Towards Universal Sentence Embeddings

Towards Universal Paraphrastic Sentence Embeddings
J. Wieting, M. Bansal, K. Gimpel and K. Livescu, ICLR 2016

A Simple But Tough-To-Beat Baseline For Sentence Embeddings
S. Arora, Y. Liang and T. Ma, ICLR 2017

Presented by Chunyuan Li

Outline

Paragram-phrase Embeddings

- Goal of **sentence embeddings**:

  Embed sentences into *a low-dimensional space*
  such that
  cosine similarity in the space corresponds to the strength of the paraphrase relationship between the sentences.

- A word sequence $x = \langle x_1, x_2, \cdots, x_n \rangle$
- **M**odel 1: Paragram-phrase (PP) embeddings

$$g_{\text{Paragram-phrase}}(x) = \frac{1}{n} \sum_{i}^{n} W_w^{x_i} \tag{1}$$

where $W_w^{x_i}$ is the **word embedding** for word x_i.

More Embeddings

- **M**odel 2: Adding Projection

$$g_{\text{proj}}(x) = W_p \left(\frac{1}{n} \sum_{i}^{n} W_w^{x_i} \right) + b \tag{2}$$

where W_p is the projection matrix and b is a bias vector.

- **M**odel 3: Generalization of M1 and M2 to multiple layers as well as nonlinear activation functions, *ie*, deep-averaging network (DAN).

More Embeddings

- **M**odel 4: Standard RNN

$$h_t = f(W_x W_w^{x_i} + W_h h_{t-1} + b) \tag{3}$$
$$g_{\text{RNN}}(x) = h_{-1} \tag{4}$$

where $\{W_x, W_h, b\}$ are parameters of standard RNN, f is activation function, and h_{-1} is hidden vector of the last token. embeddings.

- **M**odel 5: Identity-RNN (iRNN)
  - $(W_x = W_h = \mathbf{I}$, $b = 0$, $f(x) = x)$
  - Averaging output: h_{-1}/n
  - Intuition: richer architecture and can take into account word order

- **M**odel 6: Replace "standard RNN" module in M4 with LSTM

Training

- Notations:
  1. W_w: Trainable word embedding parameters
  2. W_c: All other trainable parameters (or "compositional parameters")
  3. Training data: a set X of phrase pairs $\langle x_1, x_2 \rangle$;
     t_1 and t_2 are carefully-selected negative examples (explained later)
- Training Objective:

$$\min_{W_c, W_w} \frac{1}{|X|} \Bigg(\sum_{\langle x_1, x_2 \rangle \in X} \max(0, \delta - \cos(g(x_1), g(x_2)) + \cos(g(x_1), g(t_1))) \\ + \max(0, \delta - \cos(g(x_1), g(x_2)) + \cos(g(x_2), g(t_2))) \Bigg) \\ + \lambda_c \|W_c\|^2 + \lambda_w \|W_{w_{initial}} - W_w\|^2$$

- Intuitions:
  1. First two terms:
     Two phrases to be more similar to each other ($\cos(g(x_1), g(x_2))$) than either is to their respective negative examples t_1 and t_2, by a margin of at least δ.
  2. Third term on W_c: weight decay
  3. Last term on W_w: word embedding should not be far from the pretrained embedding $W_{w_{\text{initial}}}$ from large corpora
- Dataset: Paraphrase Database (PPDB; Ganitkevitch et al., 2013).

Select negative examples

Two methods:

1. **MAX**: chooses the most similar phrase t_1 in some set of phrases (other than those in the given phrase pair $\langle x_1, x_2 \rangle$).

$$t_1 = \underset{t:\langle t,\cdot\rangle \in X_b \setminus \{\langle x_1, x_2\rangle\}}{\arg\max} \cos(g(x_1), g(t)) \tag{5}$$

   where $X_b \subseteq X$ is the current mini-batch

2. **MIX**: selects negative examples using MAX with probability 0.5 and selects them randomly from the mini-batch otherwise.

Results on transfer learning

Dataset	50%	75%	Max	PP	proj.	DAN	RNN	iRNN	LSTM (no o.g.)	LSTM (o.g.)	ST	GloVe	PSL
MSRpar	51.5	57.6	73.4	42.6	43.7	40.3	18.6	43.4	16.1	9.3	16.8	**47.7**	41.6
MSRvid	75.5	80.3	88.0	**74.5**	74.0	70.0	66.5	73.4	71.3	71.3	41.7	63.9	60.0
SMT-eur	44.4	48.1	56.7	47.3	**49.4**	43.8	40.9	47.1	41.8	44.3	35.2	46.0	42.4
OnWN	608	65.9	72.7	**70.6**	70.1	65.9	63.1	70.1	65.2	56.4	29.7	55.1	63.0
SMT-news	40.1	45.4	60.9	58.4	**62.8**	60.0	51.3	58.1	60.8	51.0	30.8	49.6	57.0
STS 2012 Average	54.5	59.5	70.3	58.7	**60.0**	56.0	48.1	58.4	51.0	46.4	30.8	52.5	52.8
headline	64.0	68.3	78.4	72.4	72.6	71.2	59.5	**72.8**	57.4	48.5	34.6	63.8	68.8
OnWN	52.8	64.8	84.3	67.7	68.0	64.1	54.6	**69.4**	68.5	50.4	10.0	49.0	48.0
FNWN	32.7	38.1	58.2	43.9	**46.8**	43.1	30.9	45.3	24.7	38.4	30.4	34.2	37.9
SMT	31.8	34.6	40.4	39.2	**39.8**	38.3	33.8	39.4	30.1	28.8	24.3	22.3	31.0
STS 2013 Average	45.3	51.4	65.3	55.8	**56.8**	54.2	44.7	56.7	45.2	41.5	24.8	42.3	46.4
deft forum	36.6	46.8	53.1	48.7	**51.1**	49.0	41.5	49.0	44.2	46.1	12.9	27.1	37.2
deft news	66.2	74.0	78.5	**73.1**	72.2	71.7	53.7	72.4	52.8	39.1	23.5	68.0	67.0
headline	67.1	75.4	78.4	69.7	**70.8**	69.2	57.5	70.2	57.5	50.9	37.8	59.5	65.3
images	75.6	79.0	83.4	**78.5**	78.1	76.9	67.6	78.2	68.5	62.9	51.2	61.0	62.0
OnWN	78.0	81.1	87.5	78.8	**79.5**	75.7	67.7	78.8	76.9	61.7	23.3	58.4	61.1
tweet news	64.7	72.2	79.2	76.4	75.8	74.2	58.0	**76.9**	58.7	48.2	39.9	51.2	64.7
STS 2014 Average	64.7	71.4	76.7	70.9	**71.3**	69.5	57.7	70.9	59.8	51.5	31.4	54.2	59.5
answers-forums	61.3	68.2	73.9	**68.3**	65.1	62.6	32.8	67.4	51.9	50.7	36.1	30.5	38.8
answers-students	67.6	73.6	78.8	**78.2**	77.8	78.1	64.7	78.2	71.5	55.7	33.0	63.0	69.2
belief	67.7	72.2	77.2	**76.2**	75.4	72.0	51.9	75.9	61.7	52.6	24.6	40.5	53.2
headline	74.2	80.8	84.2	74.8	**75.2**	73.5	65.3	75.1	64.0	56.6	43.6	61.8	69.0
images	80.4	84.3	87.1	**81.4**	80.3	77.5	71.4	81.1	70.4	64.2	17.7	67.5	69.9
STS 2015 Average	70.2	75.8	80.2	**75.8**	74.8	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.0
2014 SICK	71.4	79.9	82.8	71.6	**71.6**	70.7	61.2	71.2	63.9	59.0	49.8	65.9	66.4
2015 Twitter	49.9	52.5	61.9	52.9	52.8	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	36.3

Figure: textual similarity (Pearson's r $\times$ 100).

- Performance Ranking: PP $\simeq$ Proj. $>$ iRNN $>$ LSTM $>$ others

Results as initialization and regularization

- Tasks: The SICK similarity task, the SICK entailment task, and the Stanford Sentiment Treebank (SST) binary classification task
- **Initialize** each respective model to the learned parameters from PPDB
- **Regularize** the task-depedent objective using learned parameters

Task	word averaging	proj.	DAN	RNN	LSTM (no o.g.)	LSTM (o.g.)	w/ *universal* regularization
similarity (SICK)	**86.40**	85.93	85.96	73.13	85.45	83.41	**86.84**
entailment (SICK)	**84.6**	84.0	84.5	76.4	83.2	82.0	**85.3**
binary sentiment (SST)	83.0	83.0	83.4	86.5	86.6	**89.2**	86.9

- Embeddings as features, without updating.

Task	PARAGRAM-PHRASE			skip-thought	
	300	1200	2400	uni-skip	bi-skip
similarity (SICK)	82.15	82.85	**84.94**	84.77	84.05
entailment (SICK)	80.2	80.1	**83.1**	-	-
binary sentiment (SST)	**79.7**	78.8	79.4	-	-

Re-thinking Paragram-phase embeddings

- Pros: Simply averaging word embedding achieves impressive results
- Cons: Paired phases are needed for training.

Towards universal sentence embeddings
with large unlabeled training sets?

Random walk model for word embeddings

Latent variable generative model for text (Arora et al., 2016).

- Latent variables:
  1. A discourse vector $c_t \in \mathbb{R}^d$ represent "what is being talked about" ;
  2. Vector representation v_w of word w
- The probability of observing a word w at time t

$$Pr[w \text{ emitited at time } t|c_t] \propto \exp(\langle c_t, v_w \rangle) \quad (6)$$

- The discourse vector c_t does a slow random walk, so that nearby words are generated under similar discourses.
- It generates behavior that fits empirical works like word2vec, in terms of word-word cooccurrence probabilities

Random walk model for sentence embeddings

A single discourse vector c_s governs a sentence

- The probability of observing a word w in sentence s:

$$Pr[w \text{ emitited in sentence } \; s|c_s] = \alpha p(w) + (1-\alpha)\frac{\exp(\langle \tilde{c}_s, v_w \rangle)}{Z_{\tilde{c}_s}} \qquad (7)$$

  where $\tilde{c}_s = \beta c_0 + (1-\beta)c_s$, $c_0 \perp c_s$; $\quad Z_{\tilde{c}_s}$ is the normalizing constant

- Two types of "smoothing term" to allows a word w unrelated to the discourse c_s to be emitted ($\alpha = \beta = 0$ reduces to original model):
  1. c_0 is introduces as a common discourse, accouting for some frequent words (presumably "the", "and " etc.)
  2. $p(w)$ allows that some words occur out of context, even if they have low inner products with c_s

Computing sentence embeddings

- Assume that the word v_w's are roughly uniformly dispersed, then $Z_{\tilde{c}_s}$ is roughly the same, denoted as Z for all $\tilde{c}_s$.

$$L = \log p[s|c_s] = \sum_{w \in s} \log p(w|c_s) \tag{8}$$

$$= \sum_{w \in s} \log \left[\alpha p(w) + (1-\alpha) \frac{\exp(\langle \tilde{c}_s, v_w \rangle)}{Z} \right] = \sum_{w \in s} \log f_w(\tilde{c}_s) \tag{9}$$

- By Taylor expansion

$$f_w(\tilde{c}_s) \approx \text{constant} + \frac{(1-\alpha)/(\alpha Z)}{p(w) + (1-\alpha)/(\alpha Z)} \langle \tilde{c}_s, v_w \rangle \tag{10}$$

- Maximum likelihood estimator for $\tilde{c}_s$:

$$\tilde{c}_s^* = \arg\max \sum_{w \in s} f_w(\tilde{c}_s) \propto \sum_{w \in s} \frac{a}{p(w) + a} v_w, \quad \text{where} \quad a = \frac{1-\alpha}{\alpha Z} \tag{11}$$

- c_0 by computing the first principal component of $\tilde{c}_s$'s

Algorithm

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.

Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$

1: **for all** sentence s in $\mathcal{S}$ **do**

2: $\quad v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$

3: **end for**

4: Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector

5: **for all** sentence s in $\mathcal{S}$ **do**

6: $\quad v_s \leftarrow v_s - uu^\top v_s$

7: **end for**

1. Weighted average of the word vectors, using *smooth inverse frequency* (SIF).
2. Remove the projections of the average vectors on their first principal component (“common component removal”)

Results

- Transfer learning and supervised learning

	Results collected from (Wieting et al., 2016) except tfidf-GloVe											Our approach	
Supervised or not	Su.							Un.			Se.	Un.	Se.
Tasks	PP	PP -proj.	DAN	RNN	iRNN	LSTM (no)	LSTM (o.g.)	ST	avg-GloVe	tfidf-GloVe	avg-PSL	GloVe +WR	PSL +WR
STS'12	58.7	**60.0**	56.0	48.1	58.4	51.0	46.4	30.8	52.5	58.7	52.8	56.2	59.5
STS'13	55.8	56.8	54.2	44.7	56.7	45.2	41.5	24.8	42.3	52.1	46.4	56.6	**61.8**
STS'14	70.9	71.3	69.5	57.7	70.9	59.8	51.5	31.4	54.2	63.8	59.5	68.5	**73.5**
STS'15	75.8	74.8	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	60.0	71.7	**76.3**
SICK'14	71.6	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	66.4	72.2	**72.9**
Twitter'15	52.9	52.8	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	36.3	48.0	49.0

	PP	DAN	RNN	LSTM (no)	LSTM (o.g.)	skip-thought	Ours
similarity (SICK)	84.9	85.96	73.13	85.45	83.41	85.8	**86.03**
entailment (SICK)	83.1	84.5	76.4	83.2	82.0	-	**84.6**
sentiment (SST)	79.4	83.4	86.5	86.6	**89.2**	-	82.2

- Results
    - (a) Random Walk (WR)-based sentence embeddings is a more effective way to use word embeddings, and can achieve state-of-the-art
    - (b) WR-based initialization is significantly better for downstream supervised tasks.

Summary

Simple manipulation of word embeddings
leads to
state-of-the-art sentence embeddings

1. Averaging (perhaps with linear projection)
2. Weighted Averaging
    - smooth incerse frequency
    - common compoent removal