Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms

Dinghan Shen[1], Guoyin Wang[1], Wenlin Wang[1], Martin Renqiang Min[2], Qinliang Su[3], Yizhe Zhang[4], Chunyuan Li[1], Ricardo Henao[1], Lawrence Carin[1]

[1] Duke University, [2] NEC Laboratories America, [3] Sun Yat-sen University, [4] Microsoft Research

Association for Computational Linguistics

Motivation & Contribution

- Many deep learning architectures have been proposed to model the **compositionality** in text sequences (parameters & expensive computation);
- We performed a rigorous evaluation regarding the **added value** of sophisticated compositional functions;
- Surprisingly, **Simple Word-Embedding-based Models (SWEMs)** exhibit comparable or even superior performance in the majority of cases considered;
- The **underlying reasons** are furthered investigated.

Models

- Consider a text sequence represented as X, composed of a sequence of words. Let $\{v_1, v_2, ..., v_L\}$ denote the respective word embeddings for each token, where L is the sentence/document length;
- The compositional function, $X \rightarrow z$, aims to combine word embeddings into a fixed-length sentence/document representation z. Typically, LSTM or CNN are employed for this purpose;
- To investigate the raw modeling capacity of word embeddings, we consider a class of models ***with no additional compositional parameters*** to encode natural language sequences, termed SWEMs:

SWEM-*aver* ***(average-pooling)***: $z = \frac{1}{L}\sum_{i=1}^{L} v_i$.

SWEM-*max* ***(Max-pooling)***: $z = \text{Max-pooling}(v_1, v_2, ..., v_L)$.

SWEM-*concat* ***(Both features are concatenated)***

SWEM-*hier* ***(hierarchical-pooling)***:

- ***Locally***: an average-pooling is performed on each local window, $v_{i:i+n-1}$
- ***Globally***: a max-pooling operation is further applied on top of the representations for every window

This strategy preserves ***the local spatial information*** of a text sequence

Model	Parameters	Complexity	Sequential Ops
CNN	$n \cdot K \cdot d$	$\mathcal{O}(n \cdot L \cdot K \cdot d)$	$\mathcal{O}(1)$
LSTM	$4 \cdot d \cdot (K + d)$	$\mathcal{O}(L \cdot d^2 + L \cdot K \cdot d)$	$\mathcal{O}(L)$
SWEM	0	$\mathcal{O}(L \cdot K)$	$\mathcal{O}(1)$

Table : Comparisons of CNN, LSTM and SWEM architectures. Columns correspond to the number of compositional parameters, computational complexity and sequential operations, respectively.

Experiments

Document Classification:

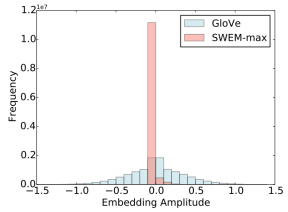
- **Empirical results:**

Model	Yahoo! Ans.	AG News	Yelp P.	Yelp F.	DBpedia
Bag-of-means*	60.55	83.09	87.33	53.54	90.45
Small word CNN*	69.98	89.13	94.46	58.59	98.15
Large word CNN*	70.94	91.45	95.11	59.48	98.28
LSTM*	70.84	86.06	94.74	58.17	98.55
Deep CNN (29 layer)†	73.43	91.27	**95.72**	**64.26**	**98.71**
fastText ‡	72.0	91.5	93.8	60.4	98.1
fastText (bigram)‡	72.3	92.5	95.7	63.9	98.6
SWEM-*aver*	73.14	91.71	93.59	60.66	98.42
SWEM-*max*	72.66	91.79	93.25	59.63	98.24
SWEM-*concat*	**73.53**	**92.66**	93.76	61.11	**98.57**
SWEM-*hier*	73.48	92.48	**95.81**	**63.79**	98.54

- **Analysis**

Model	Parameters	Speed
CNN	541K	171s
LSTM	1.8M	598s
SWEM	**61K**	**63s**

Table: Speed & Parameters on Yahoo! Answer dataset.



Politics	Science	Computer	Sports	Chemistry	Finance	Geoscience
philipdru	coulomb	system32	billups	sio2 (SiO_2)	proprietorship	fossil
justices	differentiable	cobol	midfield	nonmetal	ameritrade	zoos
impeached	paranormal	agp	sportblogs	pka	retailing	farming
impeachment	converge	dhcp	mickelson	chemistry	mlm	volcanic
neocons	antimatter	win98	juventus	quarks	budgeting	ecosystem

Table: Top five words with the largest values in a given word-embedding dimension.

Text Sequence Matching:

Model	SNLI	MultiNLI Matched	MultiNLI Mismatched	WikiQA *MAP*	WikiQA *MRR*	Quora	MSRP *Acc.*	MSRP *F1*
	Acc.	*Acc.*	*Acc.*			*Acc.*		
CNN	82.1	65.0	65.3	0.6752	0.6890	79.60	69.9	80.9
LSTM	80.6	66.9*	66.9*	**0.6820**	**0.6988**	82.58	70.6	80.5
SWEM-*aver*	82.3	66.5	66.2	**0.6808**	**0.6922**	82.68	71.0	81.1
SWEM-*max*	**83.8**	**68.2**	**67.7**	0.6613	0.6717	82.20	70.6	80.8
SWEM-*concat*	83.3	67.9	67.6	0.6788	0.6908	**83.03**	**71.5**	**81.3**

Short Sentence Classification:

Model	MR	SST-1	SST-2	Subj	TREC
RAE (Socher et al., 2011b)	77.7	43.2	82.4	–	–
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–
LSTM (Tai et al., 2015)	–	46.4	84.9	–	–
RNN (Zhao et al., 2015)	77.2	–	–	**93.7**	90.2
Constituency Tree-LSTM (Tai et al., 2015)	-	**51.0**	88.0	-	-
Dynamic CNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0
CNN (Kim, 2014)	**81.5**	48.0	**88.1**	93.4	**93.6**
DAN-ROOT (Iyyer et al., 2015)	-	46.9	85.7	-	-
SWEM-*aver*	77.6	45.2	83.9	92.5	**92.2**
SWEM-*max*	76.9	44.1	83.6	91.2	89.0
SWEM-*concat*	**78.2**	**46.1**	**84.3**	**93.0**	91.8

The Role of Word-order Information:

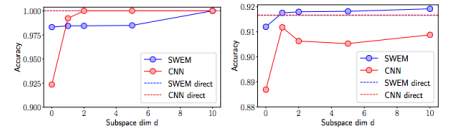
- **Removing the word-order features on the training set:**

Datasets	Yahoo	Yelp P.	SNLI
Original	72.78	95.11	78.02
Shuffled	72.89	93.49	77.68

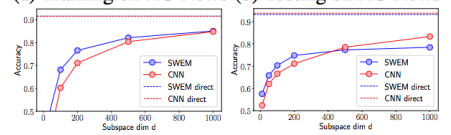
Table Test accuracy for LSTM model trained on original/shuffled training set.

The performance *does not change much* on the topic prediction and natural language inference tasks!

Comparison via subspace training:



(a) Training on AG News (b) Testing on AG News



(c) Testing on AG News (d)Testing on Yelp P.

1) Tuning word embeddings alone is enough, regardless of the model employed on top;

2) According to Occam's razor, ***simple models*** are preferred.

Table: Performance of subspace training. Word embeddings are optimized in (a)(b), and frozen in (c)(d).

Problems where deeper architectures are necessary:

Short sentence classification, sequence tagging

Conclusion

- Simple pooling operations are surprisingly effective at inferring sentence/document representations;
- The advantages of deep architectures vary from task to task;
- Other neural modules (e.g. attention or memory mechanism) can be directly applied on top of word embeddings for better representations;
- Baseline Needs More Love!

Source Code: