Factor-based Compositional Embedding Models

Mo Yu
Machine Intelligence & Translation Lab
Harbin Institute of Technology
Harbin, China
gflfof@gmail.com

Matthew R. Gormley, Mark Dredze
Human Language Technology Center of Excellence
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD, 21218
{mgormley, mdredze}@cs.jhu.edu

Introduction Word embeddings, which are distributed word representations learned by neural language models [1, 2, 3], have been shown to be powerful word representations. They have been successfully applied to a range of NLP tasks, including syntax [2, 4, 5] and semantics [6, 7, 8].

Information about language structure is critical in many NLP tasks, where substructures of a sentence and its annotations inform downstream NLP task. Yet word representations alone do not capture such structure. For example, in relation extraction the sentence may be annotated with part-of-speech tags, a dependency parse, and named entities, with the goal of predicting a relation label for a pair of target entities. Semantic role labeling has a similar form. In tasks such as these, it is important to capture information about both individual words and their interactions. The annotations evidence these interactions, and we often define features over substructures of these annotated sentences (e.g. relative positions of words, words that appear on a dependency path, words with their entity types) to make successful predictions of the label (e.g. relation type).

Our goal is to learn representations for the substructures of an annotated sentence which inherit the generalization strength of word representations but remain sufficiently expressive for the task. Typically, for each term in a large finite vocabulary, we learn a unique *word* embedding. Yet, since the set of *annotated sentences* is infinite in size, it would be difficult to learn a unique representation of each one. Therefore, research has turned to *compositional* embedding models: building a representation (embedding) for an annotated sentence based on its component word embeddings.

A traditional approach for composition is to form a linear combination (e.g. sum) of single word representations with compositional operators either pre-defined [9, 10] or learned from data [11]. However, this approach ignores the useful structural information associated with the input (e.g. the order of words in a sentence and its syntactic tree). To address this problem, recent work has designed model structures that mimic the structure of the input. For example, Convolutional Neural Networks (CNNs) [2, 12] build the representation for a sentence based on its n-grams. Recursive Neural Networks (RNNs) [6, 7] and the Semantic Matching Energy Function [13] build the representations for an input tree (from either a syntactic parser or semantic role labeler), by composing the embedding for each node based on the embeddings of its children. Previous work on compositional phrase semantics [14, 15] can be seen as special cases of this type of models for phrases. Those models work well on sentence-level representations. However, the nature of their designs also limits them to fixed types of substructures from the annotated sentence, such as chains for CNNs and trees for RNNs. Such models cannot capture arbitrary combinations linguistic annotations available for a given task, such as word order, dependency tree, and named entities used for relation extraction.

In this paper we propose a powerful, efficient, and easy-to-implement compositional model. Our model capitalizes on arbitrary types of linguistic annotations by better utilizing features associated with substructures of those annotations, including global information (Table 1). We choose features to promote different properties and to distinguish different functions of the input words. The model achieves this goal with three steps. First, it decomposes the annotated sentence into substructures (i.e. factors). Second, it extracts features for each substructure, and combines these features with the embeddings of words in this substructure to form a **substructure embedding**. Third, these substructure embeddings are combined via a simple sum-pooling layer to form a **annotated sentence**

	Model Structure	Features influencing model structure	How Features Are Used
MVRNN [6]	Tree	Binary tree	- Gives tree model structure - Concatenated with phrase embedding
CNN [2, 12]	Linear-chain	Word-order, entity positions	- Concatenated with word embedding
Our models	MLP w/sparse connections	Arbitrary (e.g. word-order, dependency parse, entity positions)	- Promotes different properties of input words - Enforces sparsity of the hidden layer

Table 1: Comparison of Models

embedding. Finally, a softmax layer predicts the output label from this sentence-level embedding. We name this model the **Factor-based Compositional Embedding Model** (FCM). We test FCM on the relation classification task from SemEval 2010. By better handling the combined structures of the chains of words and syntactic trees, and by better utilizing the global information about the target entities, our FCM obtains state-of-the-art results on this task.

Log-linear Model Before turning to our full model, we first consider two special cases of it: one log-linear and one log-quadratic. Our log-linear model has the usual form, but defines a particular utilization of the features and embeddings. Instances have the form $(\mathbf{x}, y)$, where $\mathbf{x}$ is the input (e.g. sentence, dependency parse, named entities, etc.) and y is the output label (e.g. relation) (see Fig 1a for an example, where y indicates the relation between two target mentions M_1, M_2 in annotated sentence $\mathbf{x}$). The features of the log-linear model are defined for each word w_i in the sentence and divide into two parts: a binary vector of word features $\mathbf{g_i}$ and a dense word embedding e_{w_i}. We denote the label-specific model parameters by the matrix T_y. For example in Fig 1a, the gold label corresponds to a matrix T_y where *y=Product-Producer*(M_2, M_1). Our log-linear model is given by:

$$P(y|\mathbf{x}; T) \propto \exp(\textstyle\sum_i T_y \odot (\mathbf{g_i} \otimes e_{w_i})) \tag{1}$$

where $\otimes$ is the outer-product of the two vectors and $\odot$ is the 'matrix dot product' or Frobenious inner product of the two matrices. Note that, so long as the word embeddings e_{w_i} are constant, this model has the standard log-linear form. As usual, the binary features g_i may look at the ith word and any other substructure of the annotated sentence $\mathbf{x}$. The key idea is that because we take the outer product of the word-specific binary features with the word embedding, the model parameters are able to capture specific properties of the word (e.g. its position or named entity tag) while benefiting from the generalization properties of the embeddings.

Log-quadratic Model In our log-quadratic model, the probability $P(y|\mathbf{x}; T, \mathbf{e})$ is identical to our log-linear model in Eq. (1), except that we treat the word embeddings e_{w_i} as parameters. As in the deep learning literature, we initialize these embeddings from a neural language model [3] and then *fine-tune* them for our task. The probability is now log-quadratic in the parameters $\{T, \mathbf{e}\}$. Tensor $T = [T_1 : ... : T_{|L|}]$ transforms the input matrix to the labels [16, 17]; it has three dimensions, corresponding to word embeddings e_w, features associated with factor $\mathbf{g}_i$ and the output label set L.

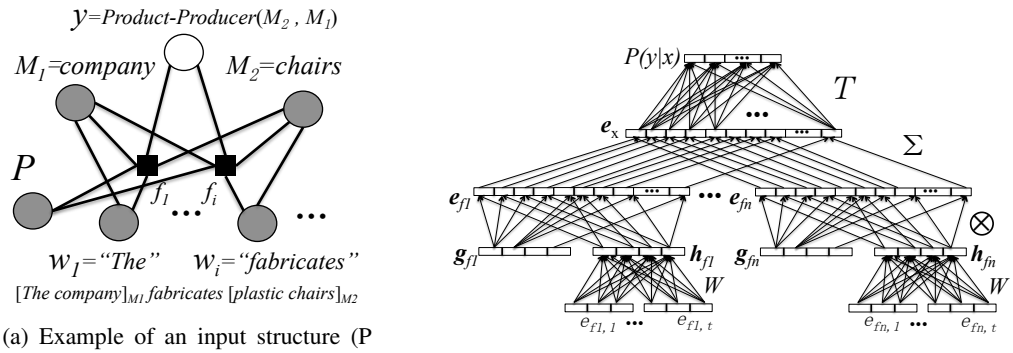
Generalized Model Form In this section we propose a new class of compositional models (FCM) which builds an embedding of a sentence and all of its linguistic annotations given an arbitrary decomposition of the annotated sentence into substructures or factors. In this way, we can reuse standard features and decompositions for a given task and avoid redesigning them from scratch.

Our full model is a slight extension of our log-quadratic model: we replace the single word embedding with a hidden layer which is itself a composition of word embeddings. The model decomposes the annotated sentence $\mathbf{x}$ into factors (i.e. substructures) following $\mathbf{x} = \{f\}$. For each factor f, there is a list of m associated features $\mathbf{g}_f$ and a list of t associated words $w_{f,1}, w_{f,2}, ..., w_{f,t} \in f$.[1] The words in a factor are transformed into a hidden layer $\mathbf{h}_f = \sigma(\sum_{j=1}^{t} e_{w_{f,j}} \cdot \mathbf{W}_j)$, where e_{w_i} is the word embedding for word w_i and $\sigma(\cdot)$ is a (possibly nonlinear) differentiable function, and $\mathbf{W}_j$ are parameters.[2] With these factors and the corresponding hidden layers, we construct our full model as below.

$$P(y|\mathbf{x}; T, \mathbf{W}, \mathbf{e}) = \exp(\textstyle\sum_f T_y \odot (\mathbf{g}_f \otimes \mathbf{h}_f)) \Big/ \sum_{y' \in L} \exp(\sum_f T_{y'} \odot (\mathbf{g}_f \otimes \mathbf{h}_f)) \tag{2}$$

[1] For notational convenience, each factor has the same number of words.

[2] Each $\mathbf{W}_j$ is a $d_e \times d_h$ matrix; d_e and d_h are the dimensions of the embeddings and hidden layers.



(a) Example of an input structure (P represents a dependency tree).

(b) Neural network representation for FCM.

Figure 1: Representation of the proposed model.

We obtain our log-quadratic and log-linear models as special cases by defining $\sigma(x) = x$ and $\mathbf{W}_j = I$ (identity matrix).

In order to better understand this model, we can consider the various compositional embeddings it constructs along the way. Further, this allows us to visualize our model as a multi-layer perceptron (MLP) (Fig. 1b). For each factor, we take the outer product between the feature vector and the hidden layer of the transformed embeddings $\mathbf{e}_{f_i} = \mathbf{g}_f \otimes \mathbf{h}_f$. We call $\mathbf{e}_{f_i}$ the **substructure embedding** for factor f_i. Next, we obtain the **annotated sentence embedding** $\mathbf{e}_x$ via a sum over the substructure embeddings, $\mathbf{e}_x = \sum_{f_i} \mathbf{e}_{f_i}$. Note that while the substructure and annotated sentence embeddings $\mathbf{e}_{f_i}$ and $\mathbf{e}_x$ are matrices, we consider their *vectorized* form in the visualization.

Learning Here we show how to train our full model.[3] In order to train the parameters we optimize the following log-likelihood objective with AdaGrad [18] and compute its gradients by backpropagation:

$$\mathcal{L}(T, \mathbf{W}, \mathbf{e}) = \frac{1}{|D|} \sum_{(y,\mathbf{x}) \in D} \log P(y|\mathbf{x}; T, \mathbf{W}, \mathbf{e}),$$

where D is the set of all training data. For each instance $(y, \mathbf{x})$ we compute the gradient of the log-likelihood $\ell = \log P(y|\mathbf{x}; T, \mathbf{W}, \mathbf{e})$. We define the vector $\mathbf{s} = [\sum_i T_y \odot (g_i \otimes e_{w_i})]_{1 \le y \le L}$, which yields $\partial \ell / \partial \mathbf{s} = [(I[y = y'] - P(y'|\mathbf{x}; T, \mathbf{W}, \mathbf{e}))_{1 \le y' \le L}]^T$, where $I[x]$ is the indicator function equal to 1 if x is true and 0 otherwise. Denote $\mathbf{a_f} = \sum_{j=1}^{t} e_{w_{f,j}} \cdot \mathbf{W}_j$. Then we have the following stochastic gradients, where $\sigma'(\cdot)$ is the gradient for any activation function $\sigma(\cdot)$ and $\circ$ is the tensor product:

$$\frac{\partial \ell}{\partial T} = \frac{\partial \ell}{\partial \mathbf{s}} \otimes \sum_{i=1}^{n} \mathbf{g}_{f_i} \otimes \mathbf{h}_{f_i}, \quad \frac{\partial \ell}{\partial \mathbf{W}_j} = \sum_{i=1}^{n} \frac{\partial \ell}{\partial \mathbf{h}_{f_i}} \frac{\partial \mathbf{h}_{f_i}}{\partial \mathbf{W}_j} = \sum_{i=1}^{n} \left(T \circ \mathbf{g}_{f_i} \circ \frac{\partial \ell}{\partial \mathbf{s}} \right) \cdot \sum_{j=1}^{t} \sigma'(\mathbf{a}_{f_i,j}) e_{w_j}^T.$$

We can fine-tune the word embeddings with FCM with the following equation:

$$\frac{\partial \ell}{\partial e_w} = \sum_{i=1}^{n} \frac{\partial \ell}{\partial \mathbf{h}_{f_i}} \sum_{j=1}^{t} I[w_j = w] \frac{\partial \mathbf{h}_{f_i}}{\partial e_w} = \sum_{i=1}^{n} \left(T \circ \mathbf{g}_{f_i} \circ \frac{\partial \ell}{\partial \mathbf{s}} \right) \cdot \sum_{j=1}^{t} I[w_j = w] \sigma'(\mathbf{a}_{f_i,j}) \mathbf{W}_j.$$

Experiments We conduct experiments on the SemEval-2010 Task 8 dataset[4][19]. We adopt the same setting as in [6]. This task is to determine the relation type (or no relation) between two entities in a sentence. We train 200-d word embeddings on the NYT portion of Gigaword5.0 corpus [20], with the default setting of the word2vec toolkit [3]. We annotate WordNet supertags and named entity (NE) tags using [21], and dependency parses using the Stanford Parser. We use 10-fold cross validation on the training data to select hyperparameters and do early-stopping. The learning rates for FCM with/without fine-tuning are 5e-3 and 5e-2 respectively.

We factorize the annotated sentence following Fig 1a. For each word w_i in the sentence, we set $\mathbf{h_{f_i}} = e_{w_i}$, equivalent to our log-linear and log-quadratic models. Our features $\mathbf{g_{f_i}}$ are over the word w_i, the two target entity mentions M_1, M_2, and their dependency path, as given in Table 2.

[3] The derivatives of the log-linear and log-quadratic models are special cases of those for the full model.

[4] SemEval-2010 website `http://docs.google.com/View?docid=dfvxd49s_36c28v9pmw`

Set	Template
HeadEmb	$\{I[i = h_1], I[i = h_2]\}$ (w_i is head of M_1/M_2) $\times\{\phi, t_{h_1}, t_{h_2}\}$
Context	$I[i = h_1 \pm 1]$ (left/right token of w_{h_1}), $I[i = h_2 \pm 1]$ (left/right token of w_{h_2})
In-between	$I[i > h_1]\&I[i < h_2]$ (in between) $\times\{\phi, t_{h_1}, t_{h_2}\}$
On-path	$I[w_i \in P]$ (on path) $\times\{\phi, t_{h_1}, t_{h_2}\}$

Table 2: Feature sets used in FCM.

Classifier	Features	F1
SVM [22] (Best in SemEval2010)	POS, prefixes, morphological, WordNet, dependency parse, Levin classes, PropBank, FrameNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
RNN	word embedding, syntactic parse	74.8
RNN + linear	word embedding, syntactic parse, POS, NER, WordNet	77.6
MVRNN	word embedding, syntactic parse	79.1
MVRNN + linear	word embedding, syntactic parse, POS, NER, WordNet	82.4
CNN [12]	word embedding, WordNet	82.7[5]
FCM (log-linear)	word embedding	77.6
	word embedding, dependency parse	79.4
	word embedding, dependency parse, WordNet	82.0
	word embedding, dependency parse, NER	81.4
FCM (log-quadratic)	word embedding	80.6
	word embedding, dependency parse	82.2
	word embedding, dependency parse, WordNet	82.5
	word embedding, dependency parse, NER	**83.0**

Table 3: Comparison of F1 for relation classification on SemEval-2010 Task 8.

Here h_1, h_2 are the indices of the two head words of M_1, M_2, $\times$ refers to the Cartesian product between two sets, t_{h_1} and t_{h_2} are WordNet supertags (or named entity tags) of the head words of two entities, and ϕ stands for empty feature. We discard the features related to t_{h_1}, t_{h_2} when there are no entity type features (WordNet/NER) available. The 'In-between' features indicate whether a word w_i is in between two target entities, and the 'On-path' features indicate whether the word is on the dependency path, on which there is a set of words P, between the two entities. We present the results of the log-linear and log-quadratic forms of our model as the additional hidden layer did not offer noticeable improvements.

Table 3 shows results for all methods. All FCMs, expect the log-linear one without any extra annotation features (77.6), achieve better performance compared to the previous compositional models (74.8/79.1 for RNN/MVRNN), showing that features indicating word positions (Table 2) can greatly help the task if they are properly utilized. Second, entity type features greatly improve the performance of the log-linear models. This is likely due to the fact that when embeddings are fixed, these features can help to distinguish different functions of embeddings. Third, in the fine-tuning setting, the embeddings themselves can be adapted to suit the target task, then introducing more entity type features makes it easier to over-fit. As a result, entity type features do not significantly improve fine-tuning performance. This also explains why using NE tags instead of WordNet tags help the log-linear model, while hurting the log-quadratic one; there are many more WordNet tags than NE tags. Finally, our best FCM obtains the best results (83.0) overall, setting a new high score for this task. It outperforms both the combinations of an embedding model and a traditional log-linear model in [6] (RNN/MVRNN + linear) and the result of CNN reported in [12]. Additionally, FCMruns much faster than both RNN and CNN models, because of its linear complexity on the dimensionality of embeddings.

Conclusion We have presented FCM, a new compositional model for deriving sentence-level and substructure embeddings from word embeddings. Compared to existing compositional models, FCM can easily handle arbitrary types of input and global information for composition, while being easy to implement. We have demonstrated that FCM attains state-of-the-art performance on the relation classification task. Our implementation is available for general use[6].

[5]We failed to reproduce the positive result in that paper and the performance of our implementation of CNN is 80.6. We checked with other researchers who also failed to re-implement this result. The problem is likely due to insufficient details in the paper for re-producing the effects of "position features.". Meanwhile the authors of the paper are unable to release their code.

[6]`https://github.com/Gorov/FCM_nips_workshop`

References

[1] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[2] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537, 2011.

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

[4] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Association for Computational Linguistics*, pages 384–394, 2010.

[5] Ronan Collobert. Deep learning for efficient discriminative parsing. In *AISTATS*, 2011.

[6] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP-CoNLL2012*, pages 1201–1211, 2012.

[7] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[8] Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. Semantic frame identification with distributed word representations. In *Proceedings of ACL*, June 2014.

[9] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, pages 236–244, 2008.

[10] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.

[11] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[12] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014*, pages 2335–2344, August 2014.

[13] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, pages 1–27, 2012.

[14] Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. Multi-step regression learning for compositional distributional semantics. *arXiv:1301.6939*, 2013.

[15] Georgiana Dinu and Marco Baroni. How to make words with vectors: Phrase generation in distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 624–633, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[16] Yuan Cao and Sanjeev Khudanpur. Online learning in tensor space. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 666–675, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[17] Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[18] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[19] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval-2 Workshop*, 2010.

[20] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword fifth edition, june. *Linguistic Data Consortium, LDC2011T07*, 2011.

[21] Massimiliano Ciaramita and Yasemin Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP2006*, pages 594–602, July 2006.

[22] Bryan Rink and Sanda Harabagiu. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259, Uppsala, Sweden, July 2010. Association for Computational Linguistics.