Learning General Purpose Distributed Sentence Representations via Large Scale Multitask Learning

LING 575

05/06/2019

Elijah Rippeth and Daniel Campos

Background

- Transfer learning has shown improvements in many NLP and computer vision tasks.
  - Visual QA, image classification with ImageNet transfer
  - Reading comp, sequence labeling with word2vec, GloVe transfer
- Systems often need to learn representation of *sequences* of words.
  - Pretrained embeddings used and representation learned from scratch.
  - Supervised signal used in a task-specific manner.
- We need transfer learning for sentence representations!

Problem Statement

- If we try to learn sentence representations for many tasks and share an encoder, will our network be able to learn better sentence representation?
  - Yes!

Related work

- Kiros, 2015
  - Skip-Thought; tries to reconstruct the surrounding sentences of an encoded passage.
- Wieting, 2015
  - Supervision from the Paraphrase Database
- Hill, 2016
  - Sequential denoising autoencoders and shallow log-linear models.
- Conneau, 2017
  - Learning sentence representations from natural language inference data
- McCann, 2017
  - Representations learned by state-of-the-art large-scale NMT systems
- Nie, 2017
  - Discourse-based objectives
- Pagliardini, 2017
  - Compositional n-Gram Features

The algorithm: intuition

Training neural nets by way of specific tasks will imbue them with inductive bias.

- NMT systems appear to capture morphology
- Sequence-to-sequence parsers more strongly encode source language syntax.

If a model is trained on multiple tasks, we can encode these biases in a single representation.

- Potentially allows our representations to generalize better. (Baxter et al. (2000))

Tasks

- Skip-thought
  - Previous sentence
  - Next sentence
- NMT
  - En-Fr
  - En-De
- Constituency parsing
- NLI

Implementation details

- Five of six tasks are sequence-to-sequence tasks
  - Use encoder-decoder architecture
    - Encoder is biGRU
    - Decoder is conditional GRU; parameterization as follows:

$$\mathbf{r_t} = \sigma(\mathbf{W_r}x_t + \mathbf{U_r}\mathbf{h_{t-1}} + \mathbf{C_r}\mathbf{h_x})$$
$$\mathbf{z_t} = \sigma(\mathbf{W_z}x_t + \mathbf{U_z}\mathbf{h_{t-1}} + \mathbf{C_z}\mathbf{h_x})$$
$$\tilde{\mathbf{h_t}} = \tanh(\mathbf{W_d}x_t + \mathbf{U_d}(\mathbf{r_t} \odot \mathbf{h_{t-1}}) + \mathbf{C_d}\mathbf{h_x})$$
$$\mathbf{h_{t+1}} = (1 - \mathbf{z_t}) \odot \mathbf{h_{t-1}} + \mathbf{z_t} \odot \tilde{\mathbf{h_t}}$$

Training Details

Require: A set of k tasks with a common source language, a shared encoder $\mathbf{E}$ across all tasks and a set of k task specific decoders $\mathbf{D_1} \ldots \mathbf{D_k}$. Let θ denote each model's parameters, α a probability vector $(p_1 \ldots p_k)$ denoting the probability of sampling a task such that $\Sigma_i^k p_i = 1$, datasets for each task $\mathbb{P}_1 \ldots \mathbb{P}_k$ and a loss function L.

while θ *has not converged* **do**

1: Sample task $i \sim \mathbf{Cat}(k, \alpha)$.
2: Sample input, output pairs $\mathbf{x}, \mathbf{y} \sim \mathbb{P}_i$.
3: Input representation $\mathbf{h}_x \leftarrow \mathbf{E}_\theta(\mathbf{x})$.
4: Prediction $\tilde{\mathbf{y}} \leftarrow \mathbf{D}_{i_\theta}(\mathbf{h}_x)$
5: $\theta \leftarrow \text{Adam}(\nabla_\theta L(\mathbf{y}, \tilde{\mathbf{y}}))$.

end

Training Details (Continued)

- Model takes 7+ days on a P100 GPU
  - Faster than previous models which needed + 4 weeks
- Indicative of trend towards larger and more complex models
  - BERT
  - ELMO
  - GPT
- To add a task: add data and a decoder and co train.

Evaluation

Subramanian et al. borrow from Conneau et al. where they test their learned representations on novel domains unseen during training and without any retraining(aka transfer learning).

They also look at an unseen classification problem(Quora Duplicate Questions) to explore how their sentence representations can leverage learning on small data/low resource tasks.

Exploring how well their word embeddings perform they also use the standard word embedding evaluation benchmarks.

Evaluate what sentence representations on sentence characteristics and syntactic properties

Evaluation (Continued)

Since their training framework contains a multitude of task they run ablandation with performance on the entire suite of evaluations with various mixes of multi task learning.

- Allows them to observe what tasks bring the most gain to all benchmarks.
  - E.G. Adding constituency parsing improves performance on sentence relatedness and entailment.

For each dataset, they compare various of their multi task training methods and compare them both to existing sentence embeddings and to task specific SOTAs.

They compare these complex models to MLPs that leverage their sentence representations

Transfer Learning Benchmarks

Multi Task learning with MLP out performs most previous models by a fair margin.

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	SICK-R	SICK-E	STSB	Δ
Transfer approaches											
FastSent	70.8	78.4	88.7	80.6	-	76.8	72.2/80.3	-	-	-	-
FastSent+AE	71.8	76.7	88.8	81.5	-	80.4	71.2/79.1	-	-	-	-
NMT En-Fr	64.7	70.1	84.9	81.5	-	82.8	-	-	-	-	-
CNN-LSTM	77.8	82.1	93.6	89.4	-	92.6	76.5/83.8	0.862	-	-	-
Skipthought	76.5	80.1	93.6	87.1	82.0	92.2	73.0/82.0	0.858	82.3	-	-
Skipthought + LN	79.4	83.1	93.7	89.3	82.9	88.4	-	0.858	79.5	72.1/70.2	-
Word Embedding Average	-	-	-	-	82.2	-	-	0.860	84.6	-	-
DiscSent + BiGRU	-	-	88.6	-	-	81.0	71.6/-	-	-	-	-
DiscSent + unigram	-	-	92.7	-	-	87.9	72.5/-	-	-	-	-
DiscSent + embed	-	-	93.0	-	-	87.2	75.0/-	-	-	-	-
Byte mLSTM	**86.9**	**91.4**	**94.6**	88.5	-	-	75.0/82.8	0.792	-	-	-
Infersent (SST)	(*)	83.7	90.2	89.5	(*)	86.0	72.7/80.9	0.863	83.1	-	-
Infersent (SNLI)	79.9	84.6	92.1	89.8	83.3	88.7	75.1/82.3	0.885	86.3	-	-
Infersent (AllNLI)	81.1	86.3	92.4	90.2	**84.6**	88.2	76.2/83.1	0.884	86.3	75.8/75.5	0.0
Our Models											
+STN	78.9	85.8	93.7	87.2	80.4	84.2	72.4/81.6	0.840	82.1	72.9/72.4	-2.56
+STN +Fr +De	80.3	85.1	93.5	90.1	83.3	92.6	77.1/83.3	0.864	84.8	77.1/77.1	0.01
+STN +Fr +De +NLI	81.2	86.4	93.4	90.8	84.0	93.2	76.6/82.7	0.884	87.0	79.2/79.1	0.99
+STN +Fr +De +NLI +L	81.7	87.3	94.2	90.8	84.0	**94.2**	77.1/83.0	0.887	87.1	78.7/78.2	1.33
+STN +Fr +De +NLI +L +STP	82.7	88.0	94.1	91.2	84.5	92.4	77.8/83.9	0.885	86.8	78.7/78.4	1.44
+STN +Fr +De +NLI +2L +STP	82.8	88.3	94.0	**91.3**	83.6	92.6	77.4/83.3	0.884	87.6	**79.2/79.1**	1.47
+STN +Fr +De +NLI +L +STP +Par	82.5	87.7	94.0	90.9	83.2	93.0	**78.6/84.4**	**0.888**	**87.8**	78.9/78.6	**1.48**
Approaches trained from scratch on these tasks											
Naive Bayes SVM	79.4	81.8	93.2	86.3	83.1	-	-	-	-	-	
AdaSent	83.1	86.3	95.5	93.3	-	92.4	-	-	-	-	
TF-KLD	-	-	-	-	-	-	80.4/85.9	-	-	-	
Illinois LH	-	-	-	-	-	-	-	-	84.5	-	
Dependency tree LSTM	-	-	-	-	-	-	-	0.868	-	-	
Neural Semantic Encoder	-	-	-	-	89.7	-	-	-	-	-	
BLSTM-2DCNN	82.3	-	94.0	-	89.5	96.1	-	-	-	-	

Word Similarity

- Per table, multi task representation outperforms other representations and most competitive benchmarks.

Embedding	Dim	V143	SIMLEX	WS353	YP130	MTurk771	RG65	MEN	QVEC
SENNA	50	0.36	0.27	0.49	0.16	0.50	0.50	0.57	-
NMT En-Fr	620	-	0.46	0.49	-	0.46	0.59	0.49	-
GloVe6B	300	0.31	0.37	0.61	0.56	0.65	0.77	0.74	0.47
GloVe840B	300	0.34	0.41	0.71	0.57	0.71	0.76	**0.80**	-
Skipgram GoogleNews	300	-	0.44	0.70	-	0.67	0.76	0.74	-
FastText	300	0.40	0.38	**0.74**	0.53	0.67	**0.80**	0.76	0.50
Multilingual	512	0.26	0.40	0.68	0.42	0.60	0.65	0.76	-
Charagram	300	-	0.71	-	-	-	-	-	-
Attract-Repel	300	-	0.75	-	-	-	-	-	-
Our Model									
+STN +Fr +De +NLI +L	512	**0.56**	**0.59**	0.66	**0.62**	0.70	0.65	0.67	**0.54**

Low Resource Learning

- Training method learns very quickly
  - Scalable to low resource tasks

Model	Accuracy				
	1k	5k	10k	25k	All (400k)
Siamese-CNN	-	-	-	-	79.60
Multi-Perspective-CNN	-	-	-	-	81.38
Siamese-LSTM	-	-	-	-	82.58
Multi-Perspective-LSTM	-	-	-	-	83.21
L.D.C	-	-	-	-	85.55
BiMPM	-	-	-	-	88.17
DecAtt (GloVe)	-	-	-	-	86.52
DecAtt (Char)	-	-	-	-	86.84
pt-DecAtt (Word)	-	-	-	-	87.54
pt-DecAtt (Char)	-	-	-	-	**88.40**
CNN (Random)	56.3	59.2	63.8	68.9	-
CNN (GloVe)	58.5	62.4	66.1	70.2	-
LSTM-AE	59.3	63.8	67.2	70.9	-
Deconv-AE	60.2	65.1	67.7	71.6	-
Deconv LVM	62.9	67.6	69.0	72.4	-
Infersent (LogReg)	68.8	73.8	75.7	76.4	-
Infersent (MLP)	71.8	75.7	77.2	78.9	84.79
Our Models					
+STN +Fr +De +NLI +L +STP (LogReg)	70.4	74.8	75.9	77.2	-
+STN +Fr +De +NLI +L +STP (MLP)	**74.7**	**77.7**	**78.3**	**81.4**	87.01

Syntactic and Sentence Characteristics

- Scores Based on which models are able to predict certain characteristics

Model	Length	Content	Order	Passive	Tense	TSS
	Sentence characteristics			*Syntatic properties*		
Majority Baseline	22.0	50.0	50.0	81.3	77.1	56.0
Infersent (AllNLI)	75.8	**75.8**	75.1	92.1	93.3	70.4
Skipthought	-	-	-	94.0	96.5	74.7
Our Models						
Skipthought +Fr +De	86.8	75.7	81.1	97.0	97.0	77.1
Skipthought +Fr +De +NLI	88.1	75.7	81.5	96.7	96.8	77.1
Skipthought +Fr +De +NLI +Par	**93.7**	75.5	**83.1**	**98.0**	**97.6**	**80.7**

Future Work

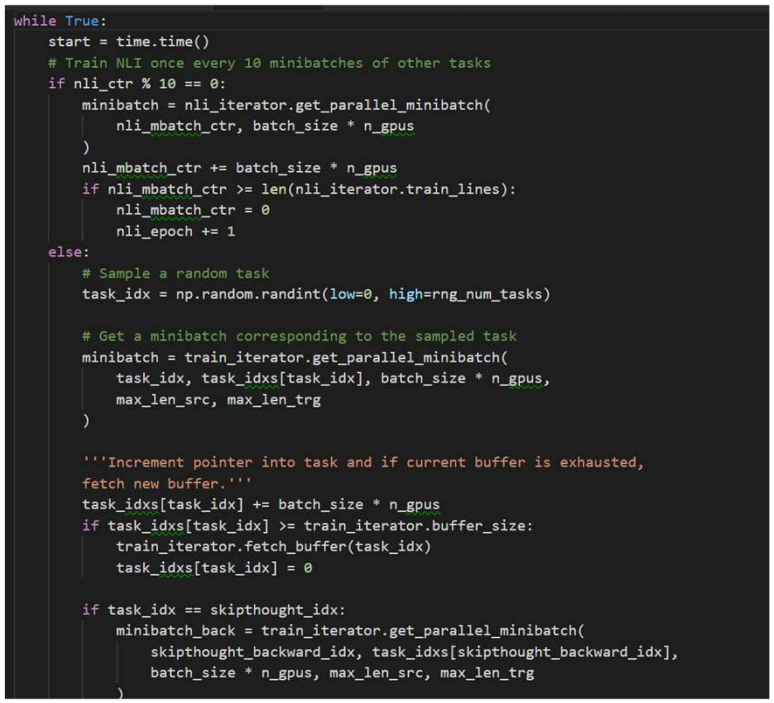
Interpret and understand what biases are learned by model in respect to tasks being added.

Use sentence representation + conditional language model to generate coherent text.

Discussion

- Sets the stage for larger fully end 2 end multi task learning
- Bigger Data/More Tasks ?= Better Model
- Longer Training ?= Better Model
- Build Complex General Models and Fine tune?
- How useful are sentence representations?

```
        if (
            task_idxs[skipthought_backward_idx] >=
            train_iterator.buffer_size
        ):
            train_iterator.fetch_buffer(skipthought_backward_idx)
            task_idxs[skipthought_backward_idx] = 0

        optimizer.zero_grad()
        decoder_logit, decoder_logit_2 = model(
            minibatch, task_idx, paired_trg=minibatch_back['input_trg']
        )

        loss_f = loss_criterion(
            decoder_logit.contiguous().view(-1, decoder_logit.size(2)),
            minibatch['output_trg'].contiguous().view(-1)
        )

        loss_b = loss_criterion(
            decoder_logit_2.contiguous().view(-1, decoder_logit_2.size(2)),
            minibatch_back['output_trg'].contiguous().view(-1)
        )

        task_losses[task_idx].append(loss_f.data[0])
        task_losses[skipthought_backward_idx].append(loss_b.data[0])
        loss = loss_f + loss_b

    else:
        optimizer.zero_grad()
        decoder_logit = model(minibatch, task_idx)

        loss = loss_criterion(
            decoder_logit.contiguous().view(-1, decoder_logit.size(2)),
            minibatch['output_trg'].contiguous().view(-1)
        )

        task_losses[task_idx].append(loss.data[0])

    loss.backward()
    torch.nn.utils.clip_grad_norm(model.parameters(), 1.)
    optimizer.step()

end = time.time()
mbatch_times.append(end - start)
```



```
while True:
    start = time.time()
    # Train NLI once every 10 minibatches of other tasks
    if nli_ctr % 10 == 0:
        minibatch = nli_iterator.get_parallel_minibatch(
            nli_mbatch_ctr, batch_size * n_gpus
        )
        nli_mbatch_ctr += batch_size * n_gpus
        if nli_mbatch_ctr >= len(nli_iterator.train_lines):
            nli_mbatch_ctr = 0
            nli_epoch += 1
    else:
        # Sample a random task
        task_idx = np.random.randint(low=0, high=rng_num_tasks)

        # Get a minibatch corresponding to the sampled task
        minibatch = train_iterator.get_parallel_minibatch(
            task_idx, task_idxs[task_idx], batch_size * n_gpus,
            max_len_src, max_len_trg
        )

        '''Increment pointer into task and if current buffer is exhausted,
        fetch new buffer.'''
        task_idxs[task_idx] += batch_size * n_gpus
        if task_idxs[task_idx] >= train_iterator.buffer_size:
            train_iterator.fetch_buffer(task_idx)
            task_idxs[task_idx] = 0

        if task_idx == skipthought_idx:
            minibatch_back = train_iterator.get_parallel_minibatch(
                skipthought_backward_idx, task_idxs[skipthought_backward_idx],
                batch_size * n_gpus, max_len_src, max_len_trg
            )
```

Code Details

Usage

Use sentence representations in a few lines of code

```
from gensen import GenSen, GenSenSingle
import numpy as np
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from matplotlib import pyplot

shakespeare = ['You speak an infinite deal of nothing.','These violent delights have violent ends And in their triump die, like fire and powder Which, as they kiss, consume']
wutang = ['Raw Ima give it to ya, with no trivia Raw like cocaine straight from Bolivia','Yo, nobody budge while I shot slugs Never shot thugs, Im runnin with thugs that flood mugs']
lukecombs = ['Seventeen, you dont think that much about life You just live it Like Kerosene dancing around a fire But youre in','I picked myself up off the floor And found something new worth liv
trump = ['The Kentucky Derby decision was not a good one. It was a rough & tumble race on a wet and sloppy track, actually, a beautiful thing to watch. Only in these days of political correctness
obama = ['Condolences to the family of John Singleton. His seminal work, Boyz n the Hood, remains one of the most searing, loving portrayals of the challenges facing inner-city youth. He opened d
idx2speaker = ['trump1','trump2','trumpinaguration','obama1','obama2','obamainaguration','shakespeare1','shakespeare2','wutang1','wutang2','lukecombs','lukecombs']
sentences = trump + obama + shakespeare + lukecombs + wutang
gensen_1 = GenSenSingle(
    model_folder='./data/models',
    filename_prefix='nli_large_bothskip',
    pretrained_emb='./data/embedding/glove.840B.300d.h5'
)
reps_h, reps_h_t = gensen_1.get_representation(
    sentences, pool='last', return_numpy=True, tokenize=True
)
x = []
for i in range(len(reps_h)):
    x.append(reps_h[i].mean(axis=0))

model = TSNE(n_components=2, perplexity=20, init='pca', method='exact', n_iter=5000)
x = model.fit_transform(x)
pyplot.figure(figsize=(20,20))
pyplot.xlim((np.min(x[:, 0])-10, np.max(x[:, 0])+10))
pyplot.ylim((np.min(x[:, 1])-10, np.max(x[:, 1])+10))
for i in range(len(x)):
    pyplot.text(x[i, 0], x[i, 1], idx2speaker[i], bbox=dict(facecolor='blue', alpha=0.1))

pyplot.savefig('TSNEGlove.png')
```

Text Representations Plotted