Supplemental document for *'Learning distributed event representations with a multi-task approach'*

Xudong Hong†, Asad Sayeed*, Vera Demberg†
†Dept. of Language Science and Technology, Saarland University
{xhong,vera}@coli.uni-saarland.de
*Dept. of Philosophy, Linguistics, and Theory of Science, University of Gothenburg
asad.sayeed@gu.se

1 Tensor Factorisation

There are $|V| \times |R| \times d$ parameters in the embedding tensor $\mathbf{T}$. Since word vocabulary size $|V|$ is often very big, the number of parameters grows tremendously when $|R|$ or d increases, which makes it hard for the model to converge. The huge number of parameters in event-participant embeddings can result in the curse of dimensionality. Moreover, this embedding tensor lacks parameter sharing across word embedding matrices between different roles, while words with the same semantic role actually have common characteristics (Tilk et al., 2016). In order to capture these characteristics, we need a method to share parameters between word embedding matrices.

We apply *tensor rank decomposition* (Hitchcock, 1927) which factorised the embedding tensor in real space:

$$\begin{aligned}\mathbf{T} &= \sum_{m=1}^{k} \lambda_m \mathbf{T}'_m \\ &= \sum_{m=1}^{k} \lambda_m \mathbf{a}_m \otimes \mathbf{b}_m \otimes \mathbf{d}_m,\end{aligned} \tag{1}$$

where tensor $\mathbf{T}$ is decomposed into k rank-1 tensors $\mathbf{T}'_m$ and λ_m is the weight of rank m. Each *rank-1 tensor* can be written as outer product of 3 row vectors $\mathbf{a}_m$, $\mathbf{b}_m$ and $\mathbf{d}_m$. Since all parameters are learned during training, we can let $\mathbf{c}_m = \lambda_m \mathbf{d}_m$. So the equation becomes:

$$\mathbf{T} = \sum_{m=1}^{k} \mathbf{a}_m \otimes \mathbf{b}_m \otimes \mathbf{c}_m, \tag{2}$$

The smallest k that makes this equation true is the *tensor rank* of tensor $\mathbf{T}$. Now the number of parameters is reduced to $(|V| + |R| + d) \times k$ which is comparatively smaller.

This factored tensor structure is very flexible. The event-participant embedding vector $\mathbf{T}_{(ij)}$ can be computed as:

$$\begin{aligned}\mathbf{T}_{(ij)} &= \sum_{m=1}^{k} \mathbf{a}_{m_{(i)}} \otimes \mathbf{b}_{m_{(j)}} \otimes \mathbf{c}_m \\ &= \sum_{m=1}^{k} \mathbf{a}_{m_{(i)}} \mathbf{b}_{m_{(j)}} \mathbf{c}_m\end{aligned} \tag{3}$$

Also, a word i can be represented as a matrix $\mathbf{T}_{(i)} \in \mathbb{R}^{|R| \times d}$:

$$\mathbf{T}_{(i)} = \sum_{m=1}^{k} \mathbf{a}_{m_{(i)}} \otimes \mathbf{b}_m \otimes \mathbf{c}_m \tag{4}$$

where the j-th row of $\mathbf{T}_{(i)}$ is the event-participant embedding vector $\mathbf{T}_{(ij)}$. Similarly, a role j can be represented as a matrix $\mathbf{T}_{(j)} \in \mathbb{R}^{|V| \times d}$:

$$\mathbf{T}_{(j)} = \sum_{m=1}^{k} \mathbf{a}_m \otimes \mathbf{b}_{m_{(j)}} \otimes \mathbf{c}_m \tag{5}$$

1.1 Interpretation of Factored Tensor

To make the factored tensor easier to interpret and implement, we group k vectors $\mathbf{a}_m$, $\mathbf{b}_m$ by column into matrices $\mathbf{A} \in \mathbb{R}^{|V| \times k}$, $\mathbf{B} \in \mathbb{R}^{|R| \times k}$ and $\mathbf{c}_m$ by row into matrix $\mathbf{C} \in \mathbb{R}^{k \times d}$ to get a matrix form of the factored tensor. Now we can get rid of the outer products.

The vector $\mathbf{T}_{(ij)}$ can be computed as:

$$\mathbf{T}_{(ij)} = (\mathbf{A}_{(i)} \circ \mathbf{B}_{(j)})\mathbf{C} \tag{6}$$

which is a representation of a participant in an event. "$\circ$" is Hadamard product which computes element-wise multiplication between two vectors. This can be interpreted as a multiplicative composition method of word embedding and role embedding. After adding the transformation from symbolic representation to distributed representation, the vector $\mathbf{T}_{(ij)}$ is written as:

$$\mathbf{p} = \mathbf{T}_{(ij)} = (\mathbf{w}_i\mathbf{A} \circ \mathbf{r}_j\mathbf{B})\mathbf{C} \tag{7}$$

where $\mathbf{w}_i$ and $\mathbf{r}_j$ are the one-hot vectors correspondingly for word i and role j.

With the matrices defined above, word embedding $\mathbf{T}_{(i)}$ can be computed as:

$$\mathbf{T}_{(i)} = \mathbf{B}\ diag(\mathbf{A}_{(i)})\ \mathbf{C} \tag{8}$$

where $diag(\mathbf{A}_{(i)})$ is a diagonal matrix with vector $\mathbf{A}_{(i)}$ on its diagonal. Although there is an intuitive mapping between matrix $\mathbf{A}$ and word embedding matrix, they are not equivalent to each other.

To discover this mapping in a simplified condition, we assume that in the word representation matrix $\mathbf{T}_{(i)}$ of any word i, each role-specific representation contributes to the word representation with a weight of w_j which is word-irrelevant. Then the matrix $\mathbf{T}_{(i)}$ can be compressed into a vector as the weighted average of its rows:

$$\mathbf{v}_i = \frac{1}{|R|}\sum_{j\in R} w_j\mathbf{T}_{(i)(j)} \tag{9}$$

For vector $\mathbf{A}_{(i)} \in \mathbb{R}^k$, computing $\mathbf{v}_i = \mathbf{A}_{(i)}\mathbf{M}$ where $\mathbf{M} \in \mathbb{R}^{k\times d}$ can be considered as a linear mapping from a vector space of $\mathbb{R}^k$ to a vector space of $\mathbb{R}^d$. Now we can prove that the word embedding $\mathbf{v}_i$ is the result of the word embedding $\mathbf{A}_{(i)}$ after performing a linear mapping:

$$\begin{aligned}\mathbf{v}_i &= \frac{1}{|R|}\sum_{j\in R} w_j(\mathbf{B}\ diag(\mathbf{A}_{(i)})\ \mathbf{C})_{(j)}\\ &= (\frac{1}{|R|}\sum_{j\in R} w_j\mathbf{B}_{(j)})\ diag(\mathbf{A}_{(i)})\ \mathbf{C}\\ &= \mathbf{A}_{(i)}\ diag(\frac{1}{|R|}\sum_{j\in R} w_j\mathbf{B}_{(j)})\ \mathbf{C}\\ &= \mathbf{A}_{(i)}\mathbf{M}\end{aligned} \tag{10}$$

where the matrix $\mathbf{M}$ is word-irrelevant. Naturally, the matrix $\mathbf{A}$ can be interpreted as embedding matrix of all words in a vector space of $\mathbb{R}^k$.

A similar interpretation can also be applied to semantic role embedding. Semantic role j can be represented by a matrix:

$$\mathbf{T}_{(j)} = \mathbf{A}\ diag(\mathbf{B}_{(j)})\ \mathbf{C} \tag{11}$$

$\mathbf{B}$ can be considered as an embedding matrix for all semantic roles.

2 Multi-task Learning Model

For each target role or target word, we apply tensor factorisation again to obtain target role/word specific classifiers. At first we define weight matrix $\mathbf{W}_w$ and weight matrix $\mathbf{W}_r$ for each target word a_x and target role b_y. Then we stack $\mathbf{W}_w$ and $\mathbf{W}_r$ from a_1 to $a_{|V|}$ and from b_1 to $b_{|R|}$. Two weight tensors $\mathbf{T}^{(w)} \in \mathbb{R}^{d\times|R|\times|V|}$ and $\mathbf{T}^{(r)} \in \mathbb{R}^{d\times|V|\times|R|}$ are obtained. But these tensors suffer the same dimensionality problem as the event-participant embedding tensor. So we apply tensor rank decomposition again on two weight tensors and extract the corresponding weight matrices for a_x and b_y as follows:

$$\begin{aligned}\mathbf{T}^{(w)}_{(x)} &= \sum_{m=1}^{k^{(w)}} \mathbf{c}^{(w)}_m \otimes \mathbf{b}^{(w)}_{m_{(x)}} \otimes \mathbf{a}^{(w)}_m\\ \mathbf{T}^{(r)}_{(y)} &= \sum_{m=1}^{k^{(r)}} \mathbf{c}^{(r)}_m \otimes \mathbf{a}^{(r)}_{m_{(y)}} \otimes \mathbf{b}^{(r)}_m\end{aligned} \tag{12}$$

Similarly, we group vectors $\mathbf{c}^{(w)}_m$, $\mathbf{c}^{(r)}_m$ by row into matrices $\mathbf{C}_w \in \mathbb{R}^{d\times k^{(w)}}$, $\mathbf{C}_r \in \mathbb{R}^{d\times k^{(r)}}$ and group $\mathbf{a}^{(w)}_m$, $\mathbf{b}^{(r)}_m$ by column into matrices $\mathbf{A}_w \in \mathbb{R}^{k^{(w)}\times|V|}$, $\mathbf{B}_r \in \mathbb{R}^{k^{(r)}\times|R|}$. After getting rid of the outer product using the same transformation as equation (6), it can be written as:

$$\begin{aligned}\mathbf{W}_w &= \mathbf{C}_w\ diag(\mathbf{b}_y)\ \mathbf{A}_w\\ \mathbf{W}_r &= \mathbf{C}_r\ diag(\mathbf{a}_x)\ \mathbf{B}_r\end{aligned} \tag{13}$$

where $diag(\mathbf{v})$ is a diagonal matrix with vector $\mathbf{v}$ on its diagonal. Since there is a vector $\mathbf{b}_y$ representing target role b_y, and there is a vector $\mathbf{a}_x$ representing target word a_x, we can now define the embedding matrices $\mathbf{B}_w \in \mathbb{R}^{k^{(w)}\times|R|}$, $\mathbf{A}_r \in \mathbb{R}^{k^{(r)}\times|V|}$ for target roles and target words. The weight matrices can be further written as:

$$\begin{aligned}\mathbf{W}_w &= \mathbf{C}_w\ diag(\mathbf{r}_y\mathbf{B}_w)\ \mathbf{A}_w\\ \mathbf{W}_r &= \mathbf{C}_r\ diag(\mathbf{w}_x\mathbf{A}_r)\ \mathbf{B}_r\end{aligned} \tag{14}$$

3 Residual Learning

All models presented above contain two factored tensors. It has been shown that training models with factored tensors is difficult (Sutskever et al., 2011; Kiros et al., 2014). Due to the fact that each tensor element is represented as a product of

ID	Role Label	Definition
0	PRD	The predicate denoting the event
1	ARG0	Proto-agent: the event participant that most likely to be the agent, volitional causer or experiencer
2	ARG1	Proto-patient: the event participant that most likely to be the patient which is being affected by the action
3	ARGM-MNR	The event participant that specifies how the action is performed
4	ARGM-LOC	The event participant that indicates where the action takes place
5	ARGM-TMP	The event participant that shows when an action took place

Table 1: Definitions of semantic role labels in this paper.

three parameters, the gradient explosion or vanishing problem is more likely to happen during training. To tackle the gradient explosion or vanishing problem in deep neural networks, He et al. (2016) proposed the *deep residual learning* framework. They made the observation that when constructing a deep model from a shallow model, one needs to add additional layers which are identity mapping over other layers copied from the shallow model. As a result, we can force the layers to fit a residual mapping. Formally, a residual block containing two hidden layers using $PReLU$ activation function can be written as:

$$\begin{aligned}\mathbf{h}^{(1)} &= PReLU(\mathbf{x}\mathbf{W}^{(1)}) \\ \mathbf{h}^{(2)} &= PReLU(\mathbf{h}^{(1)}\mathbf{W}^{(2)} + \mathbf{x})\end{aligned} \tag{15}$$

4 Corpus and Preprocessing

4.1 Corpus

We use the Rollenwechsel-English (**RW-eng**) corpus, a large-scale semantic corpus using PropBank semantic role labels (Sayeed et al., 2015). Since the head words of arguments are extracted, we have a direct access to the event participants without modifiers. This helps us to learn event-participant embeddings from generalised event knowledge.

The corpus, extracted from the ukWaC corpus and BNC, and contains about 78 million sentences over 2.3 million documents. Among those sentences, there are approximately 210 million event predicates and 710 million event participants. They divide the corpus into 3500 segments in XML format with approximately same number of documents. The sentences are labelled using SENNA which does not rely on syntactic features but has competitive performance. If sentences have multiple predicates, they are separated into several events. The head nouns are extracted with one of four heuristics: MALT, MALT-SPAN, LINEAR or FAILED.

4.2 Preprocessing

We choose the first 3472 segments as training section, the next 14 segments as test section and the next 14 segments as validation section. The word list for event predicates and participants is constructed with following steps:

1. go through the training section; for each event, find the entries denoting the event predicate and all event participants.
2. filter out every entry that has FAILED in $algorithm$; only accept entries with POS tags starting with n, v, j, r; only accept entries with English characters and the dashed line $-$.
3. extract the lower case form of the head word from each entry.
4. lemmatise each word with WordNet (Miller, 1995) lemmatiser in NLTK (Bird, 2006).
5. compute the frequency of each unique lemma.
6. generate the vocabulary from the top 50000 most frequent words.

The identifier of the out-of-vocabulary word <OOV> is 50001. Using this word list, each word a_i is assigned an identifier i in the integer set of $[1, 50001]$.

	Description	Value
k	Embeddings dimensions	256
d	Hidden layer dimensions	512
$k^{(w)}$	Word output dimensions	512
$k^{(r)}$	Role output dimensions	512
α	Role prediction weight	1.0
bs	Batch size	4000
lr	Learning rate	0.1
is	Iteration size	1×10^8
dl	Iterations to decrease lr	3
dr	Decreasing factor	0.1
es	Early stopping iterations	5

Table 2: Hyper-parameters for model training.

The role vocabulary is constructed upon the list of semantic role labels defined in Table 1 with an additional label `<OTHER>` for those semantic roles in the corpus but not in our list. We choose these roles because the head search heuristics are well-defined on these semantic roles. Since we have 6 role labels in the role vocabulary R, each role b_j has a identifier j in the range of $[0, 6]$, where the identifier for the `<OTHER>` label is 6.

Using these identifiers, we go through all three sections to construct training data, test data and validation data. Similar to vocabulary construction, we filter out all events with `FAILED` in $algorithm$. In each event, the maximum number of roles is 7. If there is more than one `<OTHER>` role in the event, we accept the last one for simplicity. For each role represented by its identifier, it is coupled with a word identifier for the corresponding head word. For any role without a head word, an identifier 0 for empty word `<NULL>` is assigned.

5 Model Implementation and Hyper-parameters

We use the same model hyper-parameter setting as Tilk et al. (2016) in order to make a fair comparison. The set of hyper-parameters for all models is listed in Table 2. We choose 1.0 as the weight of semantic role prediction α to balance between two tasks. We also experimented with alpha set to 0.1/0.01: smaller alpha will help role filling and thematic fit but harm the performance of role classification. Because the data scale is enormous, in order to speed-up the training process, we make use of all 12 GB video memory and set the batch

Model	$mean \pm std$
NNRF	0.66 ± 0.12
NNRF-MT	0.63 ± 0.13

Table 3: Cosine similarity between target events and foil events.

size bs to 4000. Using this batch size, we perform feature selection on learning rate. We train NNRF-MT on the first 10% of training set for 5 iterations and observe the perplexity on test data. We compare the values in the set of $\{0.05, 0.1, 0.15\}$. The setting of 0.05 has the worst performance, and the setting of 0.15 causes gradient explosion from time to time. The results show that 0.1 is the optimised setting. We set the iteration size is to 1×10^8. Because it is approximately $\frac{1}{5}$ of the training data size. With this setting, we can now consider 5 iterations as one epoch. We set the early stopping iterations es to 5, set the decrease learning rate iterations dl to 3 (that is, about half of es), and set decreasing factor dr to 0.1, according to the suggestion of He et al. (2016). We define the minimum difference between two values ϵ as $1.0 * 10^{-3}$. So if the difference between the loss of current iteration and the loss of best iteration is no more than $1.0 * 10^{-3}$, we consider the loss to not be decreasing. From our trial experiments, the role classification task is comparatively easier than the role-filler prediction task, which is usually overfitted around 21 epochs. As a result, we will perform early stopping and reduce the number of training iterations to $27 = 21 + es + 1$.

We implement all models using Keras 2.0.5 (Chollet et al., 2015) with Tensorflow 1.2 (Abadi et al., 2016) as backend. And then we deploy each model as a Docker container on a high performance computer with one 48-core Intel Xeon E7 CPU and 256 GB main memory. We use one Nvidia Geforce Titan X GPU with 12 GB video memory as neural network accelerator. We have made the source code publicly available on Github[1].

6 Additional Evaluations

6.1 Sensitivity to Semantic Roles

We begin by testing whether the multi-task model lives up to our initial intention for a better encoding of the semantic roles. We test this by con-

[1] `https://github.com/tony-hong/event-embedding-multitask`

Role	Precision	Recall	F_1	freq
`PRD`	99.5	100	99.7	610K
`ARG0`	94.1	86.8	90.3	205K
`ARG1`	90.6	96.5	93.5	453K
`AM-MNR`	84.4	73.9	78.8	42K
`AM-LOC`	78.3	64.7	70.9	37K
`AM-TMP`	87.3	82.0	84.6	67K
`OTHER`	99.9	99.7	99.8	161K
overall	94.9	94.9	94.8	1575K

Table 4: Results of RoFA-MT on the major semantic roles. *freq* is the frequency of each target role.

structing a dataset of 500 correct events and 500 foil events for which we switched the head words of `ARG0` and `ARG1`, e.g.:

correct: (*boy*/`ARG0`, *eat*/`PRD`, *apple*/`ARG1`)
foil: (*apple*/`ARG0`, *eat*/`PRD`, *boy*/`ARG1`)

We then test whether the multi-task model is better at distinguishing foil events from real events than the original NNRF model. In order to assess the models, we compute the cosine similarity between the embeddings for correct events vs. foil events. The results are shown in Table 3. A low similarity means that the model is good at distinguishing between correct and foil events. A two-tailed paired t-test shows that the event similarities are significantly lower in the multi-task model compared to the NNRF model ($t = -10.64$, $p < 0.0001$).

6.2 Semantic Role Classification: Details

Table 4 breaks up the semantic role classification results by role for the RoFA-MT model. The recall of the location role `ARGM-LOC` is rather low. Our analysis of the confusion matrix shows that head words with a location role were often misclassified to other roles like `ARG0` or `ARG1`.

6.3 Result With Standard Deviations

Table 5 and 6 show results with standard deviations which we use for significance testing.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*. volume 16, pages 265–283.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics, pages 69–72.

François Chollet et al. 2015. Keras.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 770–778.

Frank L Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics* 6(1-4):164–189.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 595–603.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Asad Sayeed, Stefan Fischer, and Vera Demberg. 2015. Vector-space calculation of semantic surprisal for predicting word pronunciation duration. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 763–773. http://www.aclweb.org/anthology/P15-1074.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 1017–1024.

Ottokar Tilk, Vera Demberg, Asad Sayeed, Dietrich Klakow, and Stefan Thater. 2016. Event participant modelling with neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 171–182. https://aclweb.org/anthology/D16-1017.

Model	Pado07	McRae05	F-Loc	F-Inst	GDS	avg
NNRF	43.3 ± 1.2	35.9 ± 0.5	46.5 ± 0.9	52.1 ± 2.7	57.6 ± 0.8	44.2 ± 0.7
NNRF-MT	43.2 ± 1.6	36.1 ± 0.7	46.3 ± 1.4	50.0 ± 1.6	57.2 ± 0.5	44.0 ± 0.2
RoFA-MT	52.2 ± 1.4	41.9 ± 0.6	45.9 ± 1.5	49.4 ± 2.0	60.7 ± 0.6	48.5 ± 0.5
ResRoFA-MT	53.0 ± 1.1	42.5 ± 0.8	46.3 ± 1.4	47.7 ± 2.6	60.8 ± 0.9	48.9 ± 0.5

Table 5: Results with standard deviations on human thematic fit judgement correlation task (Spearman's $\rho \times 100$) compared to previous work. The last column reports the weighted averages by numbers of entries of all five datasets.

Model	Bicknell	GS2013
NNRF	73.0 ± 2.2	34.2 ± 2.1
NNRF-MT	71.1 ± 2.1	35.7 ± 1.0
RoFA-MT	76.1 ± 2.4	34.0 ± 2.7
ResRoFA-MT	74.5 ± 2.0	36.7 ± 1.5

Table 6: Results with standard deviations on compositionality evaluation comparing to previous models.