Von Mises-Fisher Loss For Training Sequence To Sequence Models With Continuous Outputs

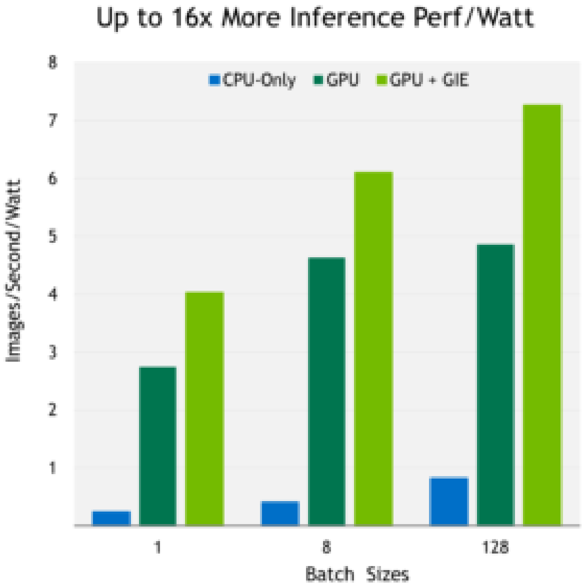
ICLR 2019

Kunmar et al.

Presenter: Jungsoo Park

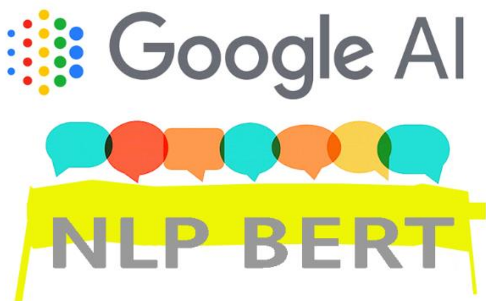
Data Mining & Information Systems Lab.
Department of Computer Science and Engineering,
College of Informatics, Korea University

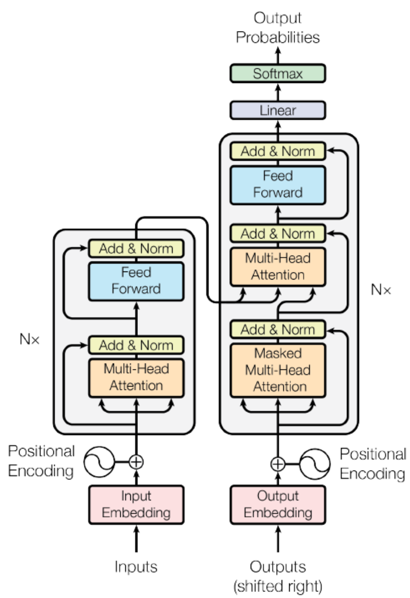
1 Introduction



GPUs have enabled the Deep Learning Accessible

1 Introduction





Pre-Training took 4 days using

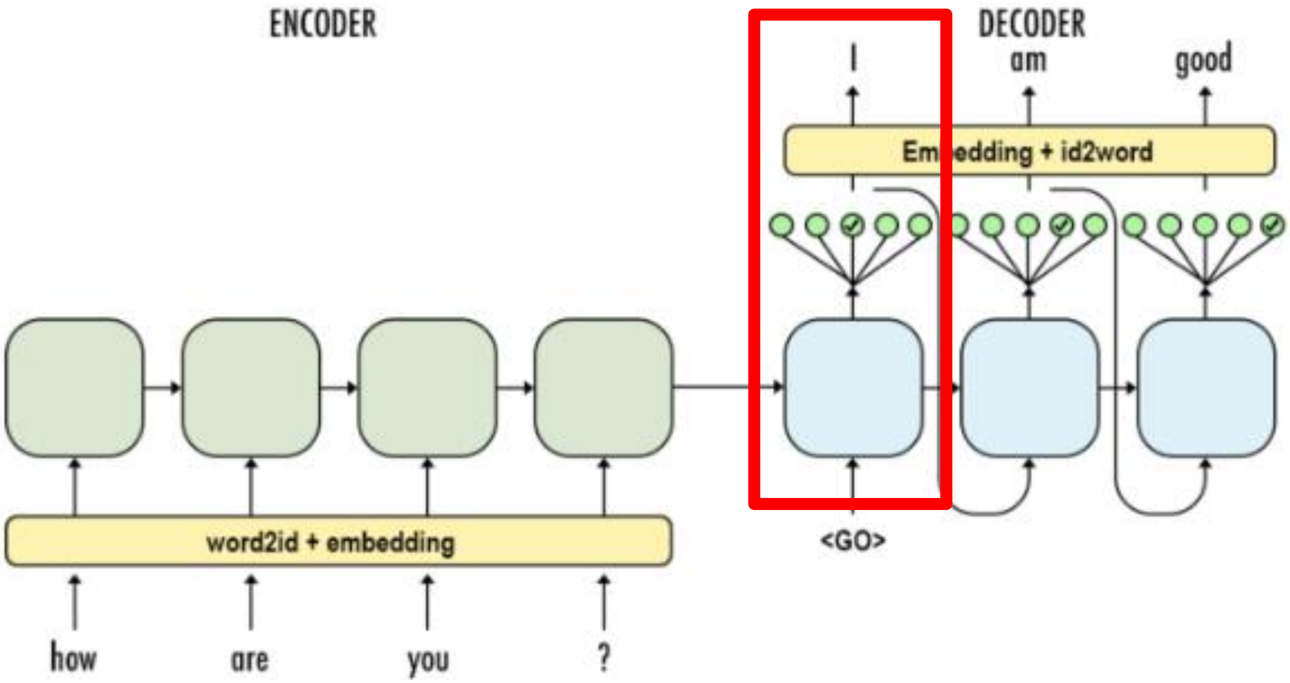
- BERT-Base: 4 Cloud TPUs (16 TPU chips total)
- BERT-Large: 16 Cloud TPUs (64 TPU chips total)

If using 8 TESLA P100,
it would take one year in training

Time Matters in Training

2 Recap

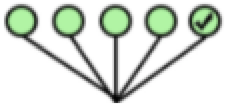
Seq2Seq



2 Recap

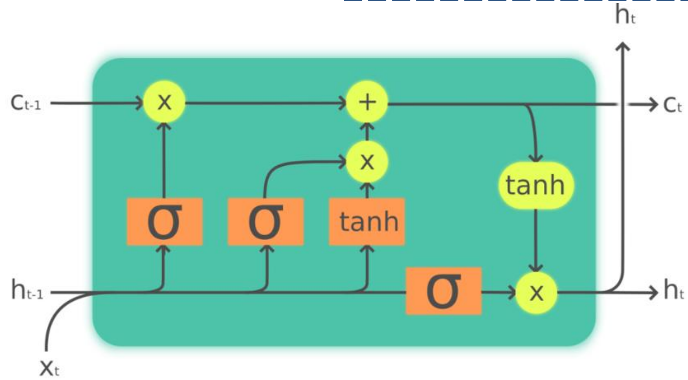
Word Generation

Output of Vocabulary Size



H -> V

Hidden State



$$s_w = W_{hw}\mathbf{h}_t + b_w$$

$$W \in \mathbb{R}^{V\text{x}H}$$

$$b \in \mathbb{R}^{v}$$

Main Bottleneck in Training

$$\boxed{\mathbf{p}_t(w) = \frac{e^{s_w}}{\boxed{\sum_{v \in \mathcal{V}} e^{s_v}}}}$$

Loss

$$\mathrm{NLL}(\mathbf{p_t}, \mathbf{o}(w)) = -\log(\mathbf{p_t}(w))$$

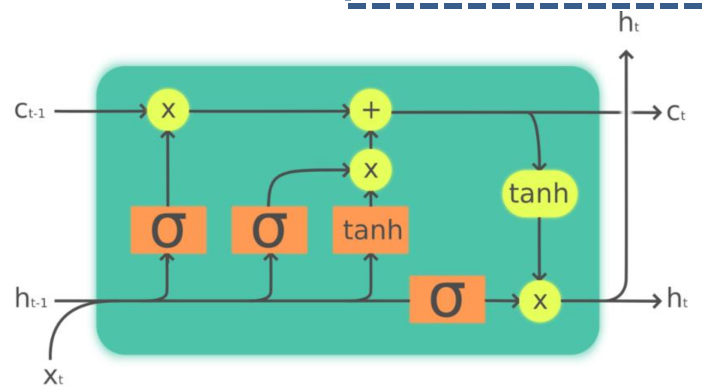
3 Method

Word Generation

Embedding Vector
(M << V)

H -> M

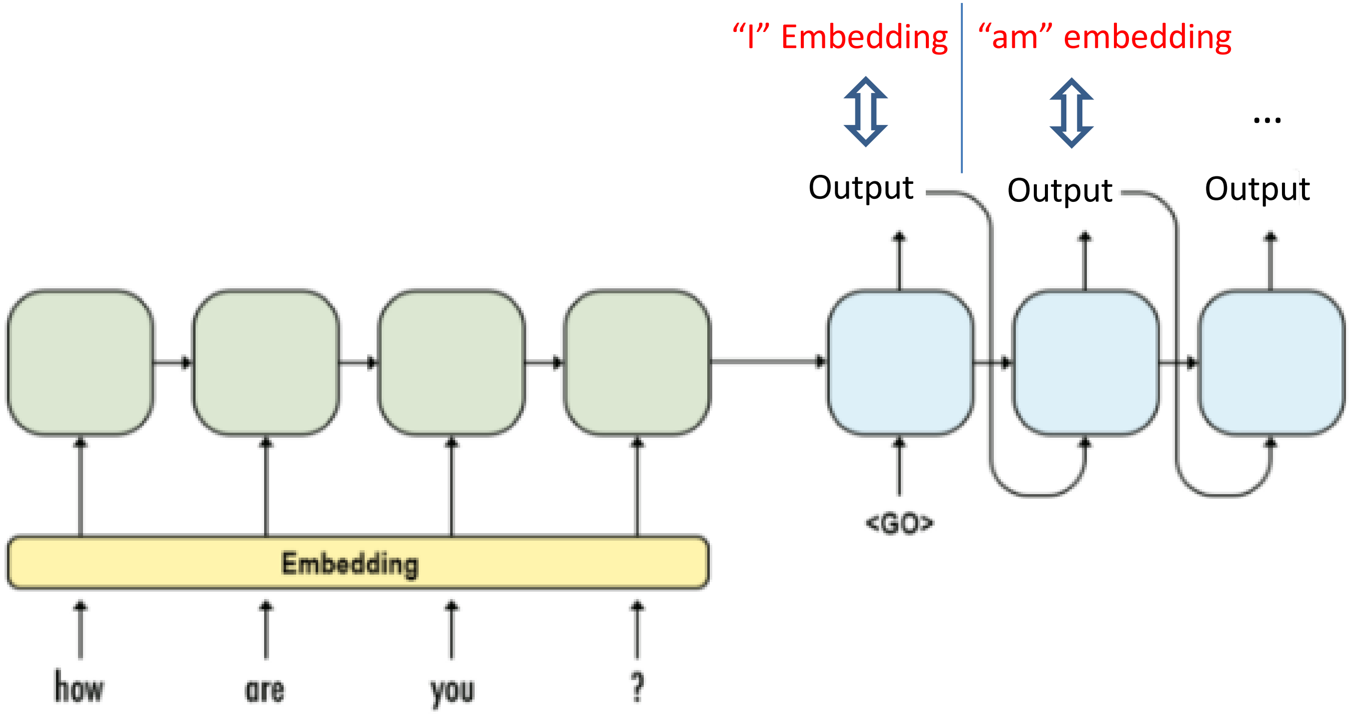
Hidden State



No need to do soft-max operation, but only find out the **nearest neighbor word in the word embedding** space

3 Method

Loss occurs from the target word(label's) embedding vector and our generated one's



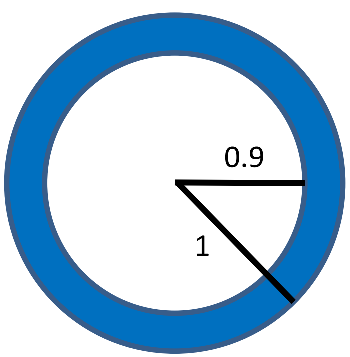
3 Method

But what loss function should we use?

L2 Loss?

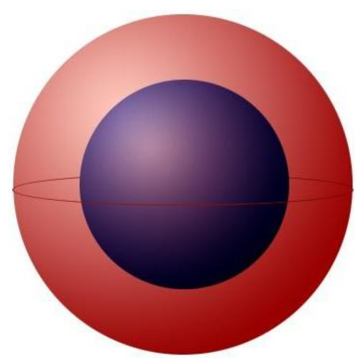
Basic assumption behind using L2 loss is that
the output space follows Gaussian distribution
and is a Distance-based metric

2D



1 – (0.9)^2

3D



1 – (0.9)^3

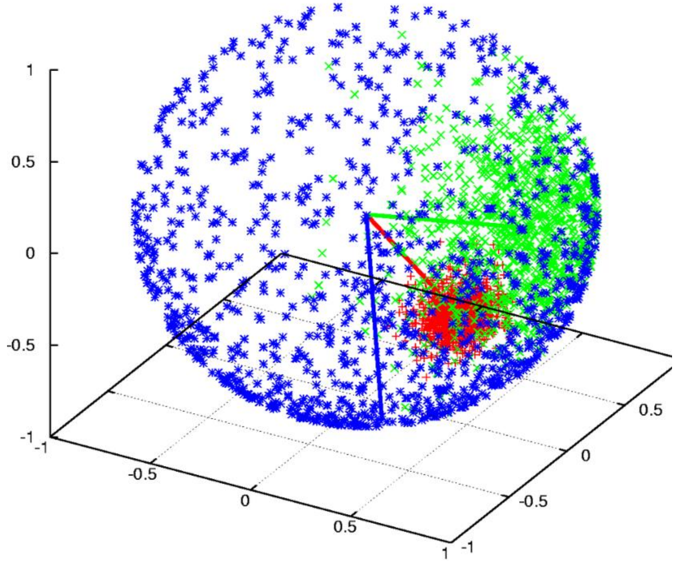
nD

1 – (0.9)^n

as n goes higher, the data will placed in the surface of a unit sphere

3 Method

Therefore, it's not the best way to use distance metric,
but instead use **direction based metric** for high dimension data



Von Mises-Fisher

Vector close to mean direction will have high probability
(directional equivalent of Gaussian)

$$p(\mathbf{e}(w); \boldsymbol{\mu}, \kappa) = C_m(\kappa) e^{\kappa \boldsymbol{\mu}^T \mathbf{e}(w)},$$

$$C_m(\kappa) = \frac{\kappa^{m/2-1}}{(2\pi)^{m/2} I_{m/2-1}(\kappa)},$$

$$p(\mathbf{e}(w); \hat{\mathbf{e}}) = \text{vMF}(\mathbf{e}(w); \hat{\mathbf{e}}) = C_m(\|\hat{\mathbf{e}}\|) e^{\hat{\mathbf{e}}^T \mathbf{e}(w)}$$

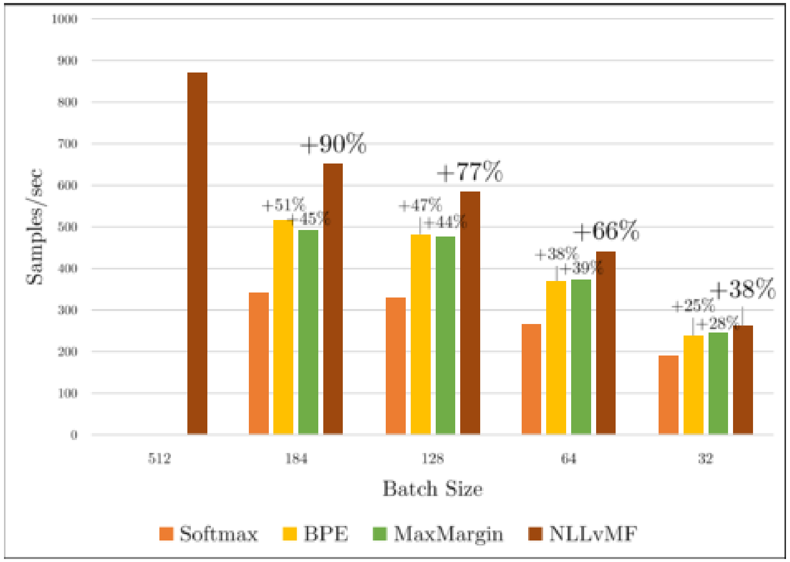
$$\boxed{\text{NLLvMF}(\hat{\mathbf{e}}; \mathbf{e}(w)) = -\log\left(C_m(\|\hat{\mathbf{e}}\|)\right) - \hat{\mathbf{e}}^T \mathbf{e}(w)}$$

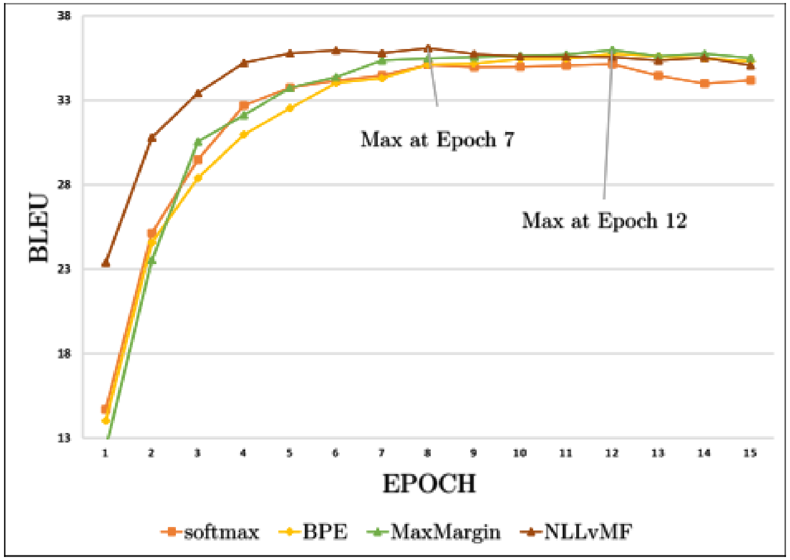
4 Experiment

Embedding Model	Tied Emb	Source Type/ Target Type	Loss	BLEU		
				fr–en	de–en	en–fr
-	no	word→word	CE	31.0	24.7	29.3
-	no	word→BPE	CE	29.1	24.1	29.8
-	no	BPE→BPE	CE	**31.4**	**25.8**	**31.0**
word2vec	no	word→emb	L2	27.2	19.4	26.4
word2vec	no	word→emb	Cosine	29.1	21.9	26.6
word2vec	no	word→emb	MaxMargin	29.6	21.4	26.7
fasttext	no	word→emb	MaxMargin	31.0	25.0	29.0
fasttext	yes	word→emb	MaxMargin	**32.1**	**25.0**	**31.0**
word2vec	no	word→emb	$\mathrm{NLLvMF}_{\mathrm{reg1}}$	29.5	22.7	26.6
word2vec	no	word→emb	$\mathrm{NLLvMF}_{\mathrm{reg1+reg2}}$	29.7	21.6	26.7
word2vec	yes	word→emb	$\mathrm{NLLvMF}_{\mathrm{reg1+reg2}}$	29.7	22.2	27.5
fasttext	no	word→emb	$\mathrm{NLLvMF}_{\mathrm{reg1+reg2}}$	30.4	23.4	27.6
fasttext	yes	word→emb	$\mathrm{NLLvMF}_{\mathrm{reg1+reg2}}$	**32.1**	**25.1**	**31.7**

- x2.5 times faster than training word -> word (baseline) but shows comparable result
- Proposed Loss results in better result than the empirical loss functions(L2, Cosine Loss, Max Margin Loss)

4 Experiment





Samples processed per second
(Proposed method outperforms)

Convergence
(NLLvMF Loss outperforms in stability)