Word Representation / Word Embeddings

Context Word Embeddings

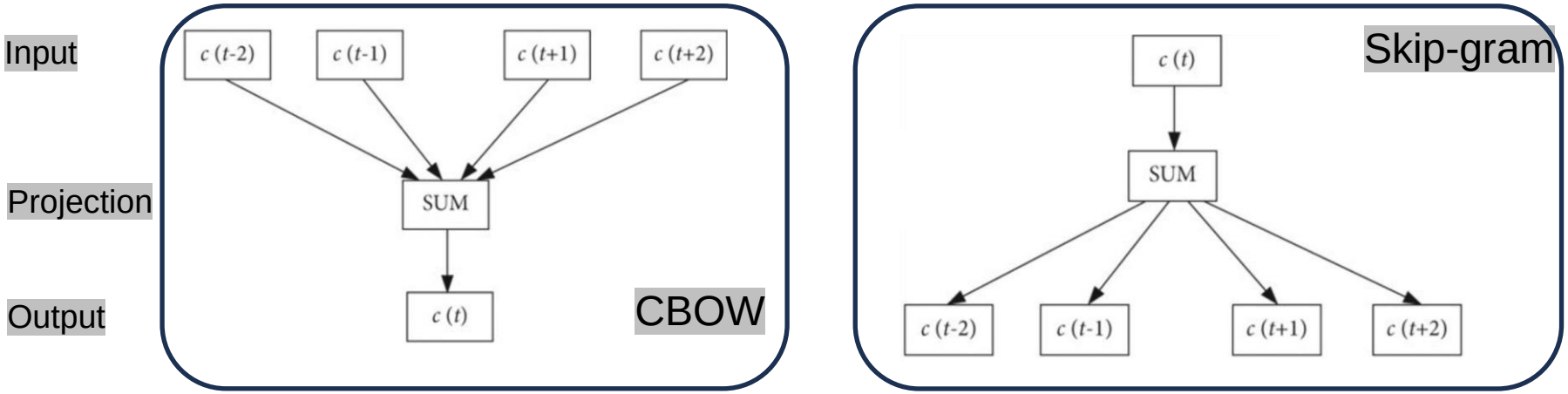
Recap

We learned about context word embeddings

We will discuss on context-dependent embeddings in next session

Text/Word Representation Recap

Word Embedding – Word2Vec



Word Embedding – Glove

Input

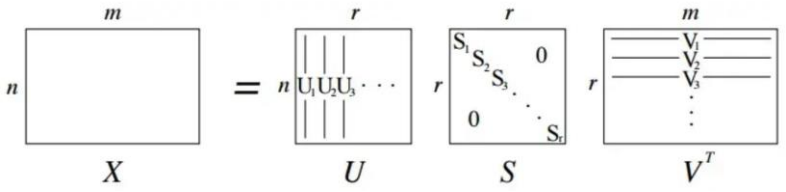
The fast cat wears no hat

The cat in the hat ran fast

Co-occurrence matrix

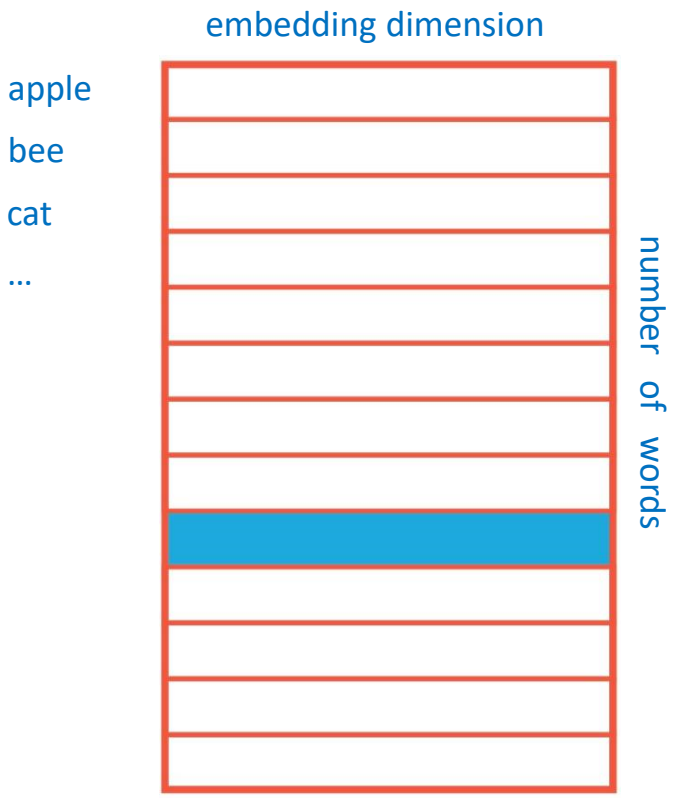
	cat	fast	hat	in	no	ran	the	wears
cat	0							
fast	2	0						
hat	2	2	0					
in	1	1	1	0				
no	1	1	1	0	0			
ran	1	1	1	1	0	0		
the	3	3	3	2	1	2	1	
wears	1	1	1	1	1	0	1	0

Output: embedding



Text/Word Representation Recap

Usage-like a lookup table



Problem-Polysemy

I saw a beautiful **python** at the zoo.
Can you help me debug this **Python** script?

I use the pandas library for data manipulation in Python.
The pandas in the zoo are a favorite attraction among children.

Solution-contextualized word embedding

Input	Output
"<bos>"	Predict: "The"
"<bos> The"	Predict: "cat"
"<bos> The cat"	Predict: "sat"
"<bos> The cat sat"	Predict: "on"
"<bos> The cat sat on"	Predict: "the"

Language Modeling : Contextualized Word Embedding

A good language model should produce good general-purpose and transferable representations from text

NLP Experts

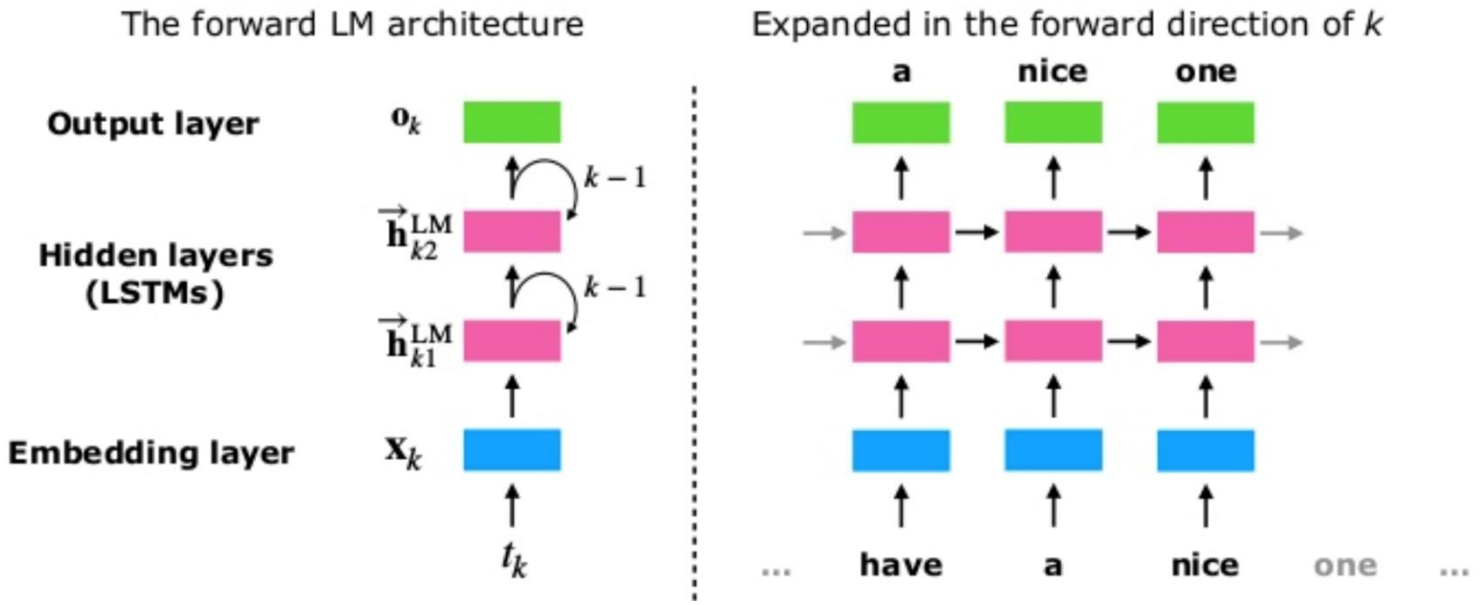
Models:
- ELMo
- BERT
- ALBERT
- RoBERTa
- ELECTRA
- ERNIE
- UniLM

Linguistic knowledge:
- The bicycles, even though old, were in good shape because ____ …
- The bicycle, even though old, was in good shape because ____ …

World knowledge:
- The University of Waterloo was founded in _____
- Ontario had a huge population boom as a launching point for expeditions to _____

Language Modeling : Contextualized Word Embedding

- Contextualized words embeddings aim at capturing word semantics in different contexts to address the issue of polysemous and the context-dependent nature of words.

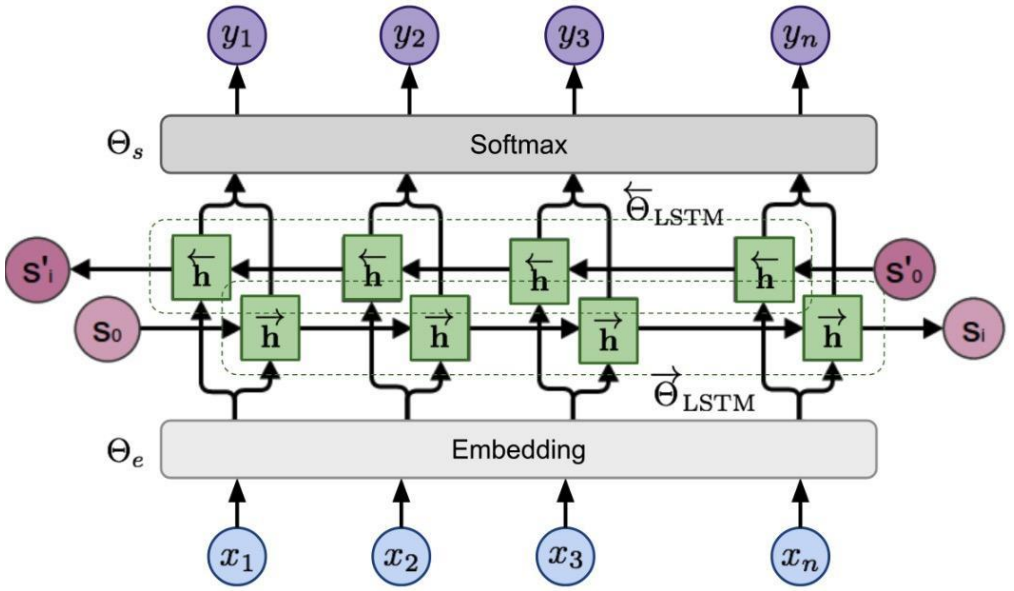


Training LM : The previous word along with the internal state of the LSTM are used to predict the next possible word (*supervised labels not needed*).

Task-specific word representation
Using the hidden states generated by the LSTM for each token to compute a vector representation of each word.

RNN Based: ELMo Architecture

Architecture



semantic features. — Top layer

syntactic features. — First layer

Θ_e embedding layer

Θ_s softmax layer

Input $(x_1, \ldots, x_n)$

Forward $p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_{i+1}, \ldots, x_n)$

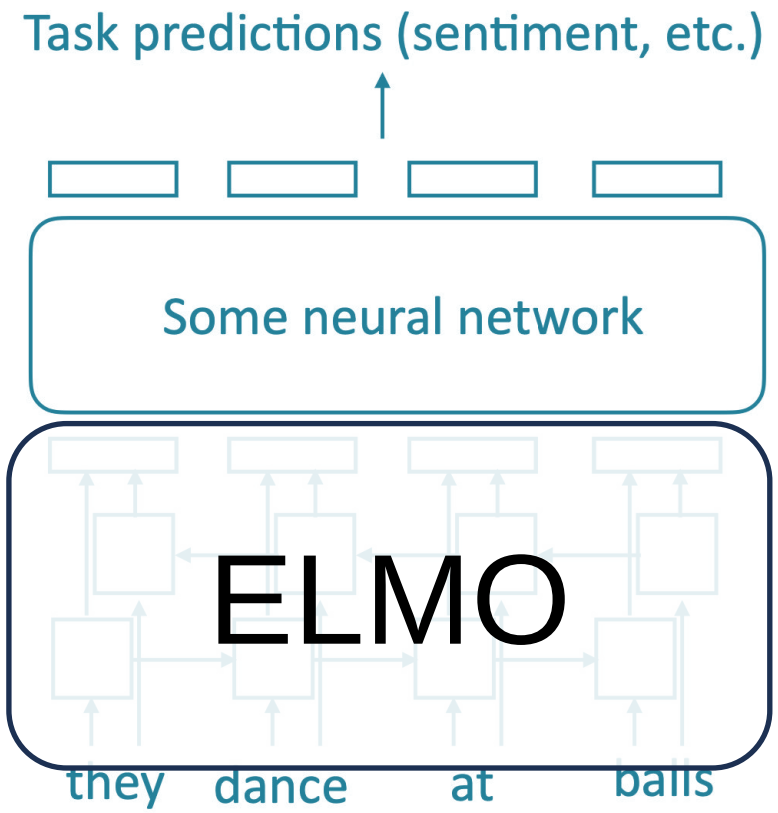
Backward $p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_{i+1}, \ldots, x_n)$

Loss

$$\mathcal{L} = -\sum_{i=1}^{n} \Big(\log p(x_i \mid x_1, \ldots, x_{i-1}; \Theta_e, \overrightarrow{\Theta}_{\text{LSTM}}, \Theta_s) + \log p(x_i \mid x_{i+1}, \ldots, x_n; \Theta_e, \overleftarrow{\Theta}_{\text{LSTM}}, \Theta_s) \Big)$$

RNN Based: ELMo Embeddings

How to Generate based on the Task

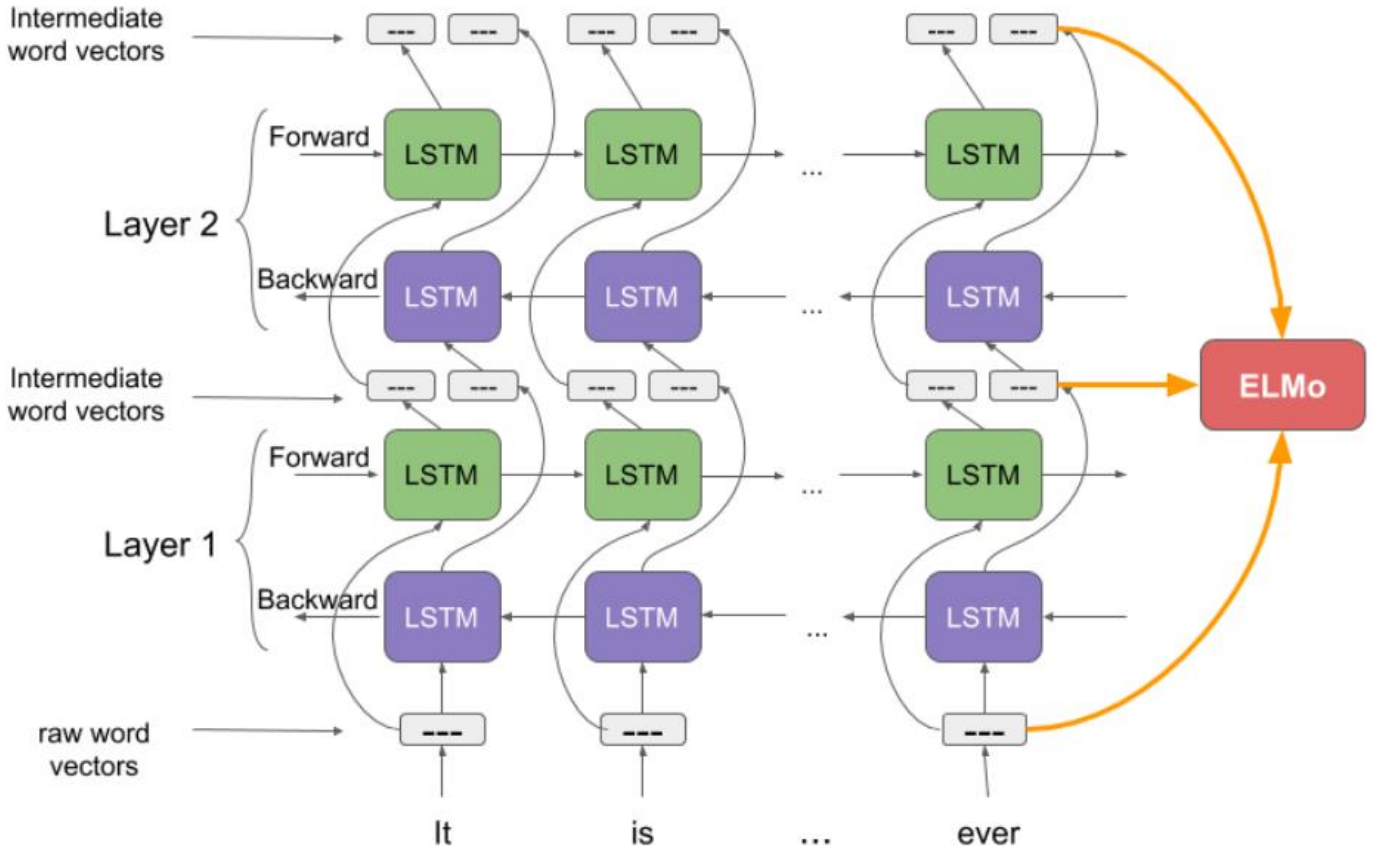


Take embeddings from ELMo

Frozen ELMo's weights, updates new neural network

Finetune both ELMo and new neural network

ELMo - Embeddings from Language Models(a biLM) in 2018



2 Layers Of Bi-directional LSTMs

Unlike traditional word embeddings such as word2vec and GLoVe, the ELMo vector assigned to a token or word is actually a function of the entire sentence containing that word. Therefore, the same word can have different word vectors under different contexts.

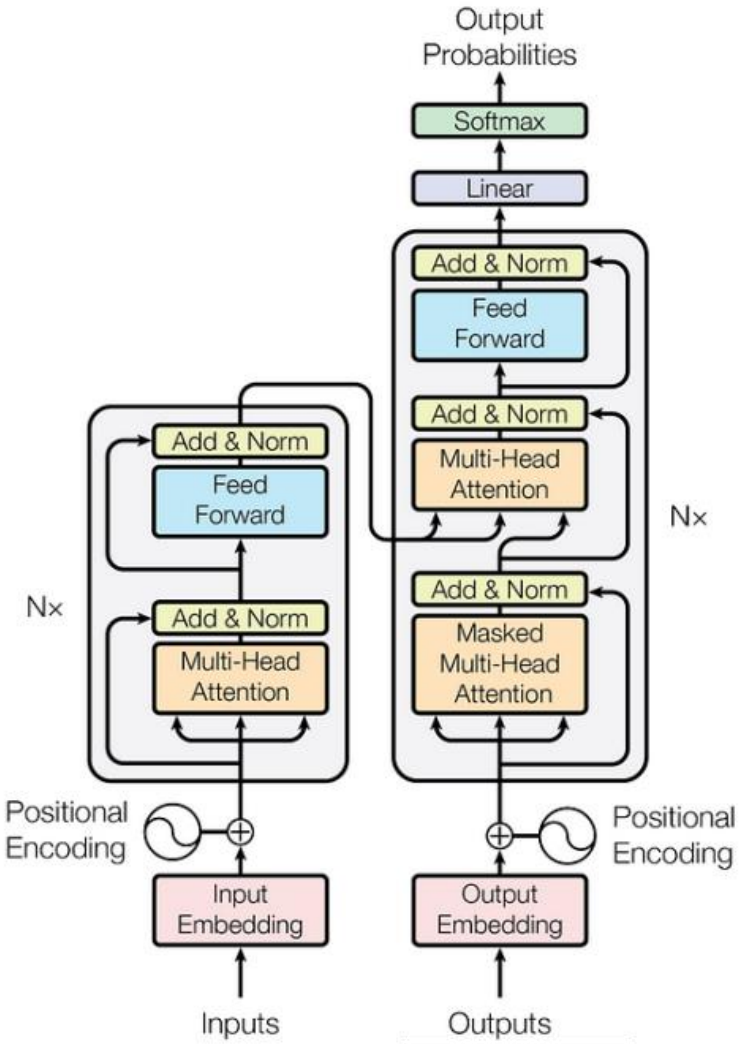
Drawbacks:

- ELMO utilizes LSTM instead of Transformer weaker feature extraction capability weaker integrated feature fusion than BERT
- Conclusion: ELMo model is far from reaching the level of human understanding of language tasks

Transformer Based

BERT

Encoder

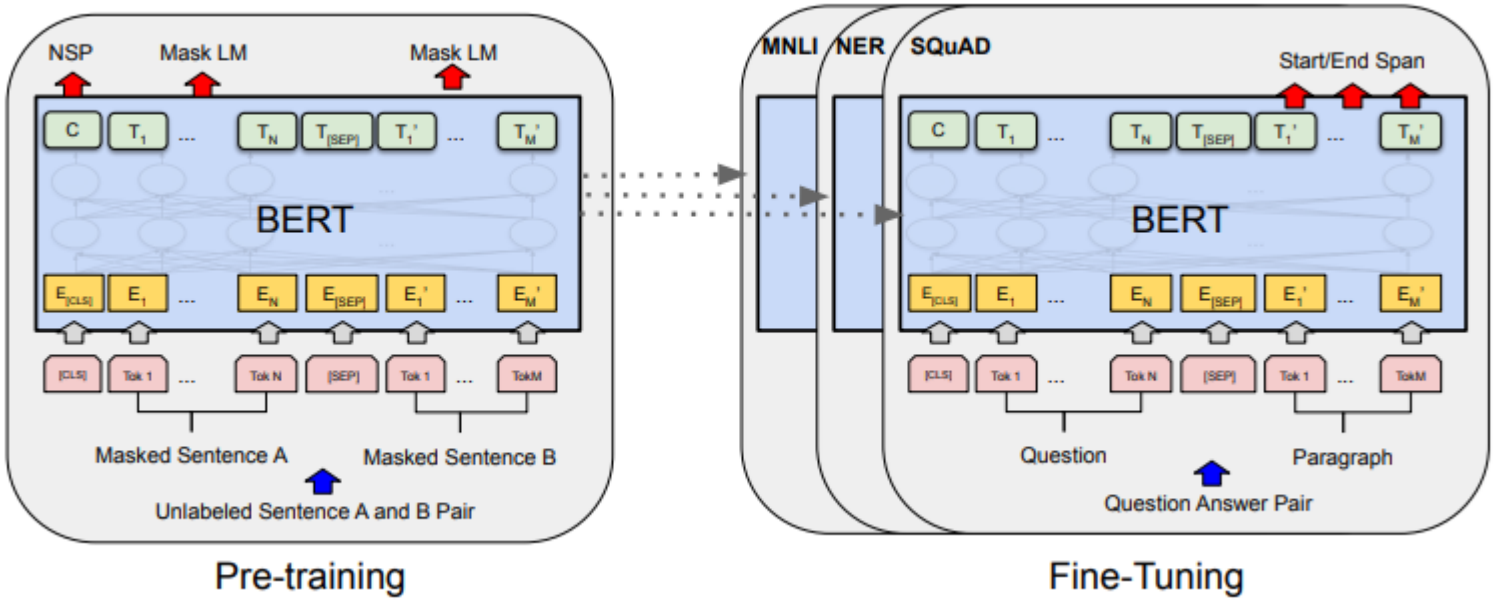


GPT

Decoder

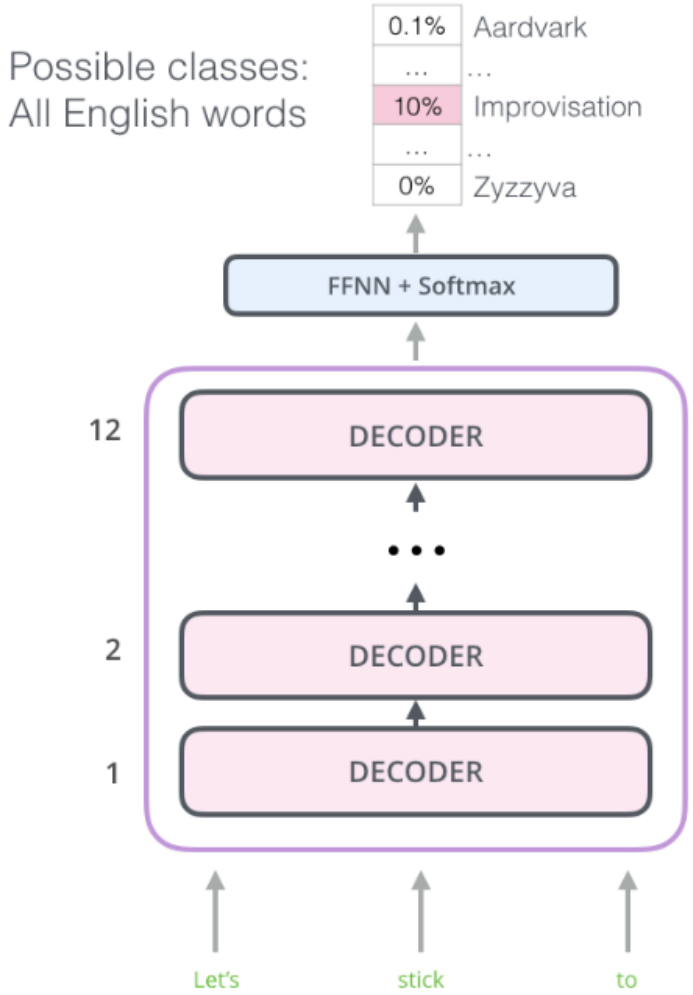
BERT

BERT, short for Bidirectional Encoder Representations from Transformers



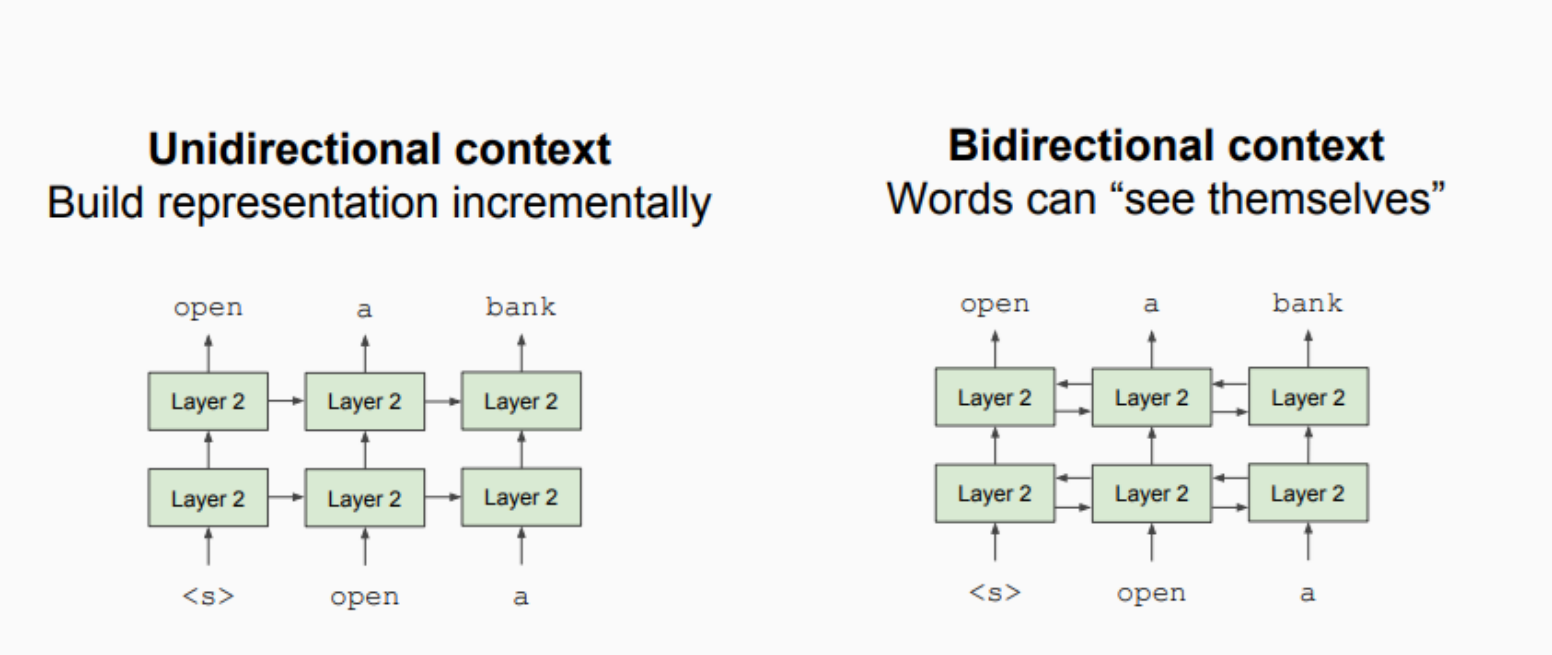
GPT

- OpenAI GPT, short for Generative Pre-training Transformer
  - Decoder-only architecture
    - Encoder part of the original Transformer is discarded
    - Each token is predicted conditioned on its previous tokens

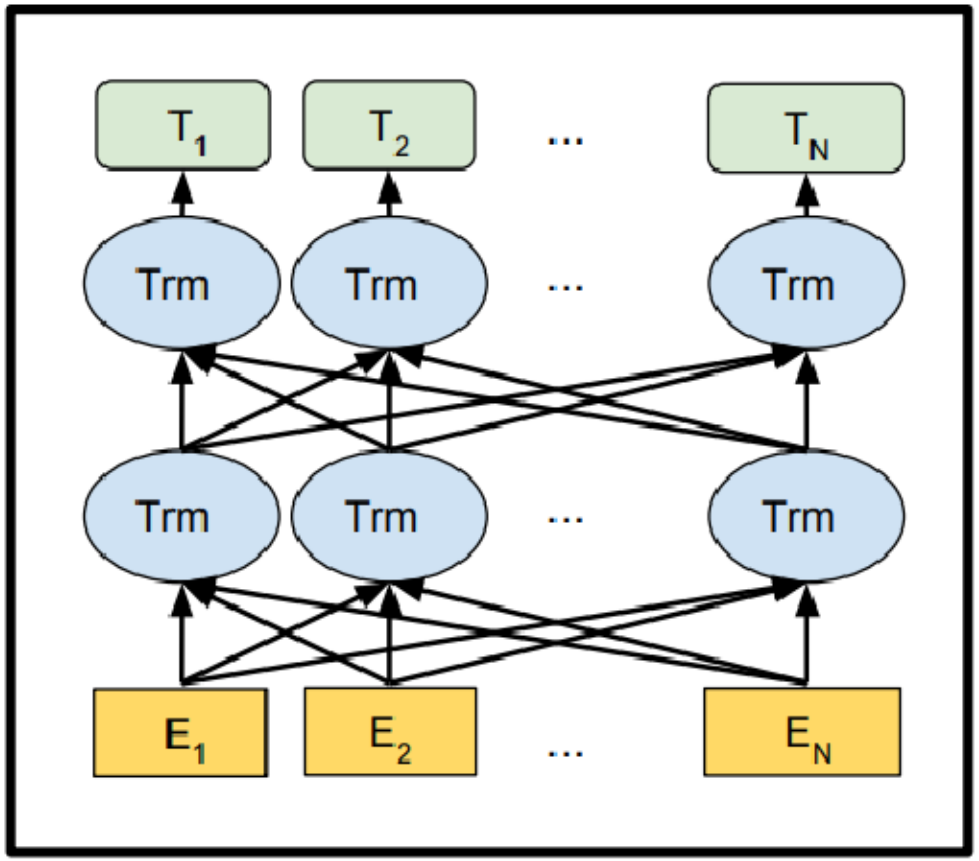


Learning Context?

- Problem: Language models are unidirectional, but language understanding is bidirectional.
  - “I went to the bank to deposit money.”
  - The meaning of “bank” depends on “to deposit money”



BERT - Architecture

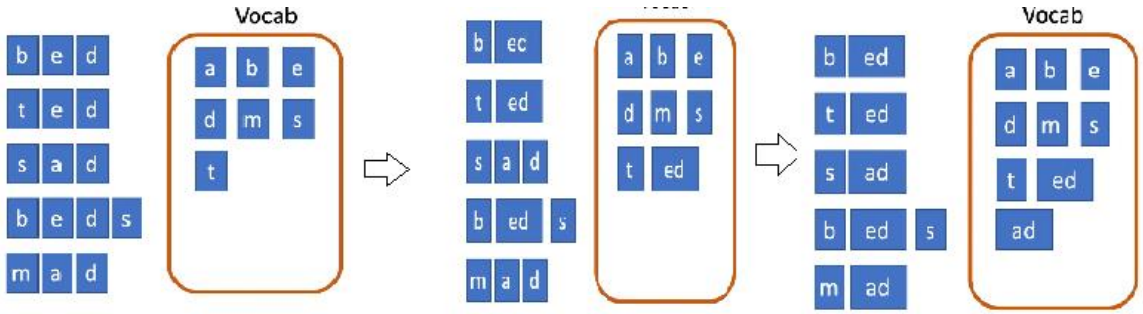


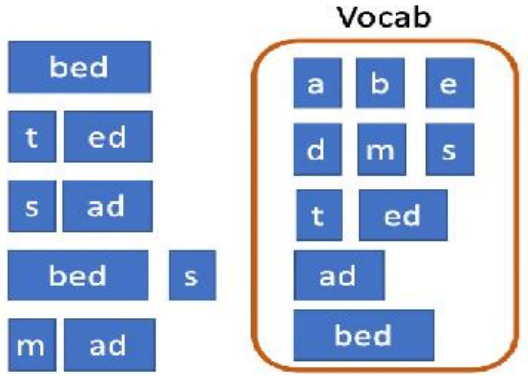
1. A word starts with its embedding representation from the embedding layer.
2. Every layer does some multi-headed attention computation on the word representation of the previous layer to create a new intermediate representation.
3. All these intermediate representations are of the same size. In the figure, **E1** is the embedding representation, **T1** is the final output and **Trm** are the intermediate representations of the same token.

BERT - Sentence Tokenization (WordPiece)

- Sub-word Tokenization Algorithms – Good OOV representation.
- more frequent words should be given unique ids,
- less frequent words should be decomposed into subword units that best retain their meaning.
- Example:
- “**wonderfully**” as a single word since it appears often in our dataset . Split “**structurally**” into “**structural**” and “**ly**” since “structually” is uncommon and we want to assist the model by giving it information on how it is composed.

- WordPiece Tokenization:
- Example Words: "bed", "ted", "sad", "beds", "mad"





BERT - Example: WordPiece Tokenization

"Jet makers feud over seat width with big orders at stake"

Word Pieces:

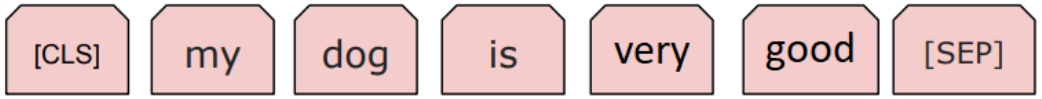
_J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

In the above example,

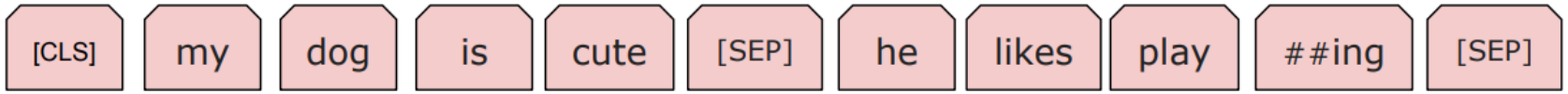
- the word "Jet" is broken into two wordpieces "_J" and "et",
- word "feud" is broken into two wordpieces "_fe" and "ud".
- The other words remain as single wordpieces.
- "_" is a special character added to mark the beginning of a word.

BERT - Input Representation For Pretraining Tasks

1. The first token of every input sequence is the special classification token – **[CLS]**. This token is used in classification tasks as an aggregate of the entire sequence representation. It is ignored in non-classification tasks.
2. For single text sentence tasks, this **[CLS]** token is followed by the WordPiece tokens and the separator token – **[SEP].**

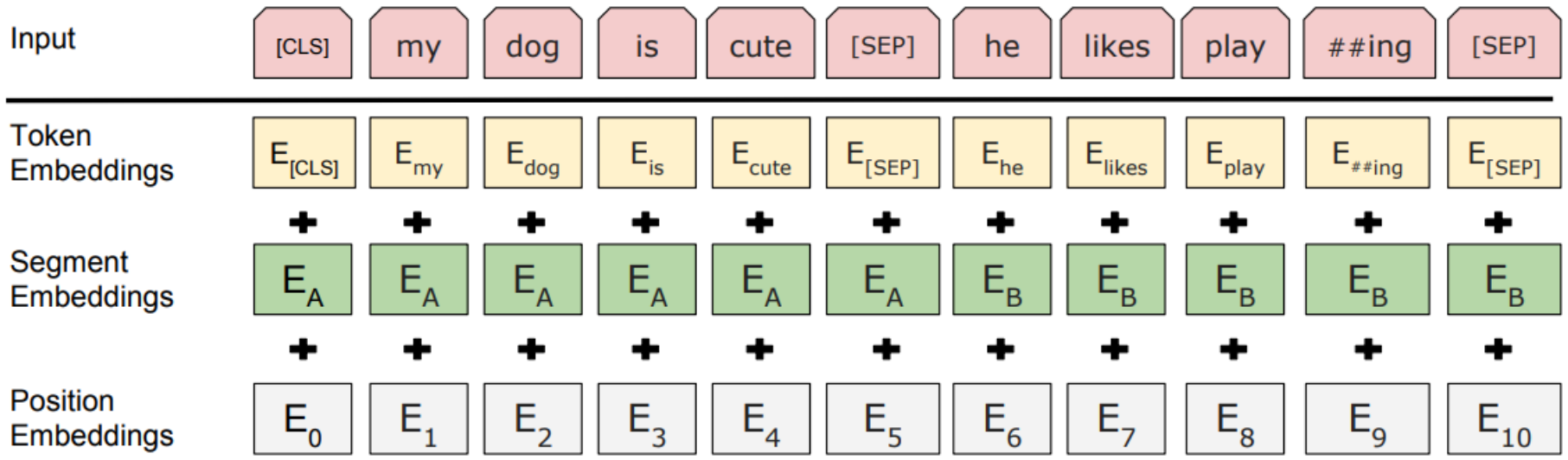


3. For sentence pair tasks, the WordPiece tokens of the two sentences are separated by another **[SEP]** token. This input sequence also ends with the **[SEP]** token.



BERT - Input Representation

Bert differentiates the sentences in two ways. First, we separate them with a special token ([SEP]). Second, a learned embedding to every token indicating whether it belongs to sentence A or sentence B is added.



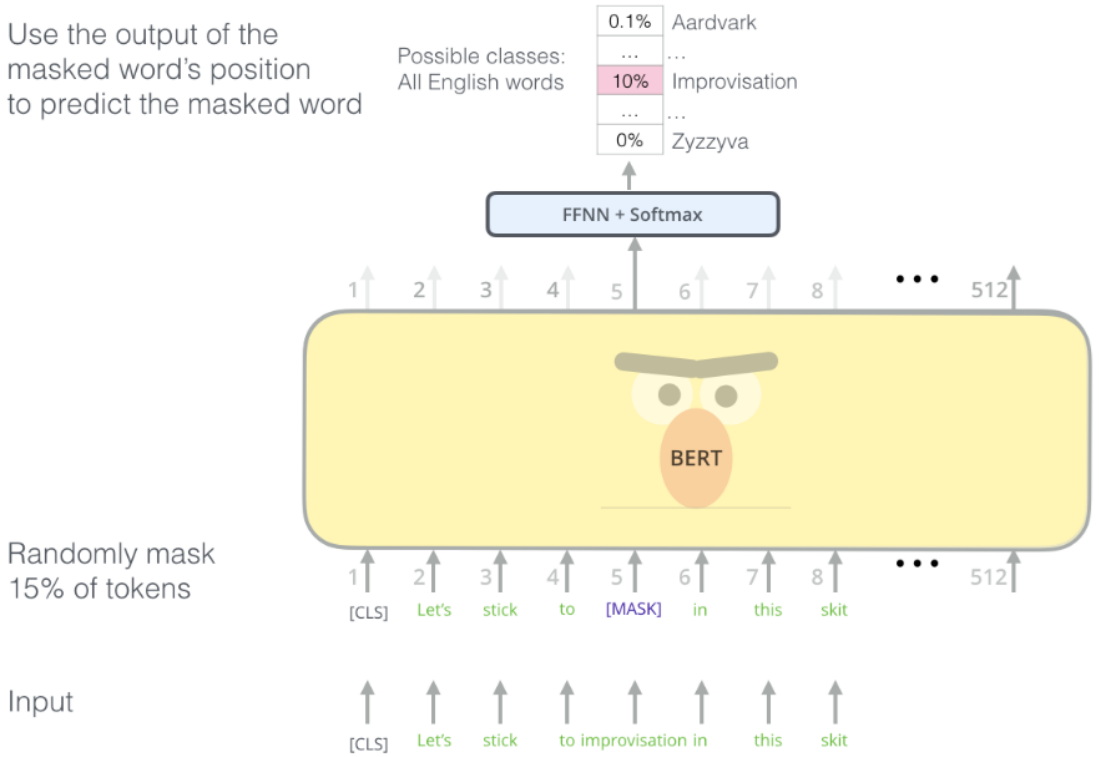
BERT - How does BERT handle OOV words?

- Vocab Size : 30,000 token vocabulary
- Max Seq Length: 512 tokens.
- OOV: Any word that does not occur in the vocabulary is broken down into sub-words greedily.
  - For example, if **play** is in vocabulary but **playing and played are OOV** words then they will be broken down into
  - Playing = play + ##ing
  - Played = play + ##ed respectively. (## is used to represent sub-words).

BERT - Pretraining Masked Language Modeling

Masked LM

- For each sentence15% of the tokens are chosen at random and –
- 80% of the time tokens are actually replaced with the token [MASK].
- 10% of the time tokens are replaced with a random token.
- 10% of the time tokens are left unchanged.
- The masked words are not always replaced with the masked token – **[MASK]** because then the masked tokens would never be seen before fine-tuning.

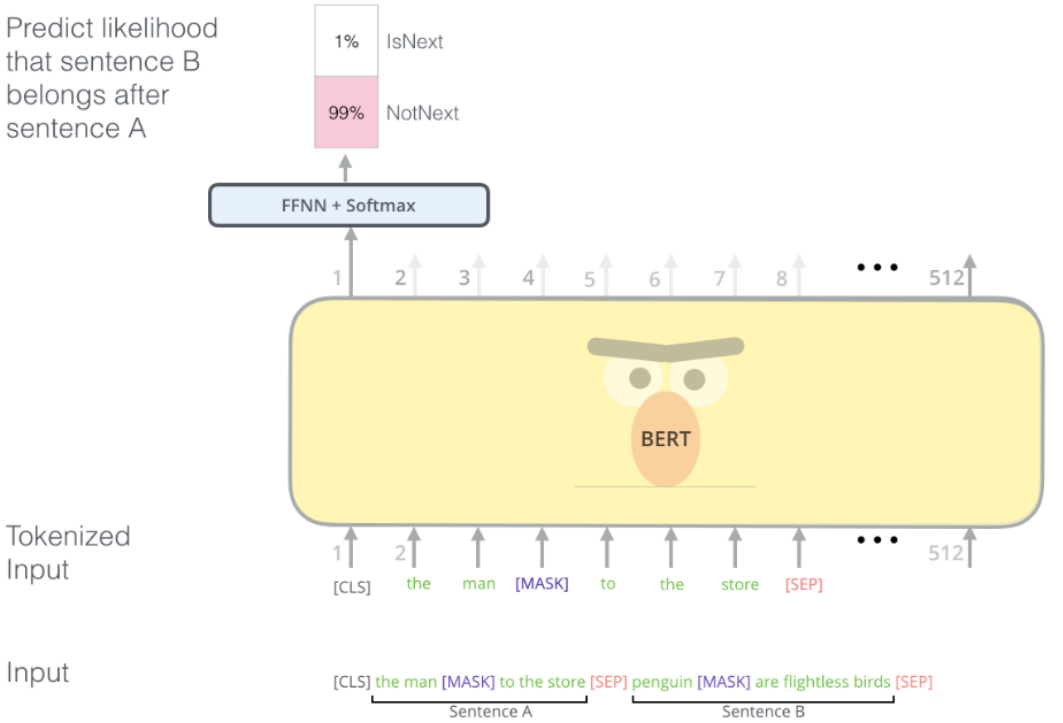


BERT's clever language modeling task masks 15% of words in the input and asks the model to predict the missing word.

BERT - Pretraining: Next Sentence Prediction

Next Sentence Prediction

- Next sentence prediction task is a binary classification task in which, given a pair of sentences, it is predicted if the second sentence is the actual next sentence of the first sentence.
- Given two sentences A and B, did B follow A in the training corpus, or not?
  - Help the model to learn relationships between sentences
  - Beneficial to downstream tasks including QA and NLI



The second task BERT is pre-trained on is a two-sentence classification task. The tokenization is oversimplified in this graphic as BERT actually uses WordPieces as tokens rather than words --- so some words are broken down into smaller chunks.

BERT - Pre-training of Deep Bidirectional Transformers for Language Understanding (2018)

- BERT is a stack of transformer layers – with multi headed self attentions

BERT-Model	Number of layers (i.e., Transformer blocks)	The hidden size as H	number of self-attention heads	Total Parameters
Base	12	768	12	110M
Large	24	1024	16	340M

- Pre-Training: Learn a good set of initial weights for a network in an unsupervised way, such that the weights can be fine-tuned for any particular NLP problem like classification, clustering etc.

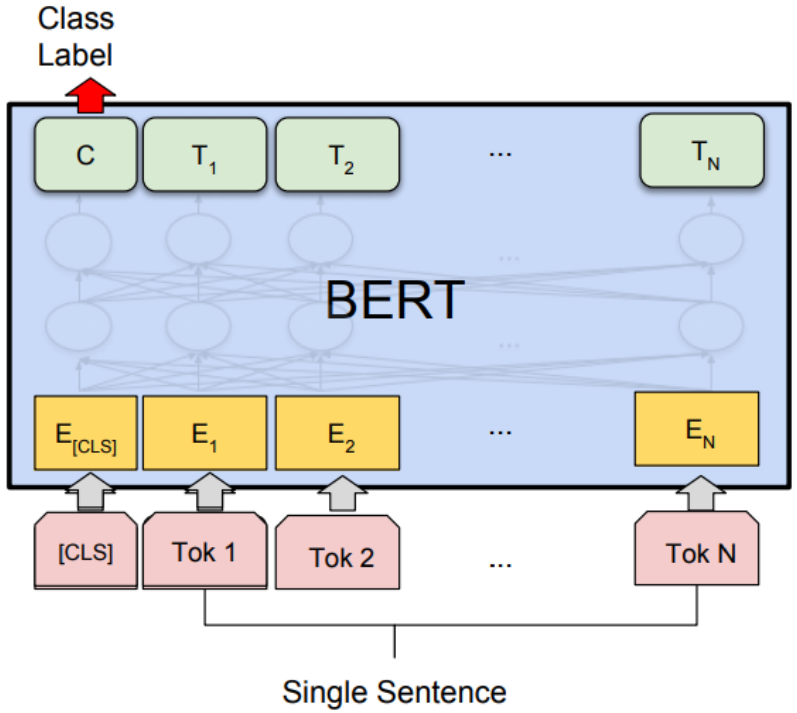
BERT - Finetuning: Specific NLP Tasks

For each task, simply plug in the task specific inputs and outputs into BERT and finetune all the parameters end-to-end.

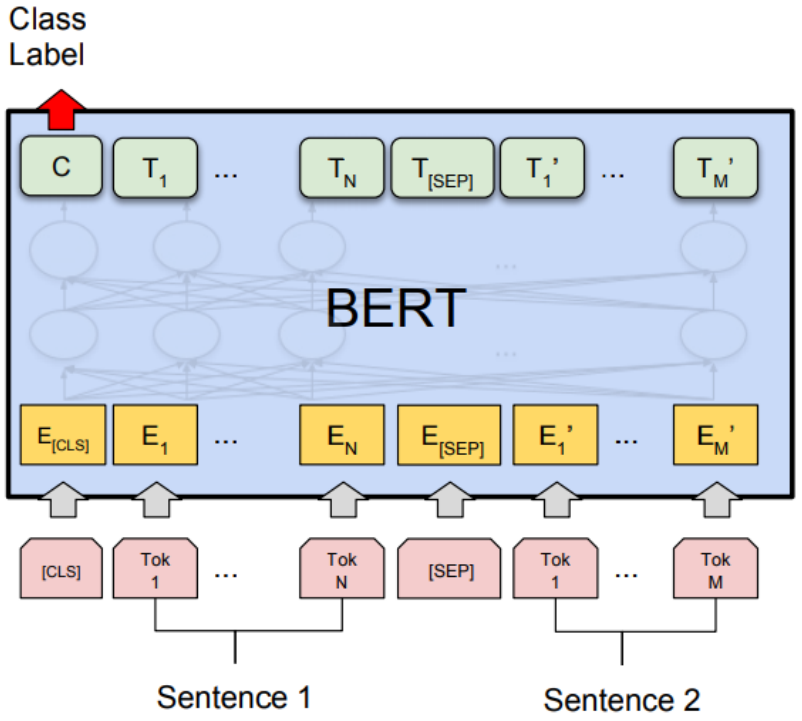
Fine Tuning Tasks
Question Answering(SQuAD: Stanford Question Answering Dataset): Question answering is a prediction task. Given a question and a context paragraph, the model predicts a start and an end token from the paragraph that most likely answers the question.
Named entity recognition, PoS tagging
Sentence Classification like sentiment
General Language Understanding Evaluation (GLUE) benchmark – paraphrase, coreference, sentence similarity.

BERT - Fine-Tuning: Add a Suitable Classifier layer

- Sentence Classification



Sentence Pair Classification

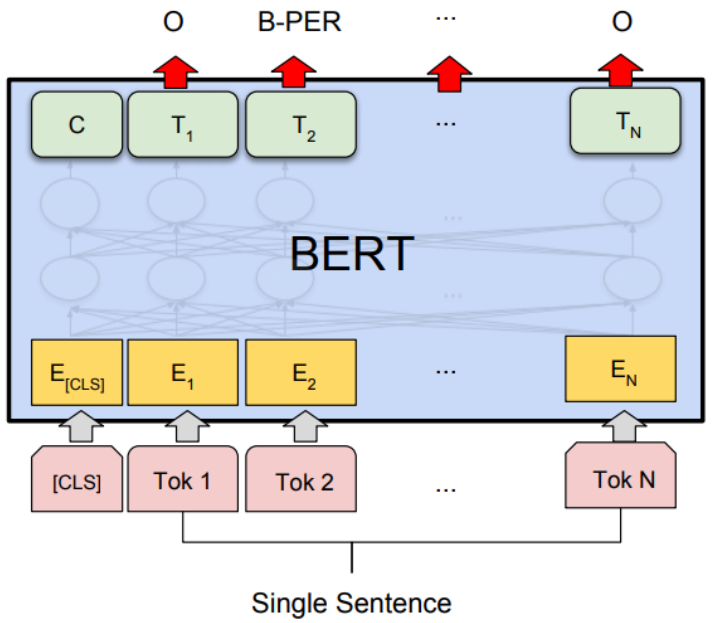


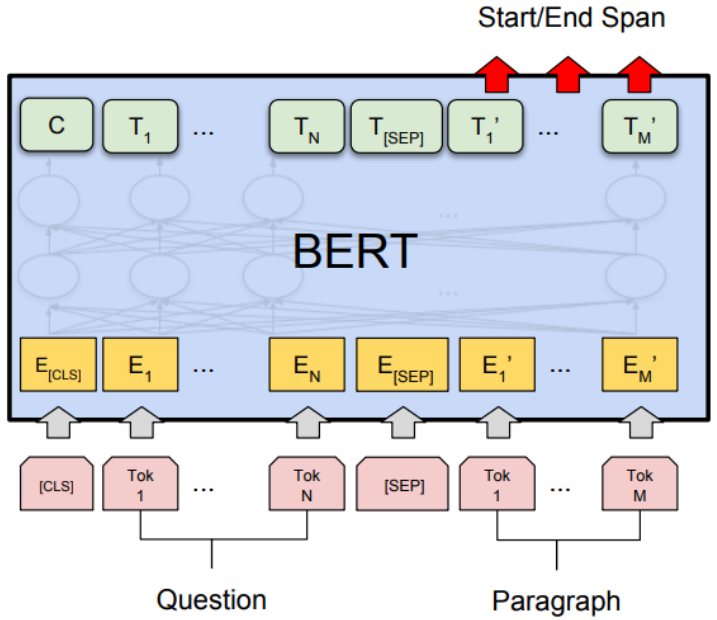
only difference is in the input representation where the two sentences

BERT - Fine-Tuning: Add a Suitable Classifier layer

- NER/PoS

  Question Answering





The final hidden states (the transformer output) of every input token is fed to the classification layer to get a prediction for every token.

2 new parameters learned during fine-tuning a **start vector** and an **end vector** with size equal to the hidden shape size.
softmax(S . K), where **S** is the start vector and **K** is the final transformer output of token **i. Similarly , softmax(E . K)**, where E is the end vector

BERT in Speech Recognition Task

- http://proceedings.mlr.press/v101/shin19a.html evaluated on LibriSpeech ASR task

- **A given sentence:**

  `move the vat over the hot fire`

- **A set of instances we create:**

  1. Input = `[MASK] the vat over the hot fire`
     Label = `move`
  2. Input = `move [MASK] vat over the hot fire`
     Label = `the`
     ...
  7. Input = `move the vat over the hot [MASK]`
     Label = `fire`

LM takes each instance and computes the likelihood of the original word in the masked position

. Finally, the score of the given sentence is obtained by summing all log-likelihoods of the masked words from each input instance.

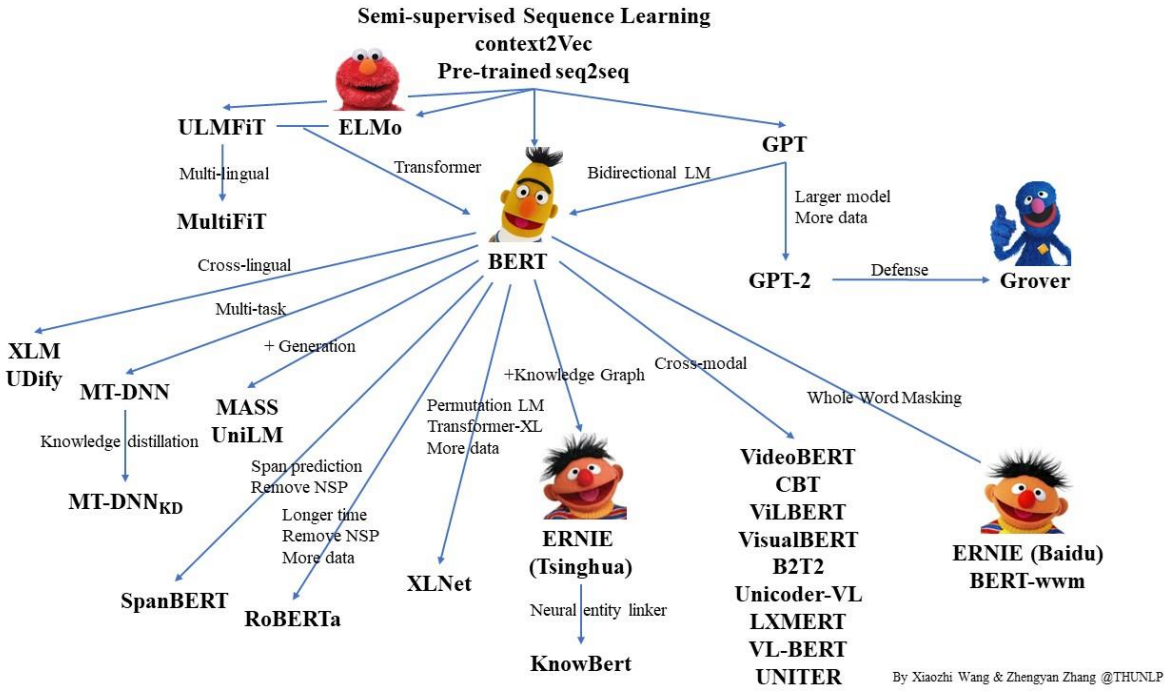
BERT Results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (`https://gluebenchmark.com/leaderboard`). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

BERT Variants

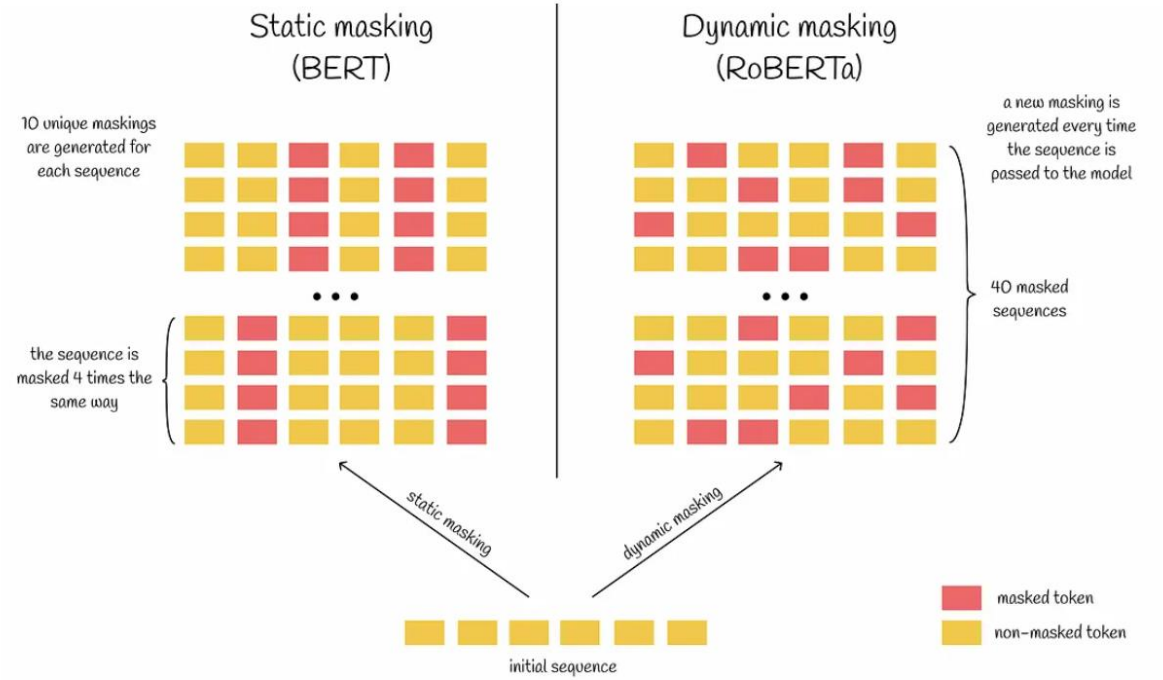
- The original BERT, with its size and data requirements, has certain limitations
- To address these limitations, several variants of BERT have been developed
  - RoBERTa
  - ALBERT
  - ERNIE
  - ELECTRA
  - ……



RoBERTa

- RoBERTA, short for Robustly optimized BERT approach
- One of the most popular variants of BERT
- Main contributions:
  - Introduce alternative design choices for BERT
  - Add a new pre-training dataset CommonCrawl News

BERT	RoBERTa
Static masking/substitution	Dynamic masking/substitution
Inputs are two concatenated document segments	Inputs are sentence sequences that may span document boundaries
Next Sentence Prediction (NSP)	No NSP
Training batches of 256 examples	Training batches of 2,000 examples
Word-piece tokenization	Character-level byte-pair encoding
Pretraining on BooksCorpus and English Wikipedia	Pretraining on BooksCorpus, CC-News, OpenWebText, and Stories
Train for 1M steps	Train for up to 500K steps
Train on short sequences first	Train only on full-length sequences



Static masking vs Dynamic masking

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

Table 1: Comparison between static and dynamic masking for $\text{BERT}_{\text{BASE}}$. We report F1 for SQuAD and accuracy for MNLI-m and SST-2. Reported results are medians over 5 random initializations (seeds). Reference results are from Yang et al. (2019).

RoBERTa Results

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

Table 5: Results on GLUE. All results are based on a 24-layer architecture. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively. RoBERTa results on the development set are a median over five runs. RoBERTa results on the test set are ensembles of *single-task* models. For RTE, STS and MRPC we finetune starting from the MNLI model instead of the baseline pretrained model. Averages are obtained from the GLUE leaderboard.

ALBERT: A Lite BERT

ALBERT, short for A Lite BERT, is a light-weighted version of BERT model

Motivation: Scaling up BERT results in better performance; however, there are several challenges:

- Memory limitation
- Longer training time

"An ALBERT configuration similar to BERT-large has 18x fewer parameters and can be trained about 1.7x faster"

The core architecture of ALBERT is BERT-like in that it uses a transformer encoder architecture, along with GELU activation. In the paper, they also use the identical vocabulary size of 30K as used in the original BERT. (V=30,000). However, ALBERT makes three substantial and important changes:

Uses 89% fewer parameters than the state-of-the-art BERT model with little loss of accuracy [BERT: 82.3% , ALBERT : 80.1 on several NLP benchmarks accuracy]

ALBERT uses two optimizations to reduce model size:

- Factorized Embedding Parameterization
- Cross Layer Parameter Sharing

ALBERT tuning is based on Sentence Order Prediction[SOP]

ALBERT : Factorize Embedding Parameterization

ALBERT authors point out that **WordPiece embeddings** are designed to learn **context *independent* representations**. The **hidden layer embeddings** are designed to learn **context *dependent* representations**.

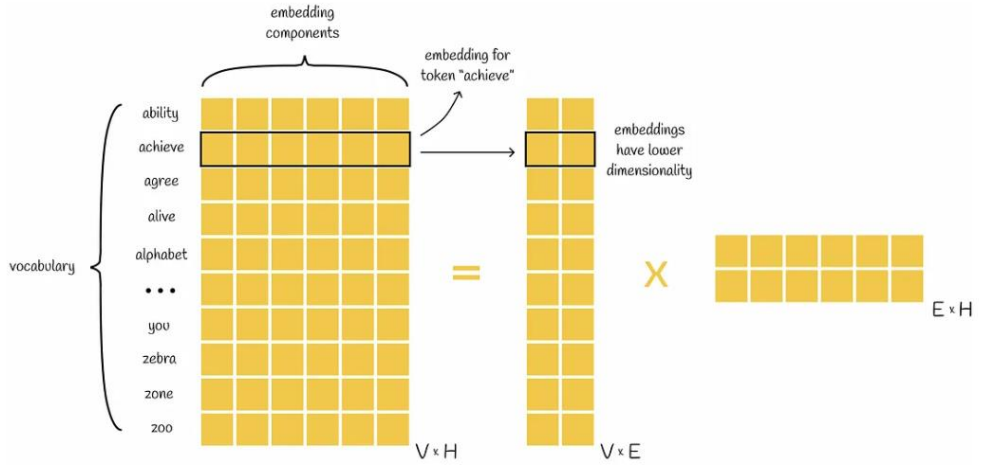
The **power of BERT** largely relies on **learning context dependent representations via the hidden layers**. If you **tie H and E**, and with NLP requiring **large V** (vocab), then your **embedding matrix E**, which is really **V*E**, must **scale** with **H** (hidden layers)…and thus you end up with models that can have **billions of parameters**, but most of which are **rarely updated** in **training**.

In BERT, the token embedding size E matches the hidden layer size H

A suboptimal choice

- The power of BERT comes from hidden-layer embeddings, so naturally H >> E
- Increase H would increase the size of the embedding matrix: V x H, which could be very large
  - V = 30k, H = 768 => 23M

- Decompose the embedding
  - O(V x E + E x H) vs. O(V x H)
  - By factorizing the embedding, the ALBERT team first projects the word vectors into a smaller-dimensional space: **128** vs BERT's **768**



Vocabulary matrix factorization

ALBERT: Parameter Sharing

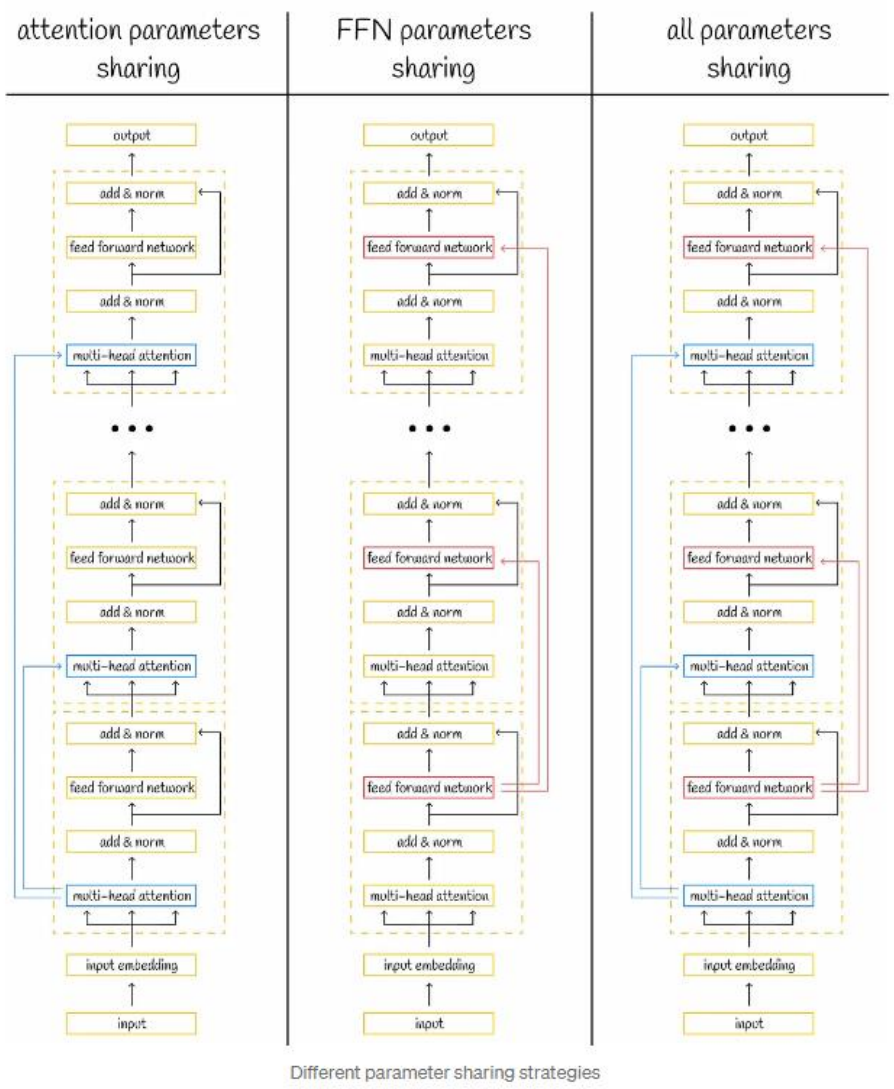
The second optimization is to share parameters across the network's layers

Transformer network layers contain both a **feed-forward component** and an **attention component**.

ALBERT's strategy is to share each component across all layers

This does result in a loss of accuracy of about 1.5 percentage points, but it does reduce the number of parameters needed from 89M to 12M.

- Parameter sharing can greatly reduce parameters without compromising the performance significantly
- Several ways of parameter sharing across layers
  - only share FFN parameters
  - only share attention parameters
  - share all the parameters (default)



Different parameter sharing strategies

ALBERT: Sentence Order Prediction

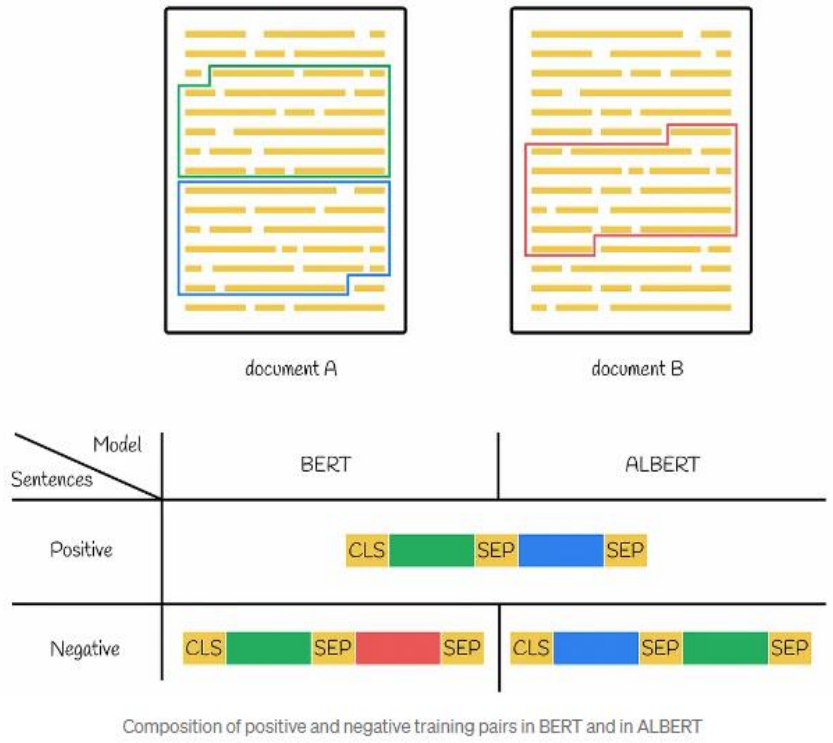
In the BERT paper, Google proposed a **next-sentence prediction(NSP)** technique to improve the model's performance in downstream tasks, but subsequent studies found this to be unreliable.

Researchers used a **sentence-order prediction (SOP)** loss to model inter-sentence coherence in ALBERT, which enabled the new model to perform more robustly in multi-sentence encoding tasks.

[S1] Give your heart to dogs. [S2]They will never break it. →1

[S1] They will never break it.[S2] Give your heart to dogs. →-1

- Multiple studies have shown NSP is ineffective and decided to remove it
- Replace NSP with Sentence Order prediction (SOP) problem
  - Predict whether two sentences are in the correct order or not
  - Model inter-sentence coherence



Composition of positive and negative training pairs in BERT and in ALBERT

ALBERT

Dataset: For pretraining baseline models, researchers used the BOOKCORPUS and English Wikipedia, which together contain around 16GB of uncompressed text.

Experiment results: The ALBERT model significantly outperformed BERT on the language benchmark tests SQuAD1.1, SQuAD2.0, MNLI SST-2, and RACE.

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.5/83.3	80.3/77.3	84.1	91.7	68.3	82.1	17.7x
	large	334M	92.4/85.8	83.9/80.8	85.8	92.2	73.8	85.1	3.8x
	xlarge	1270M	86.3/77.9	73.8/70.5	80.5	87.8	39.7	76.7	1.0
ALBERT	base	12M	89.3/82.1	79.1/76.1	81.9	89.4	63.5	80.1	21.1x
	large	18M	90.9/84.1	82.1/79.0	83.8	90.6	68.4	82.4	6.5x
	xlarge	59M	93.0/86.5	85.9/83.1	85.4	91.9	73.9	85.5	2.4x
	xxlarge	233M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	1.2x

ALBERT Results

dramatically reduce the model size :Consider the size comparison below — BERT x-large has 1.27 Billion parameters, vs ALBERT x-large with 59 Million parameters!

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
	xlarge	1270M	24	2048	2048	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	59M	24	2048	128	True
	xxlarge	233M	12	4096	128	True

1.27 Billion params in BERT vs 59M in ALBERT for the same size network (hidden/layers) ... ~21.5x smaller.

ERNIE

- **Baidu Inc**

1. https://arxiv.org/pdf/1904.09223.pdf ERNIE 1.0: Enhanced Representation through Knowledge Integration (Apr 2019)
2. https://arxiv.org/pdf/1907.12412.pdf ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding (Nov 2019)

- **Huawei Noah's Ark Lab**

1. https://arxiv.org/pdf/1905.07129.pdf ERNIE: Enhanced Language Representation with Informative Entities (Jun 2019)

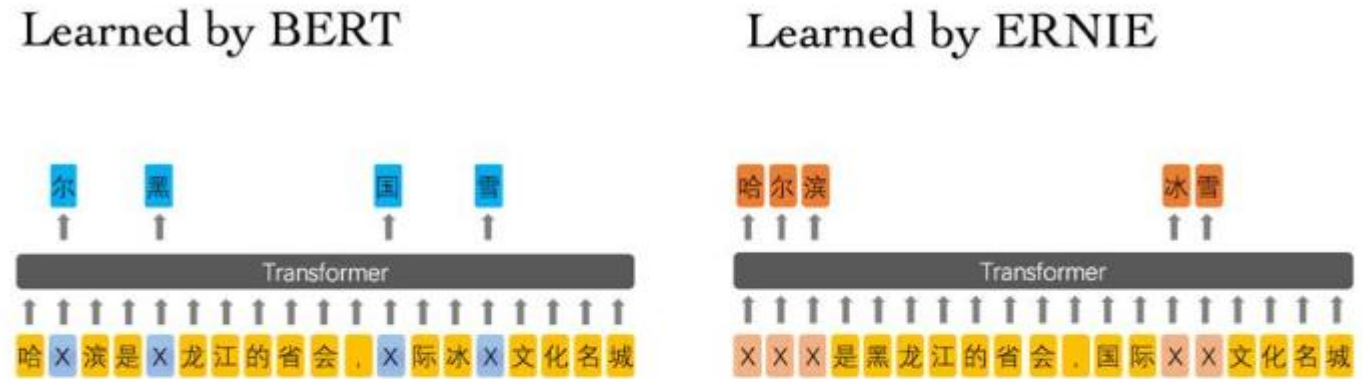
ERNIE 1.0 VS BERT

1. NO Difference in model Architecture: base model contains 12 transformer layers, 12 self-attention heads and 768-dimensional of hidden size
2. Different masking strategy to include knowledge structure
3. Heterogeneous corpora for pre-training
4. DLM based pretraining – alternating between DLM and MLM style pretraining

ERNIE: Masking in BERT Good Enough?

- When Baidu researchers realized they needed to tweak the Masking algorithm in BERT to accommodate the Chinese language.
- In English, the word serves as the semantic unit
  - a word pulled completely out of context still contains meaning.
- The same cannot be said for characters in Chinese.
  - certain characters do have inherent meaning fire (火, *huǒ*), water (水, *shuǐ*), or wood (木, *mù*)
  - The character 灵 (*líng*), for example, can either mean clever (机灵, *jīlíng*) or soul (灵魂, *línghún*), depending on its match.
- They developed masking that hides strings of characters rather than single ones.

ERNIE: Masking in Chinese



哈尔滨是黑龙江的省会，国际冰雪文化名城

Harbin is the capital of Heilongjiang Province, an international ice and snow cultural city.

ERNIE:

Problems

- Is random masking good enough?
- What about new word `APPLE VISION PRO`

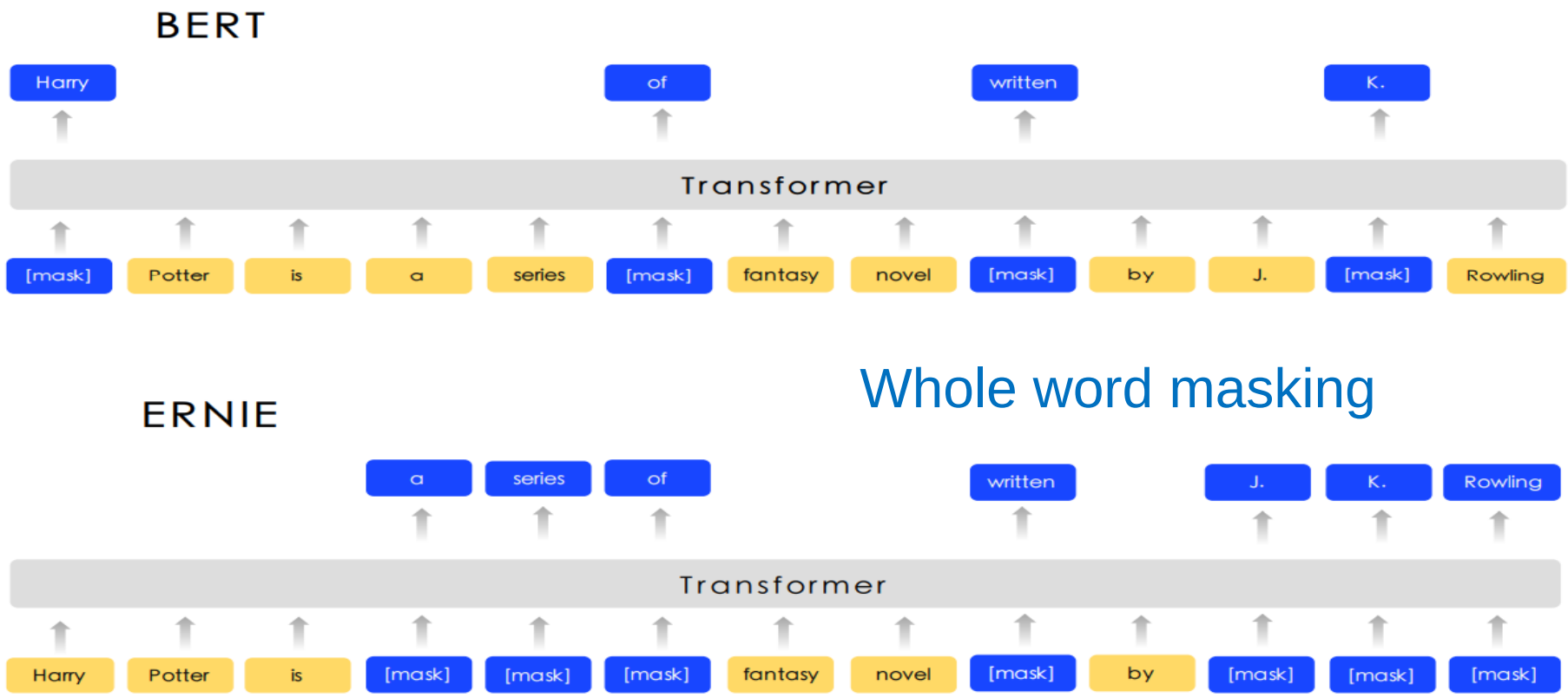


Figure 1: The different masking strategy between BERT and ERNIE

ERNIE: Multi-Stage Knowledge Masking Strategy

Sentence	Harry	Potter	is	a	series	of	fantasy	novels	written	by	British	author	J.	K.	Rowling
Basic-level Masking	[mask]	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	J.	[mask]	Rowling
Entity-level Masking	Harry	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]
Phrase-level Masking	Harry	Potter	is	[mask]	[mask]	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]

- **Basic-Level Masking**: A sentence as a sequence of basic language unit (BLU), for English, BLU is word, and for Chinese, BLU is Chinese Character.
- In the training process, randomly mask 15 percent BLU, and using other BLU in the sentence as inputs, and train a transformer to predict the mask units.
- **Phrase-Level Masking:** Phrase is a small group of words or characters together acting as a conceptual unit. Using NLP chunking tools we can get noun phrases and verb phrases.
- In the training process, randomly select a few phrases in the sentence, mask and predict all the basic units in the same phrase
- **Entity-Level Masking:** Name entities contain persons, locations, organizations, products, etc. Using named entity extraction algorithms the named entities in a sentence are identified, and then mask and predict all slots in the entities.

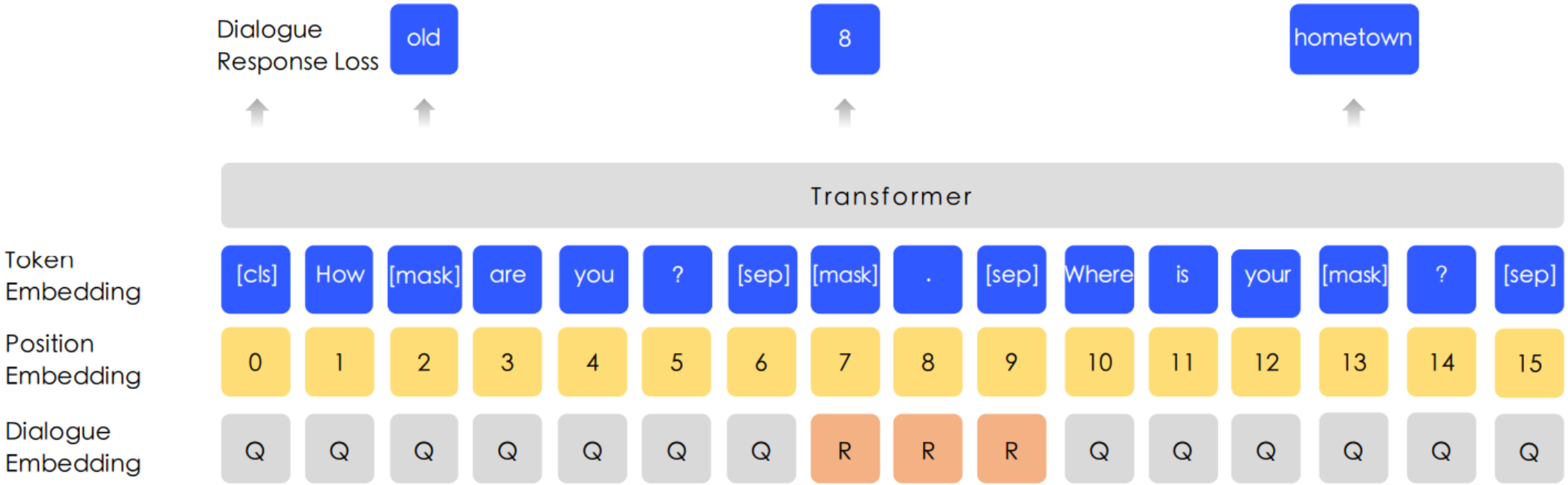
ERNIE: Heterogenous corpus for Pretraining

- ERNIE adopts Heterogeneous corpus for pretraining. Number of sentences shown in parenthesis.

1. Chinese Wikipedia (21M)
2. Baidu Baike(51M) : contains encyclopedia articles written in formal languages, which is used as a strong basis for language modeling
3. Baidu news (47M) : provides the latest information about movie names, actor names, football team names, etc.
4. Baidu Tieba (54M) : an open discussion forum like Reddits, where each post can be regarded as a dialogue thread. Tieba corpus is used in **Dialogue Language Model(DLM)** task

ERNIE: Heterogenous corpus for Pretraining

- The model architecture of DLM task is compatible with that of the MLM(masked LM) task, thus it is pre-trained alternatively with the MLM task.



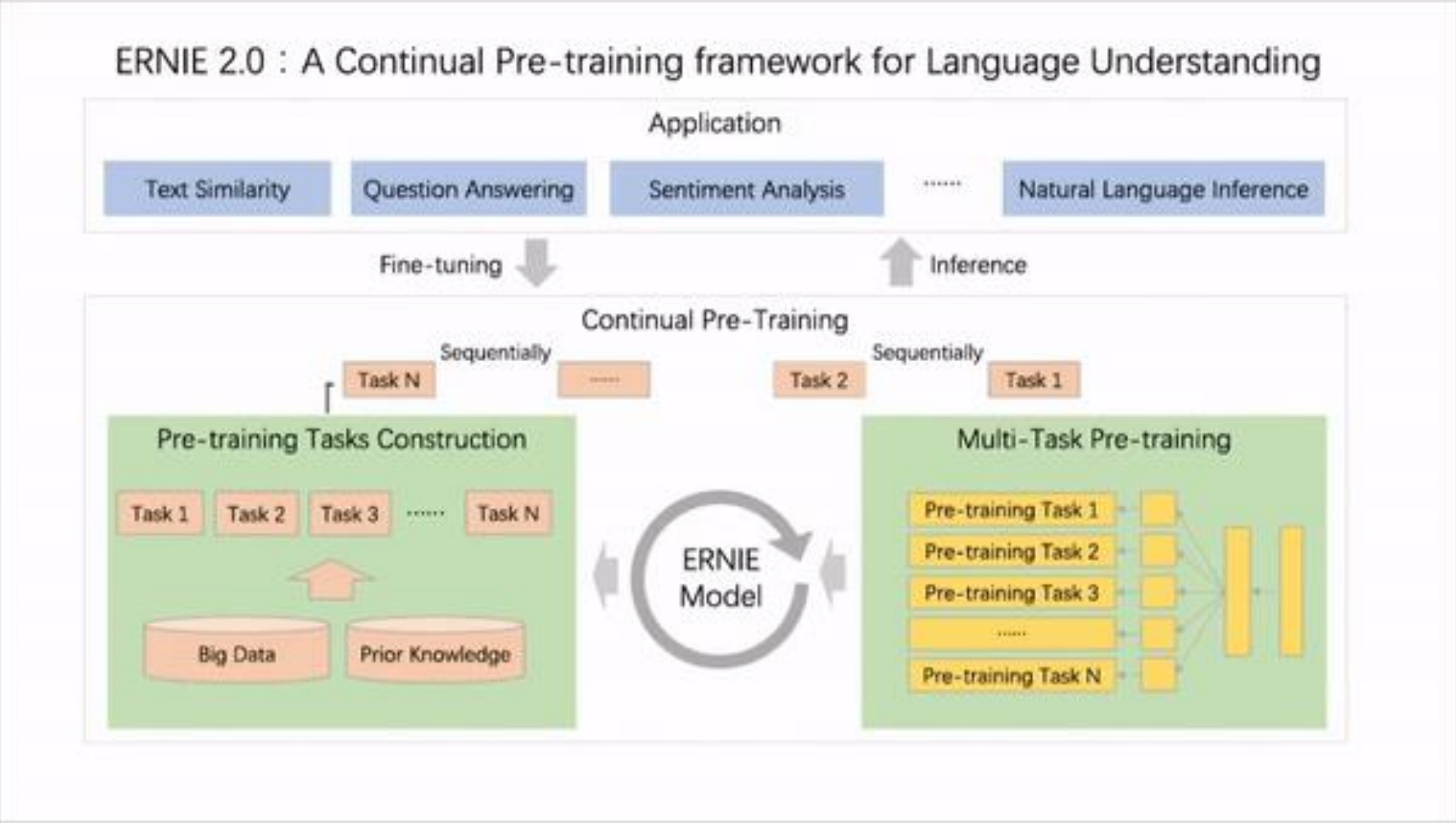
ERNIE 1.0: Results

Table 1: Results on 5 major Chinese NLP tasks

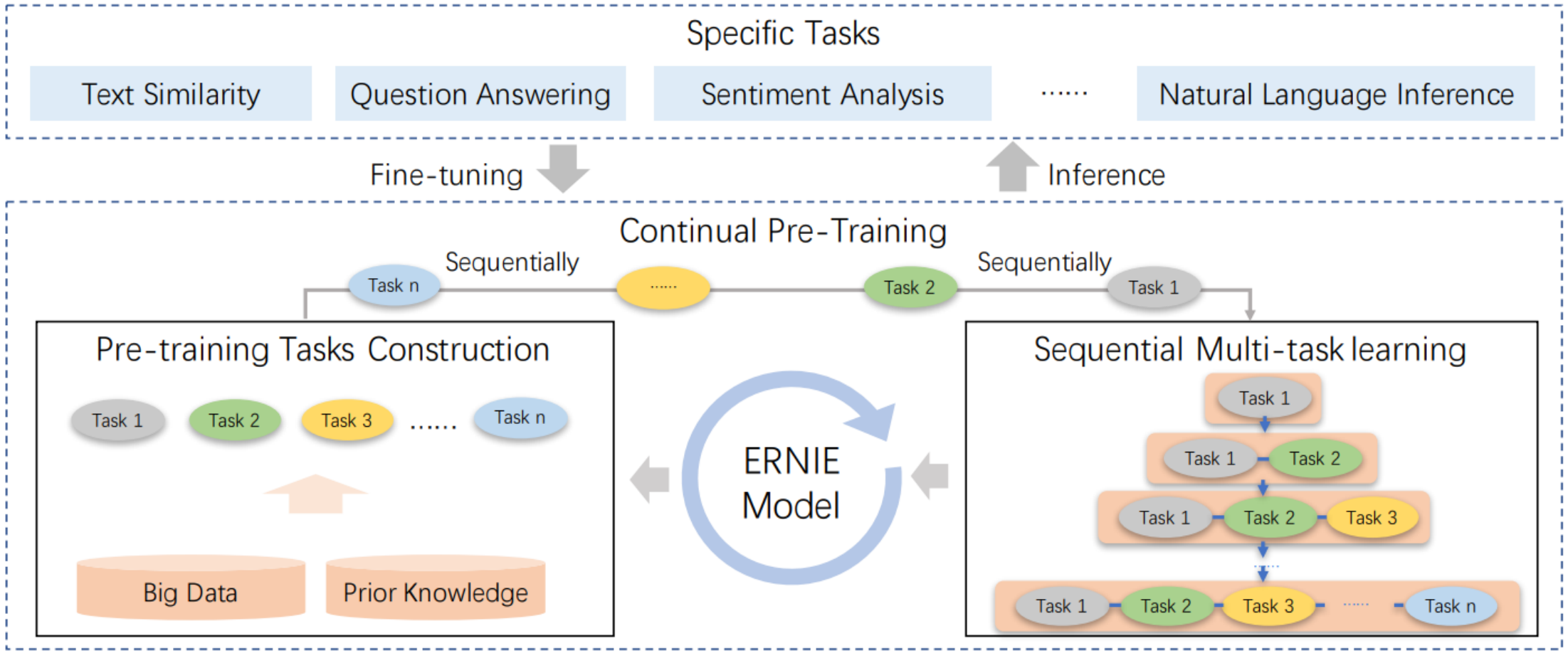
Task	Metrics	Bert		ERNIE	
		dev	test	dev	test
XNLI	accuracy	78.1	77.2	79.9 (+1.8)	78.4 (+1.2)
LCQMC	accuracy	88.8	87.0	89.7 (+0.9)	87.4 (+0.4)
MSRA-NER	F1	94.0	92.6	95.0 (+1.0)	93.8 (+1.2)
ChnSentiCorp	accuracy	94.6	94.3	95.2 (+0.6)	95.4 (+1.1)
nlpcc-dbqa	mrr	94.7	94.6	95.0 (+0.3)	95.1 (+0.5)
	F1	80.7	80.8	82.3 (+1.6)	82.7 (+1.9)

Both the knowledge integration and pre-training on heterogeneous data enable the model to obtain better language representation

ERNIE 2.

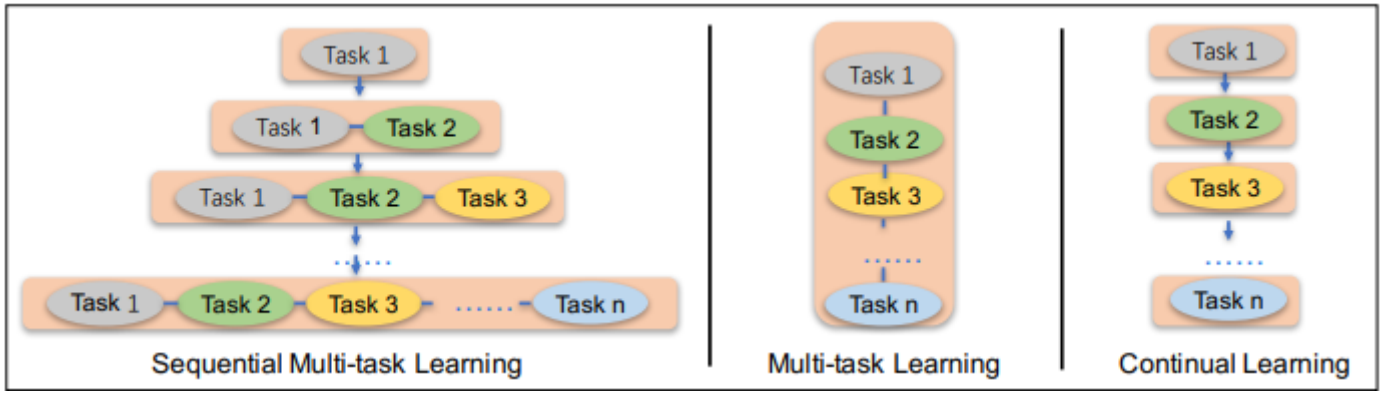

ERNIE 2.0 : A Continual Pre-training framework for Language Understanding
Application
Text Similarity
Question Answering
Sentiment Analysis
.....
Natural Language Inference
Fine-tuning
Inference
Continual Pre-Training
Sequentially
Task N
.....
Sequentially
Task 2
Task 1
Pre-training Tasks Construction
Task 1
Task 2
Task 3
.....
Task N
Big Data
Prior Knowledge
ERNIE Model
Multi-Task Pre-training
Pre-training Task 1
Pre-training Task 2
Pre-training Task 3
.....
Pre-training Task N

ERNIE 2.0 Framework



ERNIE 2.0 Framework : Continual Multi-task Learning framework

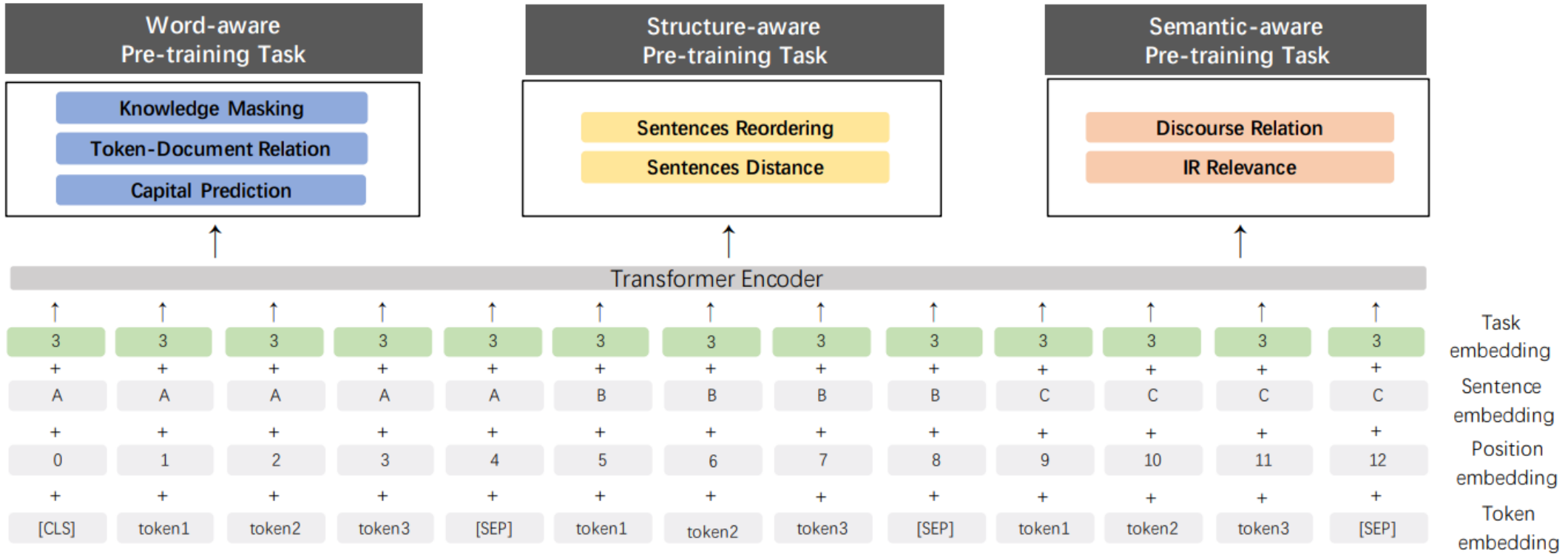
1. Multi-task learning from scratch could train multiple tasks at the same time, **Demerit:** All customized pre-training tasks are prepared before the training could proceed.
2. Continual learning method trains the model with only one task at each stage. **Demerit:** it may forget the previously learned knowledge.



3. **Sequential Multi-task Learning:** Allocate each task N training iterations. Avoids forgetting the previously trained knowledge.

ERNIE 2.0 Framework : Continual Multi-task Learning framework

- The model uses a multi-layer Transformer like BERT
- **Task Embedding** The model feeds task embedding to represent the characteristic of different tasks : task id 0 to N. Each task id is assigned to one unique task embedding.



Key Takeaway

Contextual Understanding: Models like ELMo introduced deep contextualized representations, significantly enhancing word meaning comprehension in varied contexts.
Bidirectional Training: BERT's bidirectional MLM approach revolutionized model training, enabling a deeper grasp of language syntax and semantics.
Scalability and Efficiency: Advances by GPT-2, GPT-3, and ELECTRA demonstrate that larger, efficiently trained models can achieve remarkable language understanding and generation capabilities.
Specialization and Innovation: Innovations in models like ALBERT, RoBERTa, and UniLM highlight the trend towards specialized optimizations for efficiency and task unification.
Knowledge Integration: Models like ERNIE emphasize integrating external knowledge, improving the understanding of complex concepts beyond textual data.
Towards General AI: The evolution towards models like GPT-3 suggests a move towards General AI in NLP, capable of understanding and reasoning across a wide range of tasks with minimal tuning.