Learned in Translation: Contextualized Word Vectors

Bryan McCann, James Bradbury, Caiming Xiong, Richard Socher

NIPS 2017

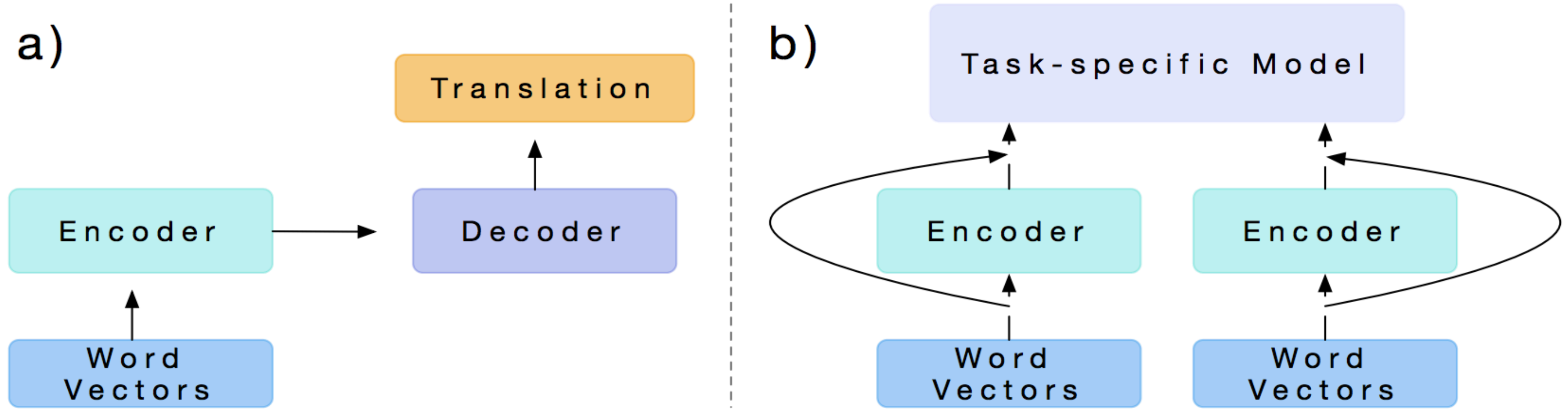
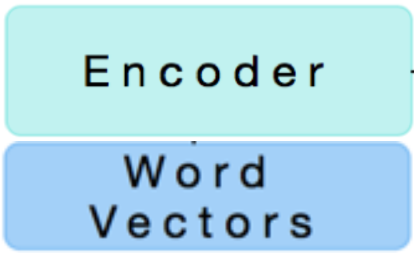


Figure 1: We a) train a two-layer, bidirectional LSTM as the encoder of an attentional sequence-to-sequence model for machine translation and b) use it to provide more context for other NLP models.

Context Vectors (CoVe)

$$\mathrm{CoVe}(w) = \text{MT-LSTM}(\mathrm{GloVe}(w))$$

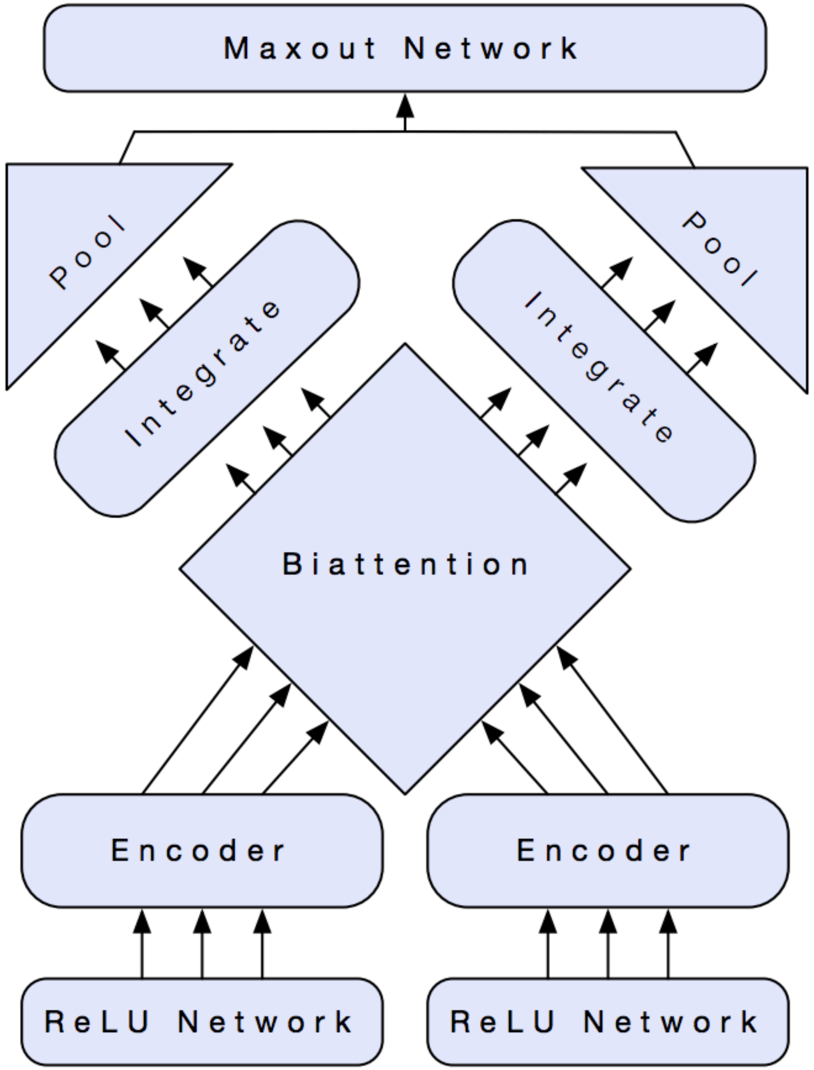
$$\tilde{w} = [\mathrm{GloVe}(w); \mathrm{CoVe}(w)]$$



Classification with CoVe

$$x = \text{biLSTM}\left(f(\tilde{w}^x)\right)$$

$$y = \text{biLSTM}\left(f(\tilde{w}^y)\right)$$



These sequences are each stacked along the time axis to get matrices X and Y.

$$A = XY^\top$$

$$A_x = \text{softmax}\left(A\right) \qquad A_y = \text{softmax}\left(A^\top\right)$$

$$C_x = A_x^\top X \qquad C_y = A_y^\top Y$$

$$X_{|y} = \text{biLSTM}\left([X; X - C_y; X \odot C_y]\right)$$

$$Y_{|x} = \text{biLSTM}\left([Y; Y - C_x; Y \odot C_x]\right)$$

$$\beta_x = \text{softmax}\left(X_{|y} v_1 + d_1\right) \qquad \beta_y = \text{softmax}\left(Y_{|x} v_2 + d_2\right)$$

$$x_{\text{self}} = X_{|y}^\top \beta_x \qquad y_{\text{self}} = Y_{|x}^\top \beta_y$$

$$x_{\text{pool}} = \left[\max(X_{|y}); \text{mean}(X_{|y}); \min(X_{|y}); x_{\text{self}}\right]$$

$$y_{\text{pool}} = \left[\max(Y_{|x}); \text{mean}(Y_{|x}); \min(Y_{|x}); y_{\text{self}}\right]$$

Question Answering with CoVe

$$x = \text{biLSTM}\left(f(\tilde{w}^x)\right)$$

$$y = \text{biLSTM}\left(f(\tilde{w}^y)\right)$$

f is a tanh activation

x is the document and y is the question in the question-document pair.

Experiments

			GloVe+				
Dataset	Random	GloVe	Char	CoVe-S	CoVe-M	CoVe-L	Char+CoVe-L
SST-2	84.2	88.4	90.1	89.0	90.9	91.1	**91.2**
SST-5	48.6	53.5	52.2	54.0	54.7	54.5	**55.2**
IMDb	88.4	91.1	91.3	90.6	91.6	91.7	**92.1**
TREC-6	88.9	94.9	94.7	94.7	95.1	95.8	**95.8**
TREC-50	81.9	89.2	89.8	89.6	89.6	90.5	**91.2**
SNLI	82.3	87.7	87.7	87.3	87.5	87.9	**88.1**
SQuAD	65.4	76.0	78.1	76.5	77.1	79.5	**79.9**

Table 2: CoVe improves validation performance. CoVe has an advantage over character n-gram embeddings, but using both improves performance further. Models benefit most by using an MT-LSTM trained with MT-Large (CoVe-L). Accuracy reported for classification tasks; F1 for SQuAD.

Experiments

Model	Reference	EM	F1
LR	Rajpurkar et al. [2016]	40.0	51.0
DCR	Yu et al. [2017]	62.5	72.1
M-LSTM+AP	Wang and Jiang [2017]	64.1	73.9
DCN+Char	Xiong et al. [2017]	65.4	75.6
BiDAF	Seo et al. [2017]	68.0	77.3
R-NET	Wang et al. [2017]	71.1	79.5
DCN+Char+CoVe	***Ours***	***71.3***	***79.9***

Table 4: Validation exact match and F1 for single-model question answering.

	Model	Test		Model	Test
SST-2	P-LSTM [Wieting et al., 2016]	89.2	TREC-6	SVM [da Silva et al., 2011]	95.0
	CT-LSTM [Looks et al., 2017]	89.4		SVM [Van-Tu and Anh-Cuong, 2016]	95.2
	TE-LSTM [Huang et al., 2017]	89.6		DSCNN-P [Zhang et al., 2016]	95.6
	NSE [Munkhdalai and Yu, 2016a]	89.7		*BCN+Char+CoVe (Ours)*	*95.8*
	BCN+Char+CoVe (Ours)	*90.3*		TBCNN [Mou et al., 2015]	96.0
	bmLSTM [Radford et al., 2017]	**91.8**		**LSTM-CNN [Zhou et al., 2016]**	**96.1**
SST-5	MVN [Guo et al., 2017]	51.5	TREC-50	SVM [Loni et al., 2011]	89.0
	DMN [Kumar et al., 2016]	52.1		SNoW [Li and Roth, 2006]	89.3
	LSTM-CNN [Zhou et al., 2016]	52.4		*BCN+Char+CoVe (Ours)*	*90.2*
	TE-LSTM [Huang et al., 2017]	52.6		RulesUHC [da Silva et al., 2011]	90.8
	NTI [Munkhdalai and Yu, 2016b]	53.1		SVM [Van-Tu and Anh-Cuong, 2016]	91.6
	BCN+Char+CoVe (Ours)	***53.7***		**Rules [Madabushi and Lee, 2016]**	**97.2**
IMDb	*BCN+Char+CoVe (Ours)*	*91.8*	SNLI	DecAtt+Intra [Parikh et al., 2016]	86.8
	SA-LSTM [Dai and Le, 2015]	92.8		NTI [Munkhdalai and Yu, 2016b]	87.3
	bmLSTM [Radford et al., 2017]	92.9		re-read LSTM [Sha et al., 2016]	87.5
	TRNN [Dieng et al., 2016]	93.8		btree-LSTM [Paria et al., 2016]	87.6
	oh-LSTM [Johnson and Zhang, 2016]	94.1		600D ESIM [Chen et al., 2016]	88.0
	Virtual [Miyato et al., 2017]	**94.1**		***BCN+Char+CoVe (Ours)***	***88.1***

Table 3: Single model test accuracies for classification tasks.

Deep Contextualized Word Representations

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer

NAACL 2018

ELMo: Embeddings from Language Models

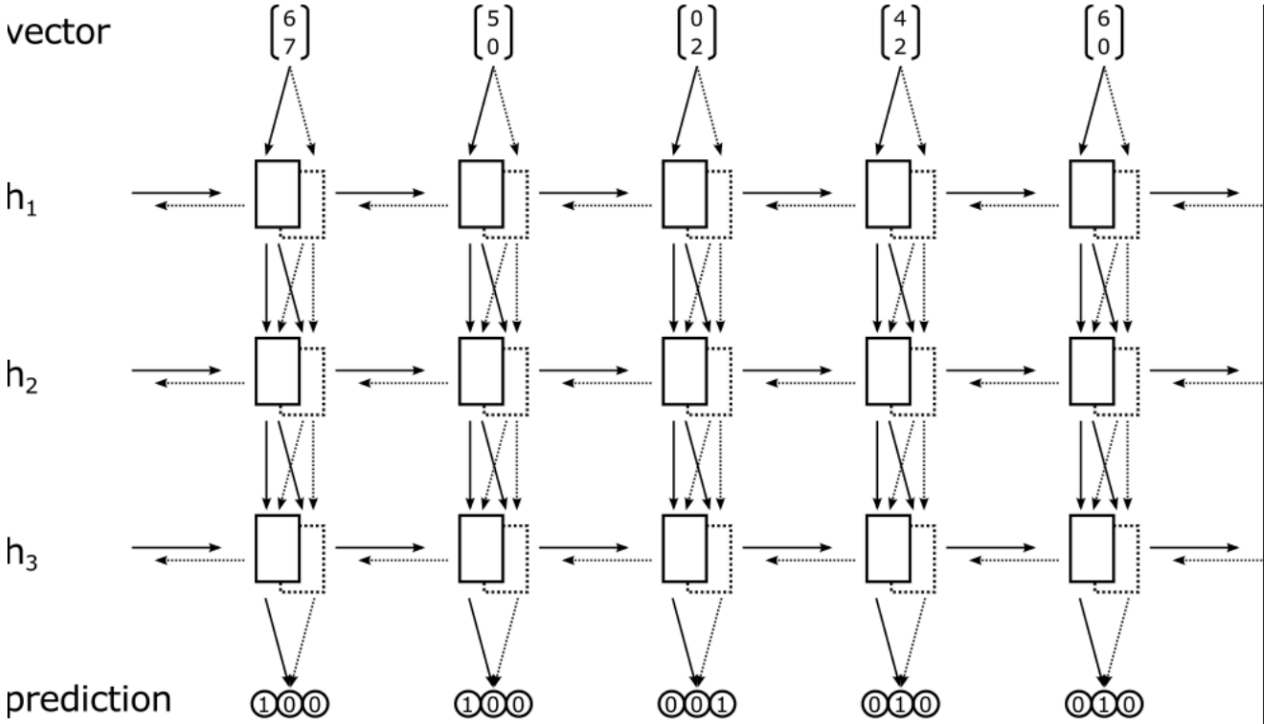
For each token t_k, a L-layer biLM computes a set of 2L + 1 representations

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \dots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \dots, L\}, \end{aligned}$$

where $\mathbf{h}_{k,0}^{LM}$ is the token layer and $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$, for each biLSTM layer.

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

In (1), $\mathbf{s}^{task}$ are softmax-normalized weights and the scalar parameter γ^{task} allows the task model to scale the entire ELMo vector. γ is of practical importance to aid the optimization process



Using biLMs for supervised NLP tasks

Concatenate the ELMo vector $\mathbf{ELMo}_k^{task}$ with $\mathbf{x}_k$ and pass the ELMo enhanced representation $[\mathbf{x}_k; \mathbf{ELMo}_k^{task}]$ into the task RNN.

For some tasks (e.g., SNLI, SQuAD), we observe further improvements by also including ELMo at the output of the task RNN by introducing another set of output specific linear weights and replacing $\mathbf{h}_k$ with $[\mathbf{h}_k; \mathbf{ELMo}_k^{task}]$.

Experiments

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

Experiments

- we found it beneficial to add a moderate amount of dropout to ELMo and in some cases to regularize the ELMo weights by adding $\lambda\|\mathbf{w}\|_2^2$ to the loss.

- For some tasks (e.g., SNLI, SQuAD), we observe further improvements by also including ELMo at the output of the task RNN by introducing another set of output specific linear weights and replacing $\mathbf{h}_k$ with $[\mathbf{h}_k; \mathbf{ELMo}_k^{task}]$.

Task	Baseline	Last Only	All layers λ=1	All layers λ=0.001
SQuAD	80.8	84.7	85.0	**85.2**
SNLI	88.1	89.1	89.3	**89.5**
SRL	81.6	84.1	84.6	**84.8**

Table 2: Development set performance for SQuAD, SNLI and SRL comparing using all layers of the biLM (with different choices of regularization strength λ) to just the top layer.

Task	Input Only	Input & Output	Output Only
SQuAD	85.1	**85.6**	84.8
SNLI	88.9	**89.5**	88.7
SRL	**84.7**	84.3	80.9

Table 3: Development set performance for SQuAD, SNLI and SRL when including ELMo at different locations in the supervised model.

Experiments: What information is captured by the biLM's representations?

- Word sense disambiguation

  accuracies using the second biLM layer are higher than the first layer

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

Model	$\mathbf{F_1}$
WordNet 1st Sense Baseline	65.9
Raganato et al. (2017a)	69.9
Iacobacci et al. (2016)	**70.1**
CoVe, First Layer	59.4
CoVe, Second Layer	*64.7*
biLM, First layer	67.4
biLM, Second layer	*69.0*

Table 5: All-words fine grained WSD F_1. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Experiments: What information is captured by the biLM's representations?

- Word sense disambiguation
- Basic syntax

  accuracies using the first biLM layer are higher than the top layer

  different layers in the biLM represent different types of information

Model	**Acc.**
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

Table 6: Test set POS tagging accuracies for PTB. For CoVe and the biLM, we report scores for both the first and second layer biLSTMs.

Experiments: Sample Efficiency

- the SRL model reaches a maximum development F1 after 486 epochs of training without ELMo. After adding ELMo, the model exceeds the baseline maximum at epoch 10.

- ELMo-enhanced models use smaller training sets more efficiently than models without ELMo.

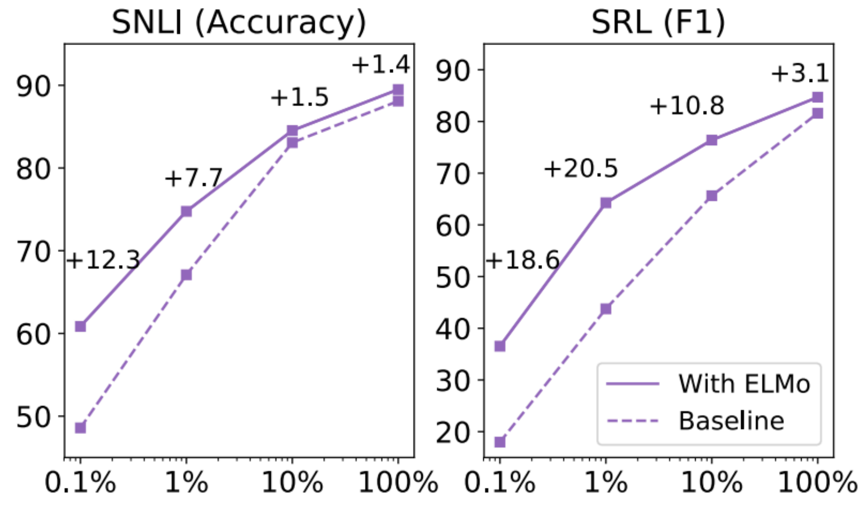


Figure 1: Comparison of baseline vs. ELMo performance for SNLI and SRL as the training set size is varied from 0.1% to 100%.

Universal Sentence Encoder

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brain Strope, Ray Kurzeweil

Google

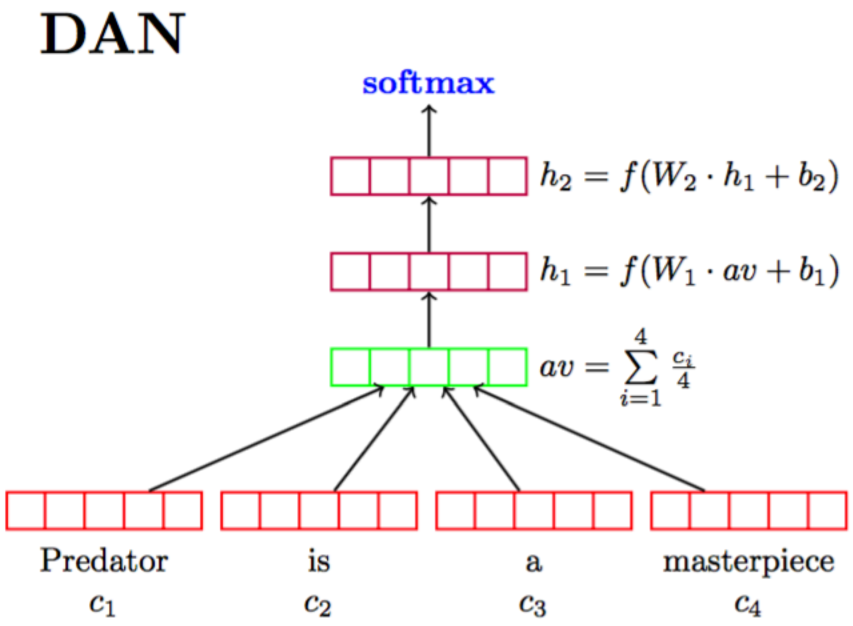
Method

Present two models for producing sentence embeddings that demonstrate good transfer to other NLP tasks

- transformer based
  - compute the element-wise-sum of the representations at each word position
  - divide by the square root of the length of the sentence
- deep averaging network (DAN) based
  - input embeddings for words and bi-grams are first averaged together and then passed through a feedforward deep neural network

Pre-trained strategy:

multi-task learning: skip-thought like task; conversational input-response task; classification task



Transfer Learning Models

- For sentence classification transfer tasks, the output of the transformer and DAN sentence encoders are provided to a task specific DNN

- For the pairwise semantic similarity task, we directly assess the similarity of the sentence embeddings produced by the two encoders

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \left(1 - \arccos\left(\frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||\;||\mathbf{v}||}\right)/\pi\right) \quad (1)$$

Experiments

- baseline:
  - random-initialized word embeddings are fed into CNN or DAN.
  - pretrained word embeddings (word2vec skip-gram) are fed into CNN or DAN.

- combined transfer models:
  - combine the sentence and word level transfer models by concatenating their representations prior to feeding the combined representation to the transfer task classification layers

Experiments

- transformer based sentence encoder > DAN based sentence encoder
- sentence + word level transfer > sentence level transfer > word level transfer

Model	MR	CR	SUBJ	MPQA	TREC	SST	STS Bench (dev / test)
Sentence & Word Embedding Transfer Learning							
USE_D+DAN (w2v w.e.)	77.11	81.71	93.12	87.01	94.72	82.14	–
USE_D+CNN (w2v w.e.)	78.20	82.04	93.24	85.87	97.67	85.29	–
USE_T+DAN (w2v w.e.)	81.32	86.66	93.90	88.14	95.51	86.62	–
USE_T+CNN (w2v w.e.)	81.18	87.45	93.58	87.32	98.07	86.69	–
Sentence Embedding Transfer Learning							
USE_D	74.45	80.97	92.65	85.38	91.19	77.62	0.763 / 0.719 (r)
USE_T	81.44	87.43	93.87	86.98	92.51	85.38	0.814 / 0.782 (r)
USE_D+DAN (lrn w.e.)	77.57	81.93	92.91	85.97	95.86	83.41	–
USE_D+CNN (lrn w.e.)	78.49	81.49	92.99	85.53	97.71	85.27	–
USE_T+DAN (lrn w.e.)	81.36	86.08	93.66	87.14	96.60	86.24	–
USE_T+CNN (lrn w.e.)	81.59	86.45	93.36	86.85	97.44	87.21	–
Word Embedding Transfer Learning							
DAN (w2v w.e.)	74.75	75.24	90.80	81.25	85.69	80.24	–
CNN (w2v w.e.)	75.10	80.18	90.84	81.38	97.32	83.74	–
Baselines with No Transfer Learning							
DAN (lrn w.e.)	75.97	76.91	89.49	80.93	93.88	81.52	–
CNN (lrn w.e.)	76.39	79.39	91.18	82.20	95.82	84.90	–

Table 2: Model performance on transfer tasks. *USE_T* is the universal sentence encoder (USE) using Transformer. *USE_D* is the universal encoder DAN model. Models tagged with *w2v w.e.* make use of pre-training word2vec skip-gram embeddings for the transfer task model, while models tagged with *lrn w.e.* use randomly initialized word embeddings that are learned only on the transfer task data. Accuracy is reported for all evaluations except STS Bench where we report the Pearson correlation of the similarity scores with human judgments. Pairwise similarity scores are computed directly using the sentence embeddings from the universal sentence encoder as in Eq. (1).

Experiments

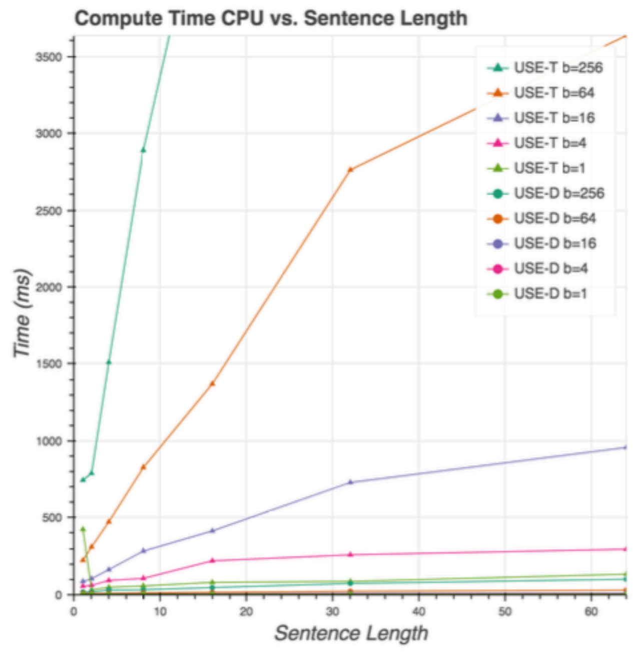
- for smaller quantities of data, sentence level transfer learning can achieve surprisingly good task performance
- as the training set size increases, models that do not make use of transfer learning approach the performance of the other models

Model	SST 1k	SST 2k	SST 4k	SST 8k	SST 16k	SST 32k	SST 67.3k
Sentence & Word Embedding Transfer Learning							
USE_D+DNN (w2v w.e.)	78.65	78.68	79.07	81.69	81.14	81.47	82.14
USE_D+CNN (w2v w.e.)	77.79	79.19	79.75	82.32	82.70	83.56	85.29
USE_T+DNN (w2v w.e.)	85.24	84.75	85.05	86.48	86.44	86.38	86.62
USE_T+CNN (w2v w.e.)	84.44	84.16	84.77	85.70	85.22	86.38	86.69
Sentence Embedding Transfer Learning							
USE_D	77.47	76.38	77.39	79.02	78.38	77.79	77.62
USE_T	84.85	84.25	85.18	85.63	85.83	85.59	85.38
USE_D+DNN (lrn w.e.)	75.90	78.68	79.01	82.31	82.31	82.14	83.41
USE_D+CNN (lrn w.e.)	77.28	77.74	79.84	81.83	82.64	84.24	85.27
USE_T+DNN (lrn w.e.)	84.51	84.87	84.55	85.96	85.62	85.86	86.24
USE_T+CNN (lrn w.e.)	82.66	83.73	84.23	85.74	86.06	86.97	87.21
Word Embedding Transfer Learning							
DNN (w2v w.e.)	66.34	69.67	73.03	77.42	78.29	79.81	80.24
CNN (w2v w.e.)	68.10	71.80	74.91	78.86	80.83	81.98	83.74
Baselines with No Transfer Learning							
DNN (lrn w.e.)	66.87	71.23	73.70	77.85	78.07	80.15	81.52
CNN (lrn w.e.)	67.98	71.81	74.90	79.14	81.04	82.72	84.90

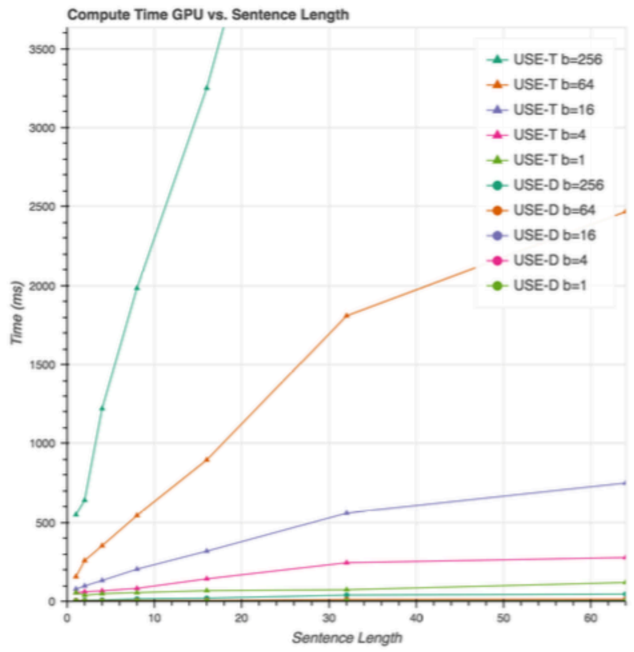
Table 3: Task performance on SST for varying amounts of training data. SST 67.3k represents the full training set. Using only 1,000 examples for training, transfer learning from USE_T is able to obtain performance that rivals many of the other models trained on the full 67.3 thousand example training set.

Experiments

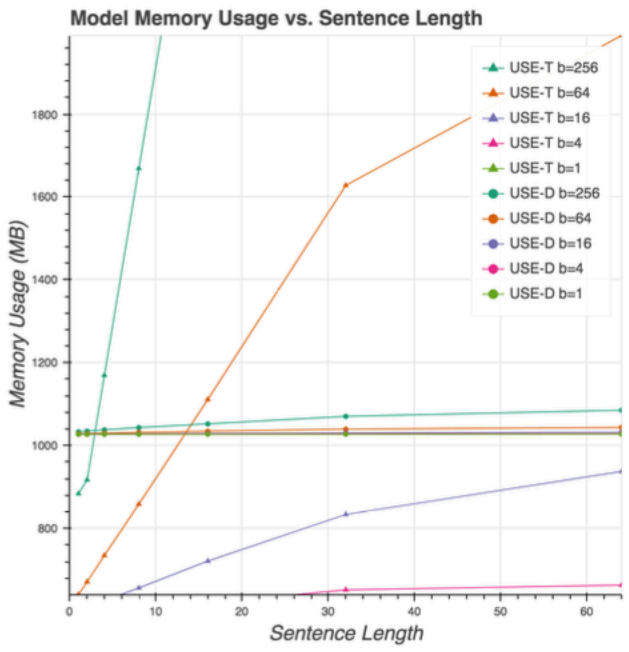
- compute time and memory usage for transformer increases noticeably as sentence length increases
- the compute time and memory usage for the DAN model stays nearly constant as sentence length is increased.



(a) CPU Time vs. Sentence Length



(b) GPU Time vs. Sentence Length



(c) Memory vs. Sentence Length