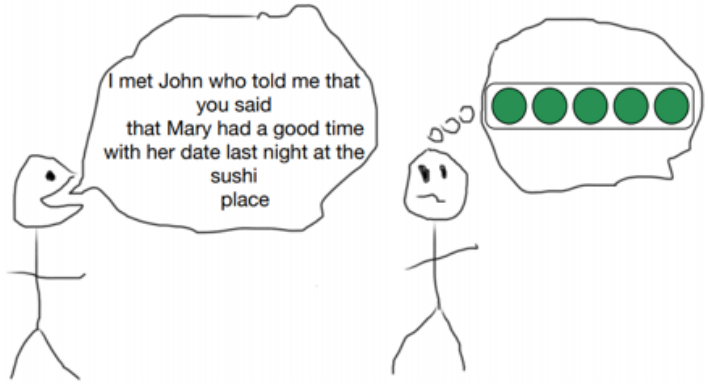
What you can cram into vectors?

NLP's quest for learning language representations

"You cannot cram the meaning of a whole sentence into a single $&!#* vector"



Prof. Ray Mooney

Cao Ying

Take-aways

1. Small breakthroughs gradually made towards BERT.
2. What are fundamental contributions between each milestone model?
    - *Fundamental learnings towards NLU which is the ultimate goal of NLP research from these works*
3. Future work after BERT's pre-training breakthroughs for NLU.

References

1. ***ULM-FiT (fast.ai 2018): Universal Language Model Fine-tuning for Text Classification***
2. ***ELMo (AllenNLP, 2018, NAACL 2018 the best paper award): Deep contextualized word representations***
    - Official blog
3. ***GPT (OpenAI, 2018): Improving Language Understanding by Generative Pre Training***
    - Official blogL Improving Language Understanding with Unsupervised Learning
4. GPT-2 (OpenAI,): Language Models are Unsupervised Multitask Learners
    - Official blog: Better Language Models and Their Implications
    - github
5. ***BERT (Google AI Language, 2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding***
6. TransformerXL (Google Brain, 2019): Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context
7. XLNet: Generalized Autoregressive Pretraining for Language Understanding
8. MASS (MSRA 2019) MASS: Masked Sequence to Sequence Pre-training
   Unified Language Model Pre-training for Natural Language Understanding and Generation
9. RoBERTa: A Robustly Optimized BERT Pretraining Approach
10. Megatron-LM (Nvidia, 2019): Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism
11. (Google, ICLR 2019 **under double-blind review**) ALBERT: A Lite BERT for Self-supervised Learning of Language Representations
    - PR: Meet ALBERT: a new 'Lite BERT' from Google & Toyota with State of the Art NLP performance and 18x fewer parameters

Backgroud (I)

Transductive(semi-supervised), Inductive and Active transfer learning

Imagine you have a training data, but only a subset of it has labels.

For example, you are trying to classify whether an image has a flower in it or not:

- You have 100,000 images,
- 1,000 images that you know definitively contain a flower;
- and another 1,000 that you know don't contain a flower;
- The other 98,000 you have no idea about: maybe they have flowers, maybe they don't.

Backgroud (II)

1. ***Transductive Transfer Learning*** (转导推理，aks. semi-supervised learning)
    - the other 98,000 images don't have labels, but they tell me something about the problem space.
    - still use the unlabeled data in training to help improve accuracy.
2. ***Inductive Transfer Learning*** （归纳推理）
    - only looks at the 2,000 labeled examples, builds a classifier on this.
3. ***Active Learning*** (主动学习)
    - when looking through the 98,000 examples, select a subset and request labels from an oracle.
    - the algorithm might say "of those 98,000, can you label this set of 50 images for me? That will help me build a better classifier".

Warmup

The most general inductive transfer setting for NLP

Given a static **source task** $\mathcal{T}_S$, and any **target task** $\mathcal{T}_T$ with $\mathcal{T}_S \neq \mathcal{T}_T$, the inductive transfer learning would like to improve the performance on $\mathcal{T}_T$.

Side Note: Why LM a perfect source task for NLP

1. It captures many facets of language relevant for downstream tasks, such as ***long-term dependencies***, ***hierarchical relations***, and ***sentiment***.
2. It provides data in ***near-unlimited quantities*** for most domains and languages.
3. A pretrained LM can be easily adapted to the idiosyncrasies of a target.
4. Language modeling already is ***a key component of many existing tasks*** such as MT and dialogue modeling.

ULM-FiT

Universal Language Model Fine-tuning

Breakthroughs

1. First prove a **universal** language model with fine-tuning works for any NLP task.
2. Summarize of a bunch of practically effective fine-tuning tricks.

Goal

A Universal Language Model Fine-tuning enables robust inductive transfer learning for any NLP task.

Challenges

The idea of LM fine-tuning is not widely adopted in the NLP field.

1. LMs overfit to small datasets.
2. LMs suffer catastrophic forgetting when fine-tuned with a classifier.
3. compared to CV, ***NLP models are typically more shallow*** and thus require different fine-tuning methods.

Approach: the three-stage workflow of ULM-FiT

Stage 1: General-domain LM pre-training

Capture general features of the language in different layers.

- three stacked AWD-LSTM: Regularizing and Optimizing LSTM Language Models
- trained on a ***general-domain corpus***.
  - Wikitext-103: 28,595 preprocessed Wikipedia articles and 103 million words.

Stage 2: Target task LM fine-tuning

Learn task-specific features.

- discriminative fine-tuning (Discr)
  - Tune each layer with different learning rates:

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta)$$

  - The paper empirically found it to work well:
    - First only fine-tue the last layer: choose the learning rate η^L
    - Tune lower layers gradually using $\eta^{l-1} = \eta^{l/2.6}$ as the learning rate.
- slanted triangular learning rates(STLR)

$$cut = \lfloor T \cdot cut\_frac \rfloor$$

$$p = \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t-cut}{cut \cdot (1/cut\_frac - 1)}, & \text{otherwise} \end{cases}$$

$$\eta_t = \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio}$$

Stage 3: Target task classifier fine-tuning

Preserve low-level representations and adapt high-level ones.

1. Fine-tuing model: two additional linear blocks
    1. the first linear layer takes as the input the pooled last hidden layer states.
        - concatenate sequence last, max pooling, and mean pooling.
    2. each block uses batch normalization, and dropout.
    3. with ReLU activations for the intermediate layer.
    4. a softmax activation at the last layer.
2. Fine-tuning method
    1. ***gradual unfreezing***
    2. ***Discr***
    3. ***STLR***.

Please check more details of the Ablation Analysis section of this paper if interested.

ELMo

Embedding from Language ModeL

***Deep contextualized* word representations**

Breakthroughs

This work demonstrates that exposing the deep internals of the pre-trained network is crucial.

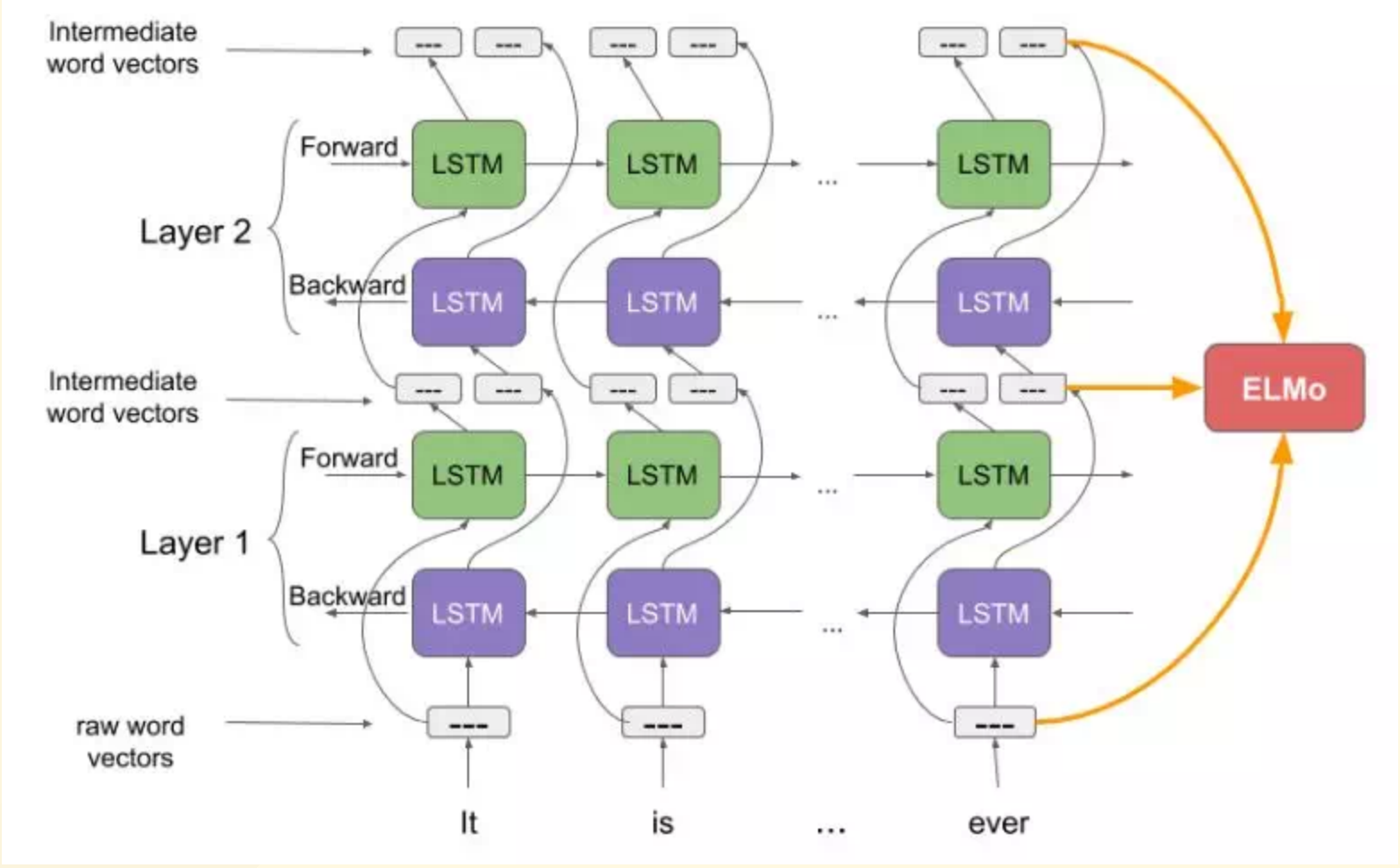
Goal

An ideal representation should model: syntax, semantics, and polysemy.

- An example of polysemy

  > “ I read the book yesterday.
  > Can you read the letter now? ”

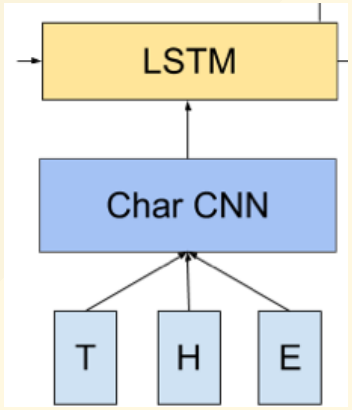
Approach: The overall ELMo architecture



Detail (I): biLM

LM used by ELMo is proposed from the paper: Exploring the Limits of Language Modeling, but

1. extends to bi-directional LSTM
2. some weights are shared between forward and backward directions.



Detail (I): biLM

Key learnings from the above paper used by ELMo, I think are twofold:

1. use big LSTM
2. CNN + LSTM

Quoted from Exploring the Limits of Language Modeling:

> most recent work on large scale LM has shown that ***RNNs are great in combination with N-grams***, as they may have different strengths that complement N-gram models, but ***worse when considered in isolation***.

Detail (II): ELMo vectors

1. for each token t_k, a L-layer biLM computes a set of $2L + 1$ representations

$$R_k = \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}_{k,j}^{LM}}, \overleftarrow{\mathbf{h}_{k,j}^{LM}} | j = 1, ..., L\}$$

2. compute a task-specific weighting of all biLM layers.

$$\text{ELMo}_k^{task} = E(R_k; \Theta) = \gamma^{task} \sum_{j=0}^{L} s_j^{stack} \mathbf{h}_{k,j}^{LM}$$

Detail (III): ELMo for downstream tasks

1. first freeze the weights of the biLM;
2. concatenate the ELMo vector ELMo_k^{task} with $\mathbf{x}_k$: $[\mathbf{x}_k; \mathrm{ELMo}_k^{task}]$;
3. further improvements are observed when including ELMo at the output layer of the target task;

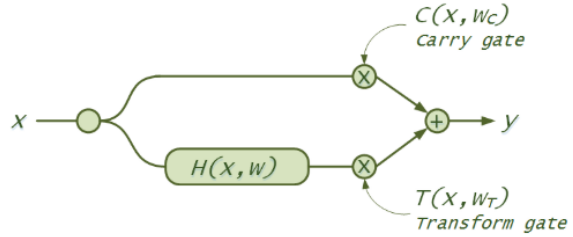
Model Capacity of ELMo

three layers exposed in ELMo:

1. input tokens flow into 2048 character-level convolution filters, followed by 2 highway layers.
2. a linear projection project CNN's output down to 512 dimensions.
3. $L = 2$, stacked 2 biLSTM layers.
    - 4096 hidden units, input/output dimension 512
    - with a residual connection between layers.

$$y = H(\boldsymbol{x}, W) \cdot T(\boldsymbol{x}, W_T) + x \cdot C(\boldsymbol{x}, W_C)$$

$T(\boldsymbol{x}, W_T)$ is the transform gate, and $C(\boldsymbol{x}, W_C)$ is the carry gate。



GPT

Generative Pre-Training

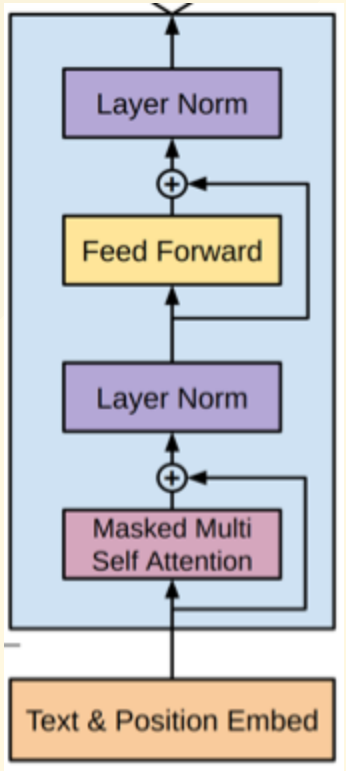
Breakthroughs

1. Use transformer decoder instead of LSTM to achieve better capture long-term linguistic structure.
2. Re-highlights the effectiveness of auxilliary tranining objectives.
3. Domonstrate that unsupervised pre-training (*generative pre-training*) + supervised fine-tuning framework works.

Approach (I)

Stage 1: Unsupervised pre-training

1. standard language modeling objective (MLE), denoted as L_1.
2. transformer decoder as the LM.



Approach (II)

Stage 2: Supervised fine-tuning

1. an extra linear output layer is added after the final transformer block's activation.
    - the learnable parameter for this layer and embeddings for **delimiter token** are the only extra parameters in fine-tuning.
2. loss function of the supervised task:

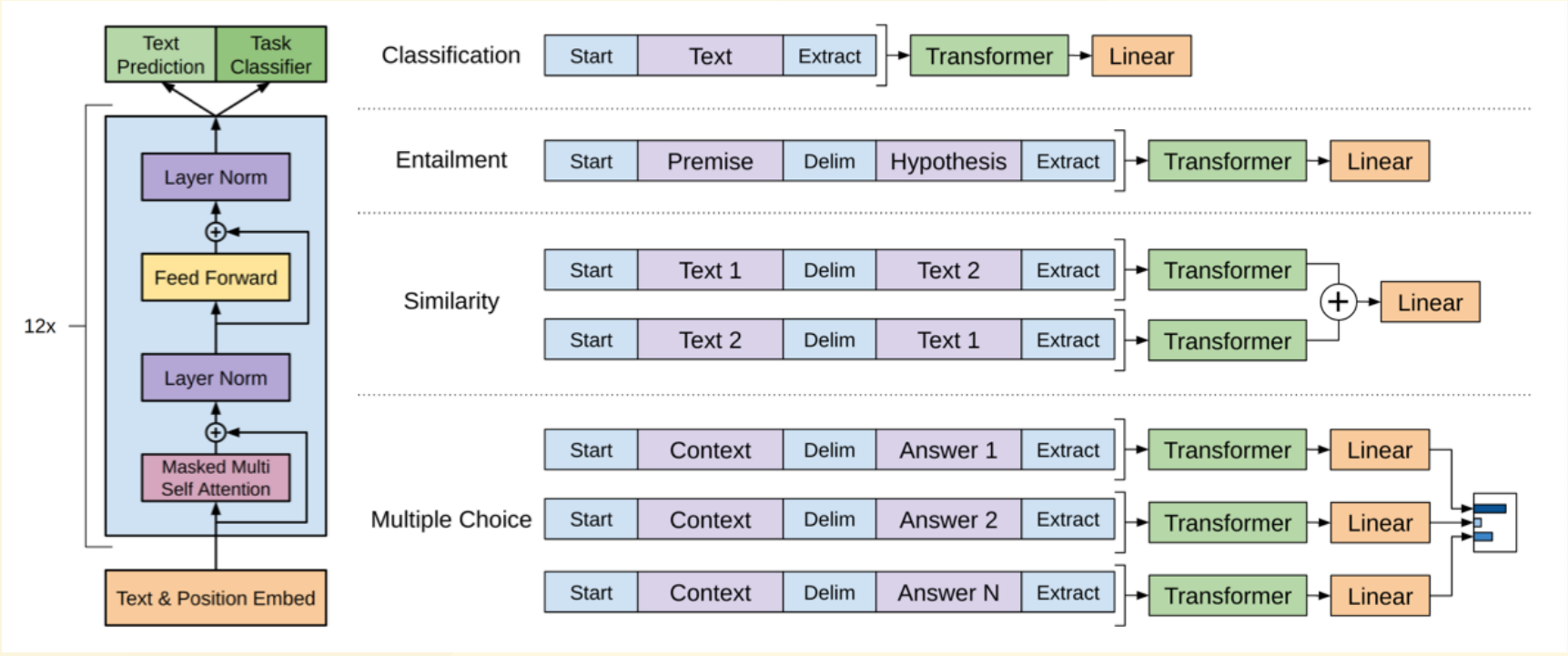
$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m)$$

3. add an auxiliary training objective with weight λ.

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

Approach (III)

task-specific input transformations



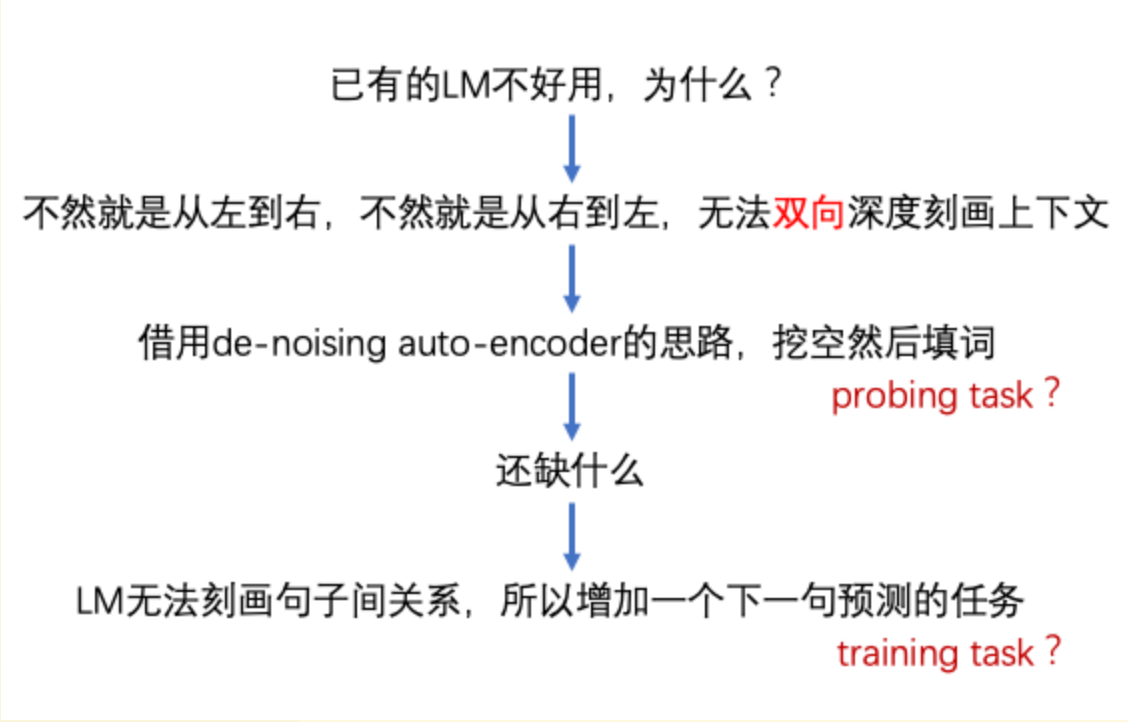
Model Capacity

1. $L = 12$, 12-layer transformer decoder (with masked self-attention) = BERT_{base}
   - hidden size $H = 768$, heads $A = 12$
   - FF/filter size = $4H = 3072$
2. train for 100 epoches on minibatches of 64 randomly sampled, contiguous sequences of 512 tokens.

BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

How BERT is designed?



[Reddit: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding]

> But for us the really amazing and unexpected result is that when we go **from a big model** (12 Transformer blocks, 768-hidden, 110M parameters) to a **really big model** (24 Transformer blocks, 1024-hidden, 340M parameters), we get huge improvements even on very small datasets (small == less than 5,000 labeled examples).

Motivations / Problems

1. A more powerful way to model bidirectional information.
2. Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two text sentences, which is not directly captured by language modeling.

	ID	sentence	label
Premise		A dog jumping for a Frisbee in the snow.	
Hypothesis	Example 1	An animal is outside in the cold weather, playing with a plastic toy.	*entailment*
	Example 2	A cat washed his face and whiskers with his front paw.	*contradiction*
	Example 3	A pet is enjoying a game of fetch with his owner.	*neutral*

Fig. NLI (蕴含/冲突/中立)

Breakthroughs

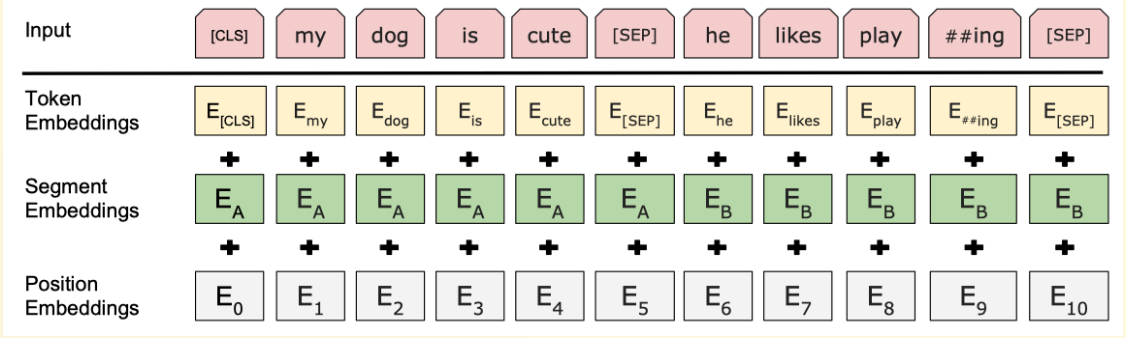
1. prove again that pre-trained representatioins eliminate the needs of many heavily-engineered task-specific architectures for sentence-level **and** token-level tasks.
2. advances the state-of-the-art for 11 **NLU tasks**.

Approach: Model Topology and Capacity

1. BERT_{base}
   - $L = 12$, 12-layer transformer **encoder** (with masked self-attention) = BERT_{base}
   - hidden size $H = 768$, heads $A = 12$
   - FF/filter size = $4H = 3072$
2. BERT_{large}
   - $L = 24$, 24-layer transformer **encoder** (with masked self-attention) = BERT_{base}
   - hidden size $H = 768$, heads $A = 12$
   - FF/filter size = $4H = 3072$

Approach I: Input Representation

BERT's input representation is able to ***unambiguously*** represent both a single text sentence or a pair of text sentences **in one token sequence**.



- We use WordPiece embeddings (Wu et al., 2016) with a 30,000 token vocabulary. We denote split word pieces with ##.
- We use learned positional embeddings with supported sequence lengths up to 512 tokens.

Approach II: Pre-training tasks

3.3.1 Masked LM

1. mask some percentage of the input tokens at random.
2. feed final hidden vectors corresponding to the mask tokens into an output softmax over the vocabulary as in a standard LM to predict **only** those masked tokens.
    - In contrast to denosing auto-encoders, masked LM only predicts the masked words rather than reconstructing the entire input.

Highlight some caveats from BERT's design choices

Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and right-to-left model. Unfortunately, standard conditional language models can only be trained left-to-right *or* right-to-left, since bidirectional conditioning would allow each word to indirectly "see itself" in a multi-layered context.

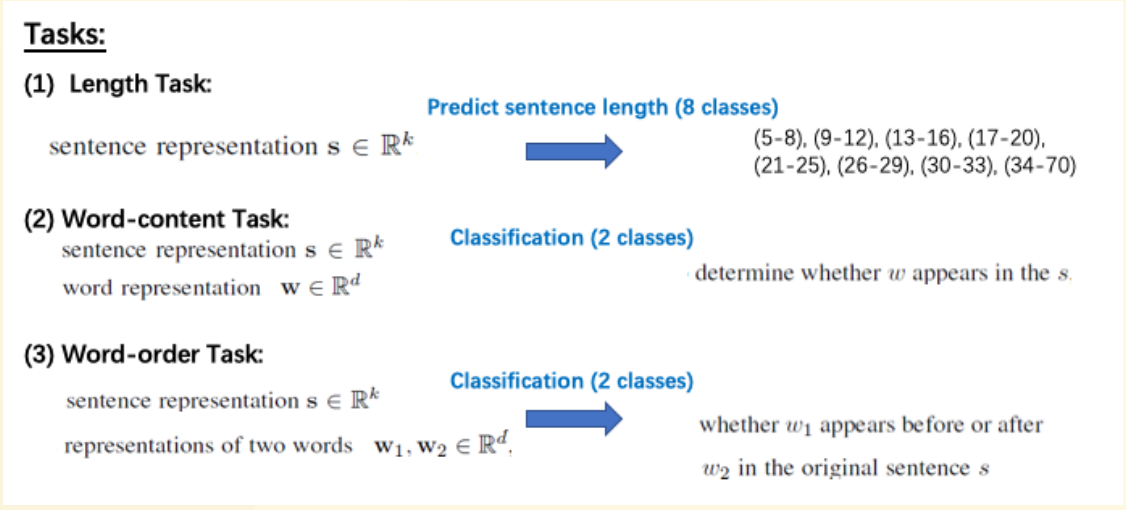
In order to train a deep bidirectional representation, we take a straightforward approach of masking some percentage of the input tokens at random, and then predicting only those masked tokens. We

The second downside of using an MLM is that only 15% of tokens are predicted in each batch, which suggests that more pre-training steps may be required for the model to converge. In Sec-

Approach II: Pre-training tasks

3.3.2 Next Sentence Prediction (the auxiliary prediction task)

Some examples of popular auxiliary prediction tasks to achieve/validate NLU



Approach III: Pre-training Procedure

- largely follows the existing literature on language model pre-training
- batch size of 256 sequences (256 sequences * 512 tokens = 128,000 tokens/batch)
- 1,000,000 steps.
    - approximately 40 epochs over the 3.3 billion word corpus.

Approach IV: Fine-tuning

BERT's fine-tuning is straight-forward.

1. for sentence-level task, obtain a fixed-dimensional pooled representation of the input sequence,
    - the final hidden state for the first token in the input.
    - by construction this corresponds to the the special `[CLS]` word embedding.
2. The only new parameters added during fine-tuning are for a classification layer W.

Some future work after BERT

1. Light weighted pre-training models.
2. Light weighted pre-training models for inference.
3. Pre-training dose not work well for NLG.
    - MASS (MSRA 2019) MASS: Masked Sequence to Sequence Pre-training Unified Language Model Pre-training for Natural Language Understanding and Generation