# Distilling Task-Specific Knowledge from BERT into Simple Neural Networks

RAPHAEL TANG, YAO LU, LINQING LIU, LILI MOU, OLGA VECHTOMOVA, JIMMY LIN

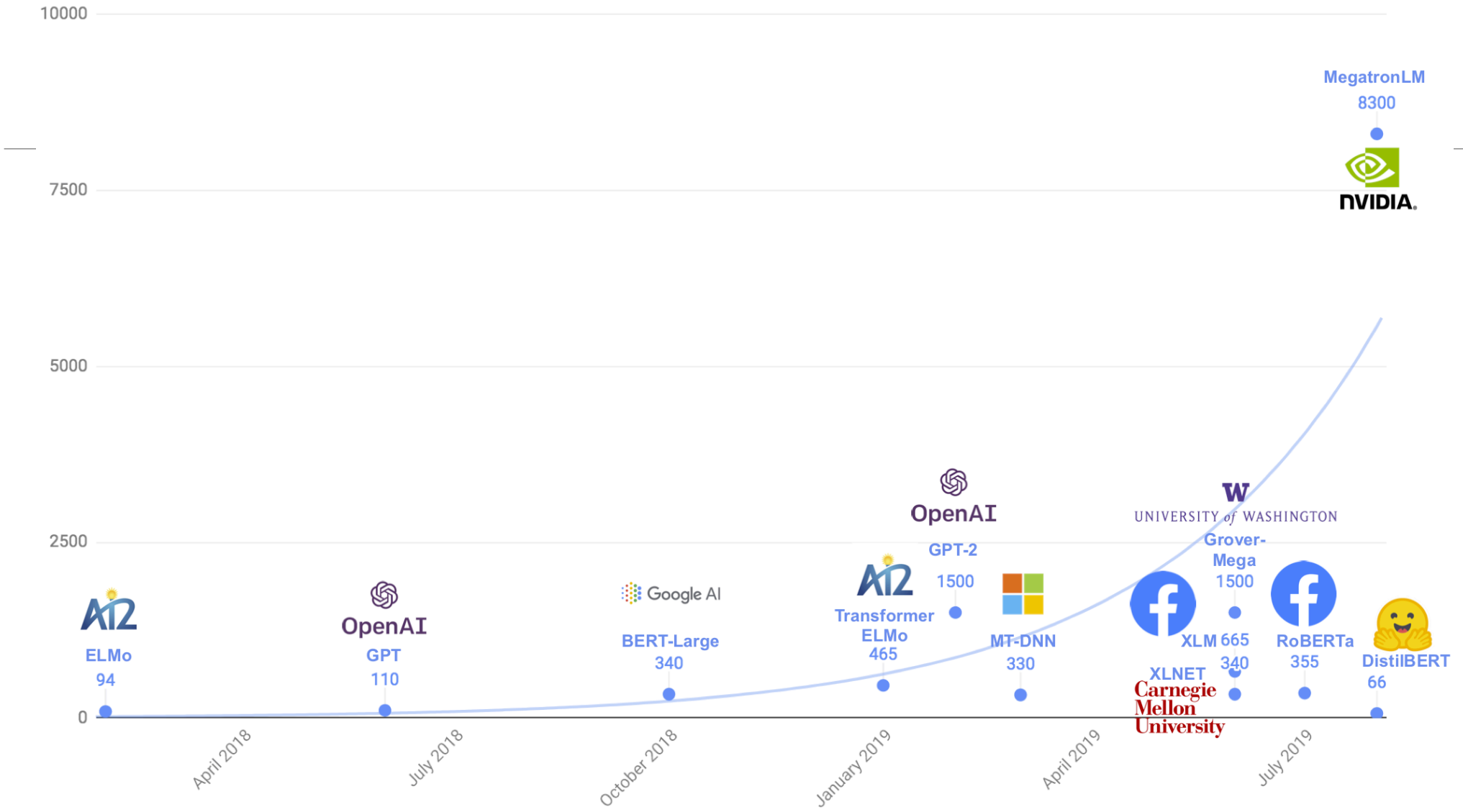
PRESENTER: CHIYU ZAHNG

# Content

- Context
- BERT
- Model Architecture
- Distillation
- Data Augmentation
- Experiments
- References

# Context

Due to ***the large number of parameters***, BERT and GPT-2, for example, are undeployable in resource-restricted systems such as ***mobile devices***. They may ***be inapplicable in real-time systems*** either, because of ***low inference-time efficiency***.


10000
7500
5000
2500
0
MegatronLM
8300
NVIDIA.
OpenAI
GPT-2
1500
UNIVERSITY of WASHINGTON
Grover-
Mega
1500
ELMo
94
OpenAI
GPT
110
Google AI
BERT-Large
340
Transformer
ELMo
465
MT-DNN
330
XLM 665
340
XLNET
Carnegie
Mellon
University
RoBERTa
355
DistilBERT
66
April 2018
July 2018
October 2018
January 2019
April 2019
July 2019

# Context

Due to ***the large number of parameters***, BERT and GPT-2, for example, are undeployable in resource-restricted systems such as ***mobile devices***. They may ***be inapplicable in real-time systems*** either, because of ***low inference-time efficiency***.

Question:

**How should we put these monsters in production?**

**How can we use such large models under low latency constraints?**

**This paper** wishes to study effective approaches to transfer knowledge from ***BERT to a BiLSTM***.

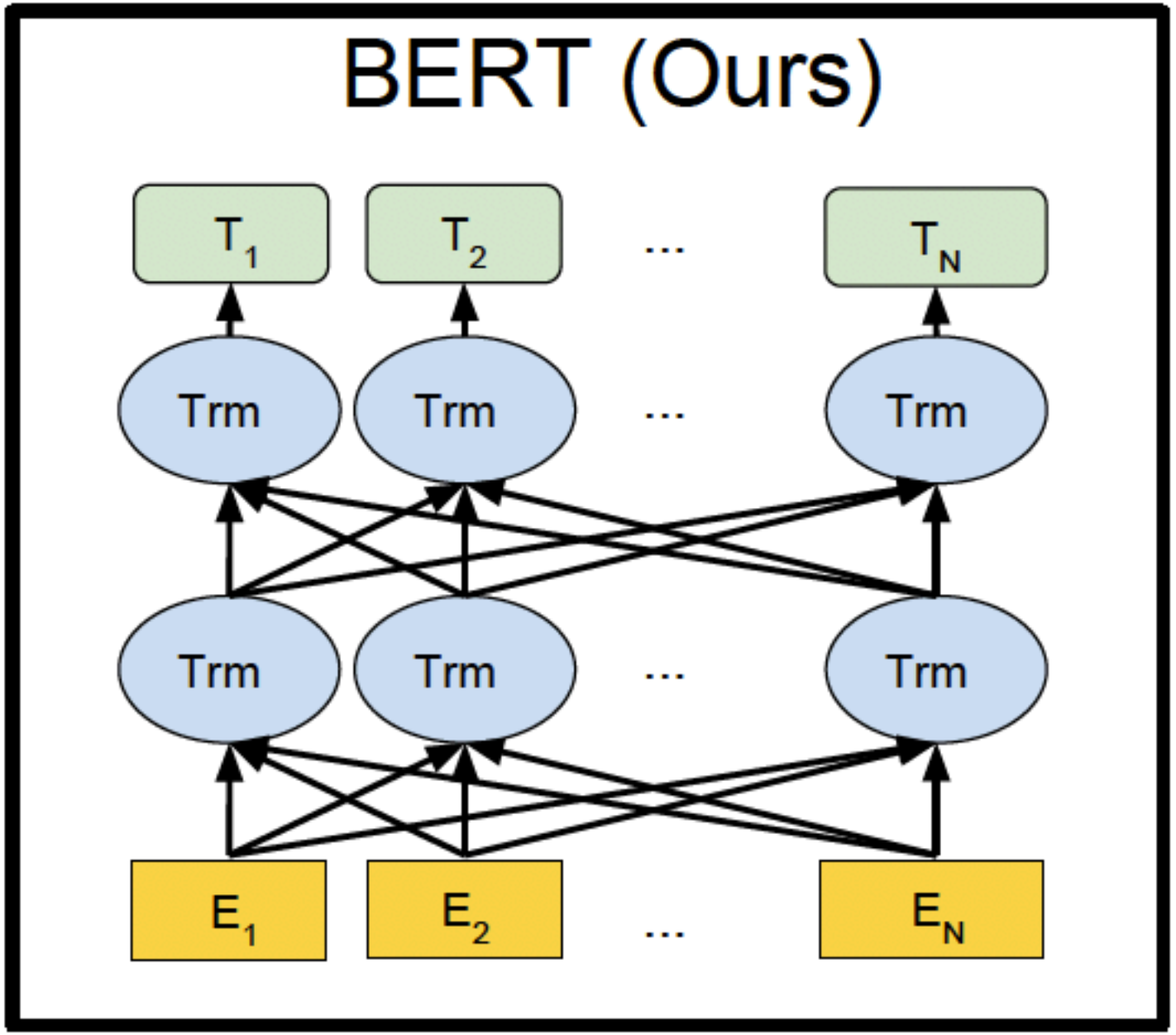
# BERT

Bidirectional Encoder Representations from Transformers.

- **Bidirectional**: BERT is naturally bidirectional model.
- **Generalizable**: Pre-trained BERT model can be finetuned easily for downstream NLP task
- **High-Performance**: Fine-tuned BERT models beat state-of-the-art results for many NLP tasks
- **Universal**: Trained on Wikipedia + BookCorpus. No special dataset needed

# BERT

BERT representations are jointly conditioned on both left and right context in all layers.



# Pre-trained Model

- `BERT-Base, Uncased` : 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Large, Uncased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Cased` : 12-layer, 768-hidden, 12-heads , 110M parameters
- `BERT-Large, Cased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Multilingual Cased (New, recommended)` : 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Multilingual Uncased (Orig, not recommended)` **(Not recommended, use** `Multilingual Cased` **instead)**: 102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Chinese` : Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters

# Classification Tasks

BERT was fine-tuned for **11 NLP tasks.**

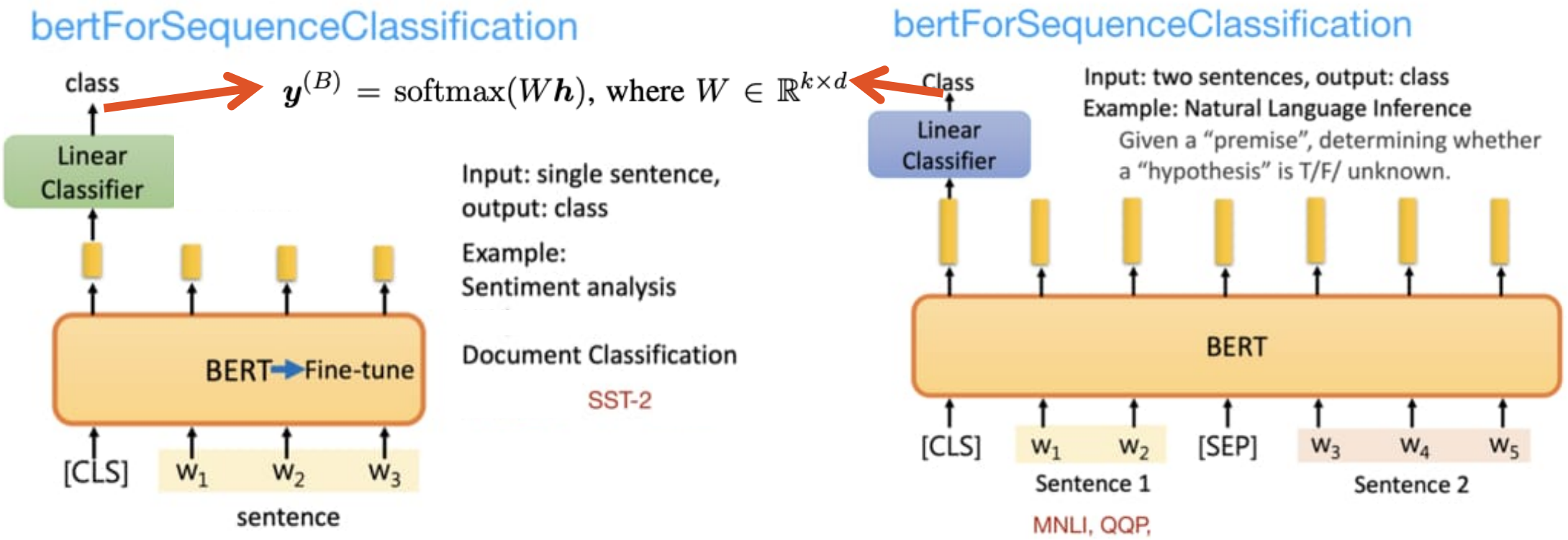
Due to restrictions in time and computational resources, they **choose the three** most widely used dataset.

- The General Language Understanding Evaluation **(GLUE)** benchmark (Wang et al., 2018)

1. **MNLI** Multi-Genre Natural Language Inference: Given ***a pair of sentences***, to predict whether the second sentence is an *entailment, contradiction, or neutral* with respect to the first one.
2. **QQP** Quora ***Question Pairs is a binary classification task*** where the goal is to determine if two questions asked on Quora are semantically equivalent.
3. **SST-2** The Stanford Sentiment Treebank: ***a binary single-sentence*** sentimental classification task (positive/negative).

# Model Architecture

# Task-Specific Models Using BERT



# BiLSTM model for single-sentence classification.

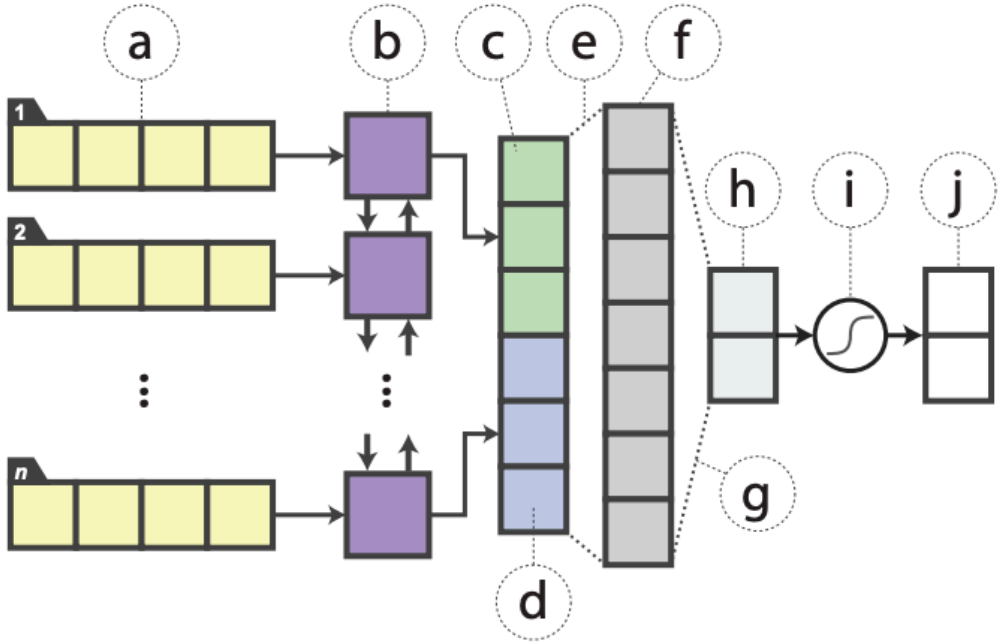



Figure 1: The BiLSTM model for single-sentence classification. The labels are **(a)** input embeddings, **(b)** BiLSTM, **(c, d)** backward and forward hidden states, respectively, **(e, g)** fully-connected layer; **(e)** with ReLU, **(f)** hidden representation, **(h)** logit outputs, **(i)** softmax activation, and **(j)** final probabilities.

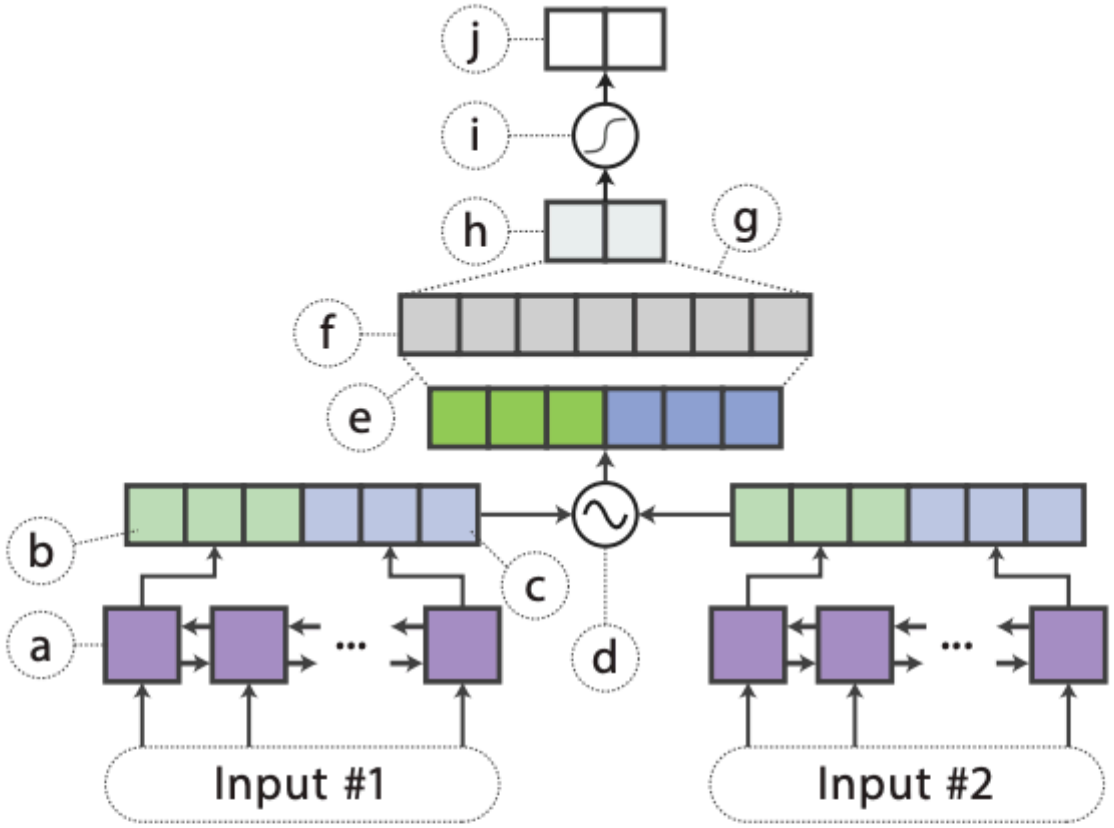
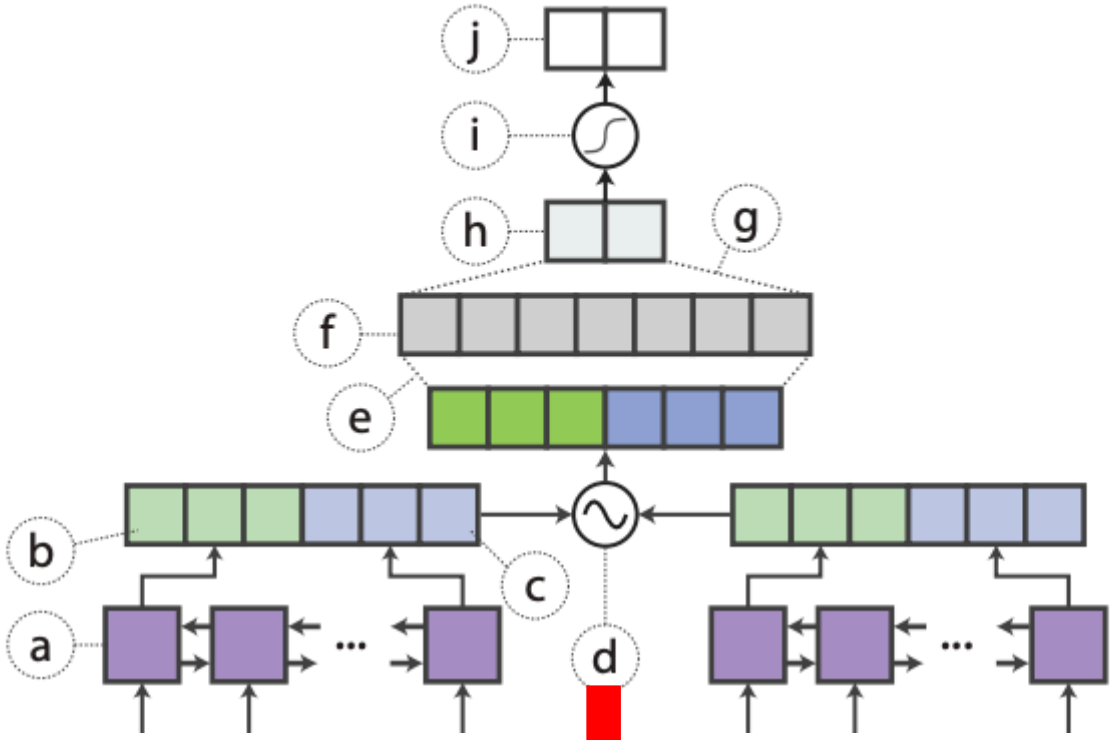



Figure 2: The siamese BiLSTM model for sentence matching, with shared encoder weights for both sentences. The labels are **(a)** BiLSTM, **(b, c)** final backward and forward hidden states, respectively, **(d)** concatenate–compare unit, **(e, g)** fully connected layer; **(e)** with ReLU, **(f)** hidden representation, **(h)** logit outputs, **(i)** softmax activation, and **(j)** final probabilities.

# BiLSTM model for sentence matching




$$f(h_{s1}, h_{s2}) = [h_{s1}, h_{s2}, h_{s1} \odot h_{s2}, |h_{s1} - h_{s2}|]$$

Figure 2: The siamese BiLSTM model for sentence matching, with shared encoder weights for both sentences. The labels are **(a)** BiLSTM, **(b, c)** final backward and forward hidden states, respectively, **(d)** concatenate–compare unit, **(e, g)** fully connected layer; **(e)** with ReLU, **(f)** hidden representation, **(h)** logit outputs, **(i)** softmax activation, and **(j)** final probabilities.

# BiLSTM model for sentence matching

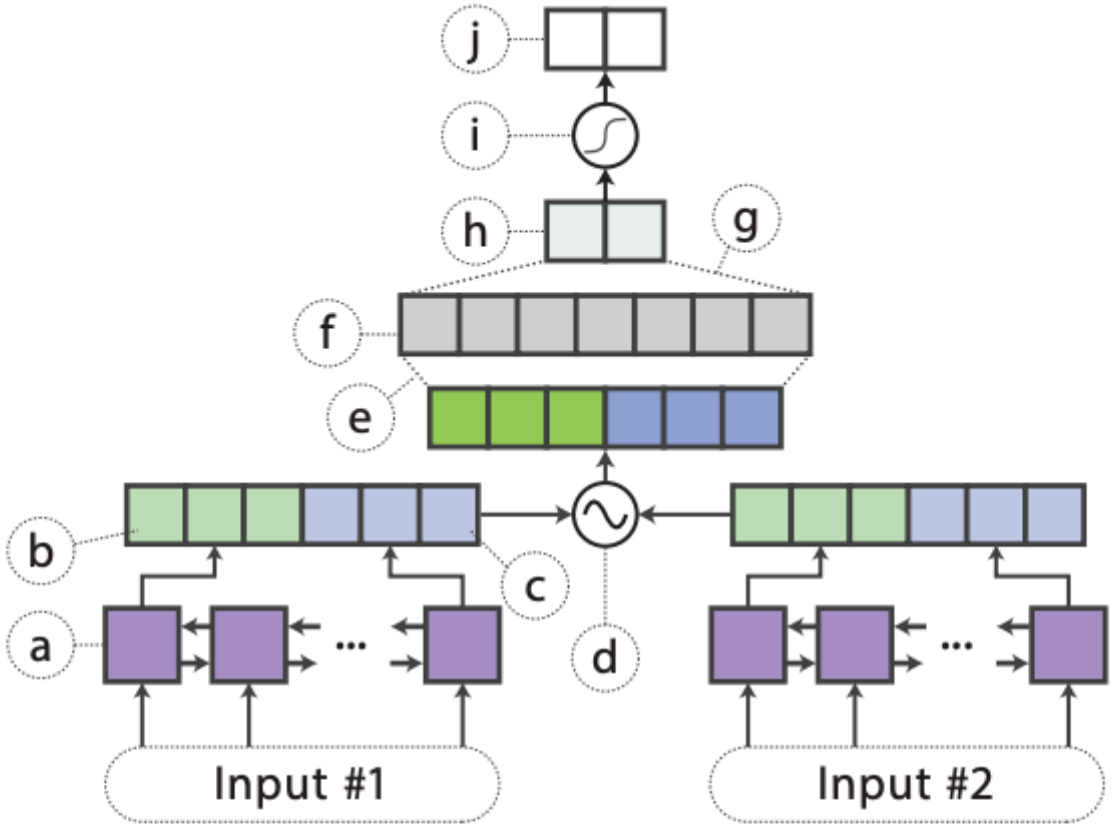



Figure 2: The siamese BiLSTM model for sentence matching, with shared encoder weights for both sentences. The labels are **(a)** BiLSTM, **(b, c)** final backward and forward hidden states, respectively, **(d)** concatenate–compare unit, **(e, g)** fully connected layer; **(e)** with ReLU, **(f)** hidden representation, **(h)** logit outputs, **(i)** softmax activation, and **(j)** final probabilities.

# BiLSTM model for sentence matching

They restrict the architecture engineering to a ***minimum*** to revisit the representation power of ***BiLSTM itself.*** They avoid any additional tricks, such as ***attention and layer normalization***.

How can we transfer knowledge from ***BERT to a BiLSTM?***

# Distillation

The ***distillation approach*** accomplishes ***knowledge transfer at the output level***; that is, the ***student network*** learns to mimic a ***teacher network***'s behavior given any data point ( "***teacher-student" learning).***

***In supervised learning***, a classification model is generally trained to predict ***a gold class*** by maximizing its ***probability (softmax of logits)***. In many cases, a good performance model will predict an output distribution with the ***correct class having a high probability***, leaving other classes with **probabilities near zero**.

**But, some of these "almost-zero" probabilities are larger than the others, and this reflects, in part, the generalization capabilities of the model.**

***In supervised learning***, a classification model is generally trained to predict ***a gold class*** by maximizing its ***probability (softmax of logits)*** using the log-likelihood signal. In many cases, a good performance model will predict an output distribution with the correct class having a high probability, leaving other classes with **probabilities near zero**.

**But, some of these "almost-zero" probabilities are larger than the others, and this reflects, in part, the generalization capabilities of the model.**

**For instance,** a *desk chair* might be mistaken with an *armchair* but should usually not be mistaken with a *mushroom*.

This uncertainty is sometimes referred to as the **"dark knowledge"**

# Copy The Dark Knowledge

In the **teacher-student training**, we train a student network to mimic the **full output distribution** of the teacher network (its knowledge).

In this paper, they follow works of Ba and Caruana (2014).

# Distillation

The discrete probability output of a neural network is given by:

$$\widetilde{y}_i = \text{softmax}(\boldsymbol{z}) = \frac{\exp\{\boldsymbol{w}_i^\top \boldsymbol{h}\}}{\sum_j \exp\{\boldsymbol{w}_j^\top \boldsymbol{h}\}}$$

where $\boldsymbol{w}_i$ denotes the $i^{\text{th}}$ row of softmax weight $W$, and $\boldsymbol{z}$ is equivalent to $\boldsymbol{w}^\top \boldsymbol{h}$. The argument of the softmax function is known as *logits*.

# Ba and Caruana (2014) state that:

By training the student model directly ***on the logits***, the student is better able to learn ***the internal model learned*** by the teacher, without suffering from ***the information loss*** that occurs from passing through logits to probability space.

Ba, J., & Caruana, R. (2014). Do deep nets really need to be deep?. In *Advances in neural information processing systems* (pp. 2654-2662).

# Example

**Logit**

X_1[10,20,30]

X_2[-10,0,10]

***Softmax(x)***

**Probability**

$[2*10^{-9},4*10^{-5},0.9999]$

# Distillation

$$\widetilde{y}_i = \text{softmax}(\boldsymbol{z}) = \frac{\exp\{\boldsymbol{w}_i^\top \boldsymbol{h}\}}{\sum_j \exp\{\boldsymbol{w}_j^\top \boldsymbol{h}\}}$$

The ***distillation objective*** is to minimize the ***mean-squared-error (MSE)*** loss between the ***student network's logits*** against the ***teacher's logits***:

$$\mathcal{L}_{\text{distill}} = ||\boldsymbol{z}^{(B)} - \boldsymbol{z}^{(S)}||_2^2$$

At training time, the distilling objective can be used in ***conjunction with a traditional cross entropy loss*** against a one-hot label *t*.

$$\begin{aligned}\mathcal{L} &= \alpha \cdot \mathcal{L}_{\text{CE}} + (1-\alpha) \cdot \mathcal{L}_{\text{distill}} \\ &= -\alpha \sum_i t_i \log y_i^{(S)} - (1-\alpha)||\boldsymbol{z}^{(B)} - \boldsymbol{z}^{(S)}||_2^2\end{aligned}$$

# Distillation

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Softmax with Temperature

High temperatures (***T -> inf***), all actions have nearly the **same probability** (softer).

Low temperature (***T -> 0)***, the probability of the action with the **highest expected** reward tends to **1**.

**The first objective function** is the ***cross-entropy (or Kullback–Leibler divergence)*** with the **soft targets** and this cross entropy is computed using the ***same temperature*** in the **softmax of the student model** as was used for generating the soft targets from the **teacher model**.

**The second objective function** is the **cross-entropy** with **the correct labels** and this is computed using exactly the same logits in softmax of the **student model** but at a temperature of 1.

# Distillation

The ***distillation objective*** is to minimize the ***mean-squared-error (MSE)*** loss between the ***student network's logits*** against the ***teacher's logits***:

$$\mathcal{L}_{\text{distill}} = ||\boldsymbol{z}^{(B)} - \boldsymbol{z}^{(S)}||_2^2$$

At training time, the distilling objective can be used in ***conjunction with a traditional cross entropy loss*** against a one-hot label *t*.

$$\begin{aligned}\mathcal{L} &= \alpha \cdot \mathcal{L}_{\text{CE}} + (1-\alpha) \cdot \mathcal{L}_{\text{distill}} \\ &= -\alpha \sum_i t_i \log y_i^{(S)} - (1-\alpha)||\boldsymbol{z}^{(B)} - \boldsymbol{z}^{(S)}||_2^2\end{aligned}$$

# Data Augmentation for Distillation

In the distillation approach, a small dataset may not suffice for the teacher model to fully express its knowledge.

Therefore, they augment the training set with a large, ***unlabeled dataset, with pseudo-labels provided by the teacher***, to aid in effective knowledge distillation.

# Data Augmentation

**Masking**

With probability $p_{mask}$, they randomly replace a word with [MASK]

I loved the comedy. → I [MASK] the comedy.

**POS-guided word replacement**

With probability $p_{pos}$, they replace a word with another of the same POS tag.

What do pigs eat? → How do pigs eat?

***n*-gram sampling**

With probability $p_{ng}$, we randomly sample **an *n*-gram** from the example, where n is randomly selected from $\{1, 2, \ldots, 5\}$.

# Data Augmentation Procedure

- Given a training example $\{w_1, \ldots w_n\}$, they iterate over the words, drawing from the uniform distribution $X_i \sim UNIFORM[0, 1]$ for each $w_i$.
- If $X_i < p_{mask}$, they apply **masking** to $w_i$
- If $p_{mask} \leq X_i < p_{mask} + p_{pos}$, they apply **POS-guided word replacement.**
- They treat masking and POS-guided swapping as ***mutually exclusive***.
- After iterating through the words, with probability $p_{ng}$, they apply $n$-gram sampling to this entire synthetic example.
- They apply this procedure $n_{iter}$ times per example to generate up to $n_{iter}$ samples from ***a single example***, with any duplicates discarded.
- For ***sentence pair*** datasets, we cycle through augmenting the **first sentence only** (holding the second fixed), **the second sentence only** (holding the first fixed), and **both sentences**.

# Data Augmentation Procedure

- $p_{mask} = p_{pos} = 0.1$ and $p_{ng} = 0.25$ across all datasets.
- $n_{iter} = 20$ for SST-2
- $n_{iter} = 10$ for both MNLI and QQP, since they are larger.

# Experiments

They use the pre-trained BERT_LARGE as the teacher network.

BERT_LARGE: 24-layer, 1024-hidden, 16-heads, 340M parameters.

Fine-tune four models using the Adam optimizer with ***learning rates*** $\{2, 3, 4, 5\} \times 10^{-5}$ , picking the best model on the validation set.

# Experiments

With their best BERT model:

They feed the ***original*** dataset together with the ***generated*** examples to the task specific, fine-tuned BERT model to obtain the **predicted logits**. They denote their distilled BiLSTM trained on logit targets as $\text{BiLSTM}_{\text{SOFT}}$, which corresponds to choosing $\alpha = 0$.

# Experiments

With their best BERT model:

They feed the ***original*** dataset together with the ***generated*** examples to the task specific, fine-tuned BERT model to obtain the **predicted logits**. They denote their distilled BiLSTM trained on logit targets as $\text{BiLSTM}_{\text{SOFT}}$, which corresponds to choosing $\alpha = 0$.

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{CE}} + (1 - \alpha) \cdot \mathcal{L}_{\text{distill}} \qquad (3)$$

$$= -\alpha \sum_i t_i \log y_i^{(S)} - (1 - \alpha) ||\boldsymbol{z}^{(B)} - \boldsymbol{z}^{(S)}||_2^2$$

# Hyperparameters

They choose either ***150 or 300 hidden units*** for the BiLSTM, **and *200 or 400* units** in the ReLU activated hidden layer, depending on the validation set performance.

They use the traditional ***300-dimensional word2vec*** **embeddings trained on Google News** and ***multichannel*** **embeddings.**

They use AdaDelta with its default ***learning rate of 1.0 and $\rho = 0.95$***.

For SST-2, they use a ***batch size of 50***; for MNLI and QQP, due to their larger size, they choose ***256 for the batch size***.

# Results

| # | Model | SST-2 | QQP | MNLI-m | MNLI-mm |
|---|---|---|---|---|---|
| | | Acc | $F_1$/Acc | Acc | Acc |
| 1 | $BERT_{LARGE}$ (Devlin et al., 2018) | 94.9 | 72.1/89.3 | 86.7 | 85.9 |
| 2 | $BERT_{BASE}$ (Devlin et al., 2018) | 93.5 | 71.2/89.2 | 84.6 | 83.4 |
| 3 | OpenAI GPT (Radford et al., 2018) | 91.3 | 70.3/88.5 | 82.1 | 81.4 |
| 4 | BERT ELMo baseline (Devlin et al., 2018) | 90.4 | 64.8/84.7 | 76.4 | 76.1 |
| 5 | GLUE ELMo baseline (Wang et al., 2018) | 90.4 | 63.1/84.3 | 74.1 | 74.5 |
| 6 | Distilled $BiLSTM_{SOFT}$ | **90.7** | **68.2/88.1** | **73.0** | **72.6** |
| 7 | BiLSTM (our implementation) | 86.7 | 63.7/86.2 | 68.7 | 68.3 |
| 8 | BiLSTM (reported by GLUE) | 85.9 | 61.4/81.7 | 70.3 | 70.8 |
| 9 | BiLSTM (reported by other papers) | 87.6† | – /82.6‡ | 66.9* | 66.9* |

Table 1: Test results on different datasets. The BiLSTM results reported by other papers are drawn from Zhou et al. (2016),† Wang et al. (2017),‡ and Williams et al. (2017).* All of our test results are obtained from the GLUE benchmark website.

# Results

| | # of Par. | Inference Time |
|---|---|---|
| $\text{BERT}_{\text{LARGE}}$ | 335 (349×) | 1060 (434×) |
| ELMo | 93.6 (98×) | 36.71 (15×) |
| $\text{BiLSTM}_{\text{SOFT}}$ | 0.96 (1×) | 2.44 (1×) |

Table 2: Single-sentence model size and inference speed on SST-2. # of Par. denotes number of millions of parameters, and inference time is in seconds.

Trained on a single NVIDIA V100 GPU, we perform model inference with a batch size of 512 on all 67350 sentences of the SST-2 training set.

# Conclusion and Future Work

The distilled model achieves comparable results with ELMo, while using much ***fewer parameters and less inference time***.

To explore extremely ***simple architectures*** in the extreme, such as convolutional neural networks and even support vector machines and logistic regression

To explore ***slightly more complicated architectures*** using tricks like pairwise word interaction and attention.

# Reference


Ba, J., & Caruana, R. (2014). Do deep nets really need to be deep?. In *Advances in neural information processing systems* (pp. 2654-2662).

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Tang, R., Lu, Y., Liu, L., Mou, L., Vechtomova, O., & Lin, J. (2019). Distilling Task-Specific Knowledge from BERT into Simple Neural Networks. *arXiv preprint arXiv:1903.12136*.

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., & Anandkumar, A. (2018). Born again neural networks. *arXiv preprint arXiv:1805.04770*.

https://medium.com/huggingface/distilbert-8cf3380435b5

https://medium.com/neuralmachine/knowledge-distillation-dc241d7c2322

https://github.com/huggingface/pytorch-transformers