How Does Selective Mechanism Improve Self-Attention Networks?

Xinwei Geng[1], Longyue Wang[2], Xing Wang[2], Bing Qin[1], Ting Liu[1], Zhaopeng Tu[2]

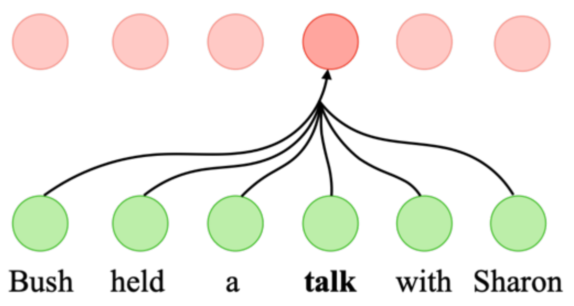
[1]Research Center for Social Computing and Information Retrieval [2]Tencent AI Lab

1. Introduction

- **Self-attention networks (SANs)** have achieved promising progress in various natural language processing tasks such as machine translation, summarization
  - The appealing strength of SANs derives from high parallelism and flexibility in modeling dependencies among the input elements
  - Towards generating sentence representations, SANs calculate the attentive output by glimpsing the entire sequence

$$\mathbf{O} = \mathrm{ATT}(\mathbf{Q},\mathbf{K})\mathbf{V}$$

$$\mathrm{ATT}(\mathbf{Q},\mathbf{K}) = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})$$



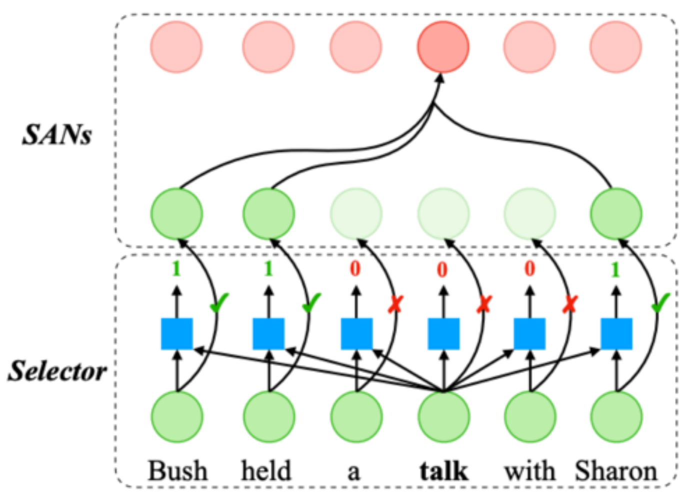
2. Our Approach

- In most case, only a subset of input elements are important to generate the sentence representations
- Towards tackling this issue, we adopt a universal and flexible implementation of selective mechanism called **selective self-attention networks (SSANs)**
- SSANs select a subset of input words by an additional selector module, on top of which self-attention networks are conducted

$$\mathbf{E} = \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}$$

$$\mathrm{ATT}_n^{se} = \frac{\mathbf{A}_n \odot \exp(\mathbf{E}_n)}{\sum_{n'=1}^{N} \mathbf{A}_{n'} \odot \exp(\mathbf{E}_{n'})}$$

$$\mathbf{O}^{se} = \mathrm{ATT}^{se}(\mathbf{A},\mathbf{Q},\mathbf{K})\mathbf{V}$$



2.1 Selector

- Parameterize selection action a ∈ {SELECT, DISCARD} for each input element with an auxiliary policy network
  - **SELECT**(1) indicates that the element is selected
  - **DISCARD**(0) represents to abandon the element

$$\mathbf{E}_p = \mathbf{Q}_p\mathbf{K}_p^T$$

$$\pi(\mathbf{A}|\mathbf{Q}_p,\mathbf{K}_p) = sigmoid(\mathbf{E}_p)$$

2.2 Gumbel Relaxation

- gumbel-sigmoid to approximate the sampling
- G' and G'' are gumbel noises
- τ is temperature parameter

$$\begin{aligned} &\textit{Gumbel-Sigmoid}(\mathbf{E}_s) \\ &= \textit{sigmoid}((\mathbf{E}_s + \mathbf{G}' - \mathbf{G}'')/\tau) \\ &= \frac{\exp((\mathbf{E}_s + \mathbf{G}')/\tau)}{\exp((\mathbf{E}_s + \mathbf{G}')/\tau) + \exp(\mathbf{G}''/\tau)} \end{aligned}$$

3. NLP Benchmarks

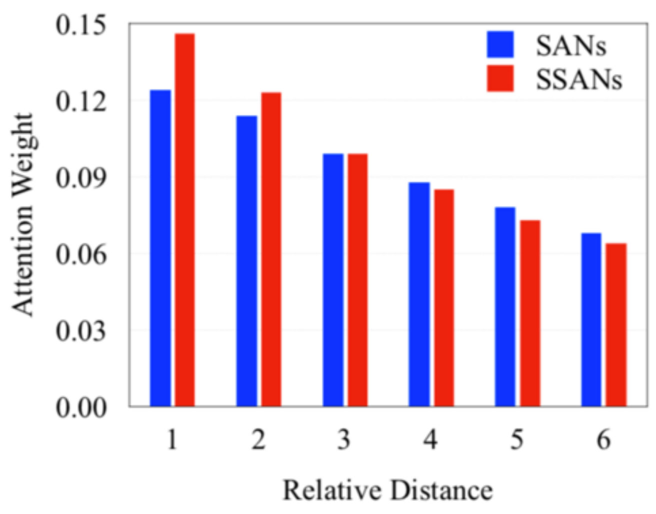
Task	Size	SANs	SSANs	△
Natural Language Inference (Accuracy)				
SNLI	550K	85.60	86.30	*+0.8%*
Semantic Role Labeling (F1 score)				
CoNLL	312K	82.48	82.88	*+0.5%*
Machine Translation (BLEU)				
En⇒Ro	0.18M	23.22	23.91	*+3.0%*
En⇒Ja	0.44M	31.56	32.17	*+1.9%*
En⇒De	4.56M	27.60	28.50	*+3.3%*

4. Evaluation of Word Order Encoding

4.1 Detection of Local Word Reordering

- Bigram order shift detection aims to test whether an encoder is sensitive to local word orders
- A certain portion of sentences are randomly extracted to construct instances with illegal word order
- e.g. *What **are you** doing out there?* => *What **you are** doing out there?*

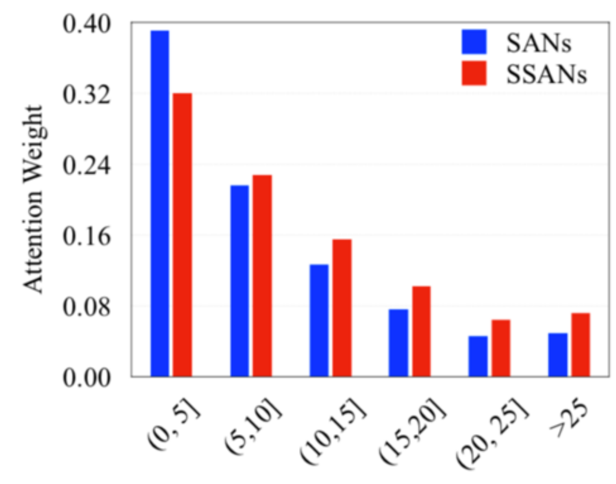
Model	Layer	Acc.	△
SANs	–	52.23	–
SSANs	1	**62.55**	*+19.8%*
	2	53.73	*+2.9%*
	3	54.65	*+4.6%*
	4	54.29	*+3.9%*
	5	54.78	*+4.9%*
	6	54.23	*+3.8%*



4.2 Detection of Global Word Reordering

- A random word is popped and inserted into another position
- The objective is to detect both the original position the word is popped out, and the position the word is inserted
- e.g. *Bush **held** a talk with Sharon.* => *Bush a talk **held** with Sharon.*

Model	Layer	Insert	Original	Both
SANs	–	73.20	66.00	60.10
SSANs	1	**81.52**	**72.19**	**66.77**
	2	80.14	70.01	63.97
	3	79.82	69.69	63.93
	4	79.08	70.22	63.67
	5	80.19	69.84	64.12
	6	80.27	69.50	63.73



5. Evaluation of Syntactic Structure Modeling

5.1 Structures Embedded in Representations

- *Tree Depth* check whether examined model can group sentences by depth of longest path from root to any leaf

Class	Ratio	SANs	SSANs	△
5	6.9%	68.66	75.22	*+9.6%*
6	14.3%	56.10	64.09	*+14.2%*
7	16.3%	46.63	55.05	*+18.1%*
8	17.9%	39.68	50.88	*+28.2%*
9	17.4%	38.33	50.97	*+33.0%*
10	15.3%	35.54	49.88	*+40.3%*
11	11.9%	48.86	56.39	*+15.4%*
All	100%	45.68	55.90	*+22.4%*

- *Top Constituent* classify sentence in terms of sequence of top constituents immediately below the root node

Type	Ratio	SANs	SSANs	△
Ques.	10%	95.90	97.06	+1.2%
Decl.	60%	88.48	91.34	+3.2%
Clau.	25%	72.78	78.32	+7.6%
Other	5%	50.67	61.13	+20.6%
All	100%	83.78	87.25	*+4.1%*

5.2 Structures Modeled by Attention

- Constructing constituency trees from the attention distributions
- Attention distribution within phrases is stronger than the other
- When splitting a phrase with span (i, j), the target is to look for a position k maximizing the scores of the two resulting phrases
- Utilize Stanford CoreNLP toolkit to annotate English sentences as golden constituency trees

$$k = \arg\max_{k'}(\mathrm{score}(i,k') \cdot \mathrm{score}(k',j))$$

Metric	SANs	SSANs	△
BP	21.09	22.07	*+4.7%*
BR	22.05	23.07	*+4.6%*
F1	21.56	22.56	*+4.2%*

6. Analysis on Linguistic Properties

Type		TreeDepth			TopConst			En⇒De Translation		
		SANs	*SSANs*	△	*SANs*	*SSANs*	△	*SANs*	*SSANs*	△
Content	Noun	0.149	0.245	*+64.4%*	0.126	0.196	*+55.6%*	0.418	0.689	*+64.8%*
	Verb	0.165	0.190	*+15.2%*	0.165	0.201	*+21.8%*	0.146	0.126	*-13.7%*
	Adj.	0.040	0.069	*+7.3%*	0.033	0.054	*+63.6%*	0.077	0.074	*-3.9%*
	Total	0.354	0.504	*+42.4%*	0.324	0.451	*+39.2%*	0.641	0.889	*+38.7%*
Content-Free	Prep.	0.135	0.082	*-39.3%*	0.123	0.119	*-3.3%*	0.089	0.032	*-64.0%*
	Dete.	0.180	0.122	*-32.2%*	0.103	0.073	*-29.1%*	0.070	0.010	*-85.7%*
	Punc.	0.073	0.068	*-6.8%*	0.078	0.072	*-7.7%*	0.098	0.013	*-86.7%*
	Others	0.258	0.224	*-13.2%*	0.373	0.286	*-23.3%*	0.102	0.057	*-41.1%*
	Total	0.646	0.496	*-23.3%*	0.676	0.549	*-18.8%*	0.359	0.111	*-69.1%*