA which models latent topics directly.

We demonstrated the utility of our learned word vectors on two tasks of sentiment classification. Both were tasks of a semantic nature, and our methods' word vectors performed better than word vectors trained with the more syntactic objective of language modeling. Using the mean of word vectors to represent documents ignores vast amounts of information that could help categorization – negated phrases for example. Future work could better capture the information conveyed by words in sequence using convolutional models over word vectors.

Acknowledgments

We thank Chris Potts, Dan Ramage, Richard Socher, and Chris Manning for insightful discussions. This work is supported by the DARPA Deep Learning program under contract number FA8650-10-C-7020.

References

[1] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Supervised semantic indexing. In *Proceeding of CIKM*, 2009.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3(6):1137–1155, August 2003.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, May 2003.

[4] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the ACL*, 2007.

[5] C. K. Chung and J. W. Pennebaker. The psychological function of function words. *Social Communication*, pages 343–359, 2007.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing. *Proceedings of the 25th ICML*, 2008.

[7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

[8] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[9] D. Grangier, F. Monay, and S. Bengio. A discriminative approach for the retrieval of images from text queries. In *Proceedings of the ECML*, 2006.

[10] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of ACM SIGIR*, 1999.

[11] A. Kennedy and D. Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125, May 2006.

[12] T. Landauer, P. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2):259–284, 1998.

[13] J. Martineau and T. Finin. Delta tfidf: An improved feature space for sentiment analysis. In *Proceedings of the third AAAI internatonal conference on weblogs and social media*, 2009.

[14] A. Mnih and G. E. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th ICML*, 2007.

[15] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *Neural Information Processing Systems*, volume 22, 2009.

[16] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, 2005.

[17] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, volume 2004, 2004.

[18] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Empirical methods in natural language processing*, 2002.

[19] R. Salakhutdinov and G. E. Hinton. Replicated softmax: an undirected topic model. In *Advances in Neural Information Processing Systems*, volume 22, 2009.

[20] M. Steyvers and T. L. Griffiths. Probabilistic Topic Models. In *Latent Semantic Analysis: A Road to Meaning*, 2006.

[21] J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the ACL*, 2010.

[22] P. D. Turney. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the ACL*, 2002.

[23] P. D. Turney and P. Pantel. From Frequency to Meaning : Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

[24] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. In *Proceedings of the ECML*, 2010.

[25] C. Whitelaw, N. Garg, and S. Argamon. Using appraisal taxonomies for sentiment analysis. In *Proceedings of CIKM*, 2005.