Natural Language Representation Learning & Understanding

苏勤亮

中山大学 计算机学院

suqliang@mail.sysu.edu.cn

What is Text Representation?

- Unlike images, natural language (textual data) does not appear in a numerical format by nature

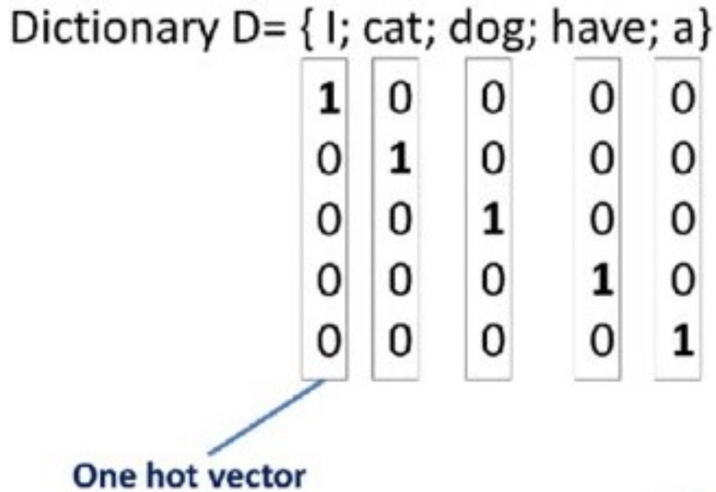
“Today is a good day”

- To learn from textual data, the first thing we need to do is to represent the textual data in a form that computers can process

Outline

- One-Hot Representation
- Word2Vec Representation
- BERT-Based Representation
- Fine-tuning BERT for Downstream Tasks

- The simplest way to represent textual data is to encode every word in the vocabulary set by a one-hot vector, *e.g.*,



- After that, sentences are represented as the concatenation of the one-hot vectors

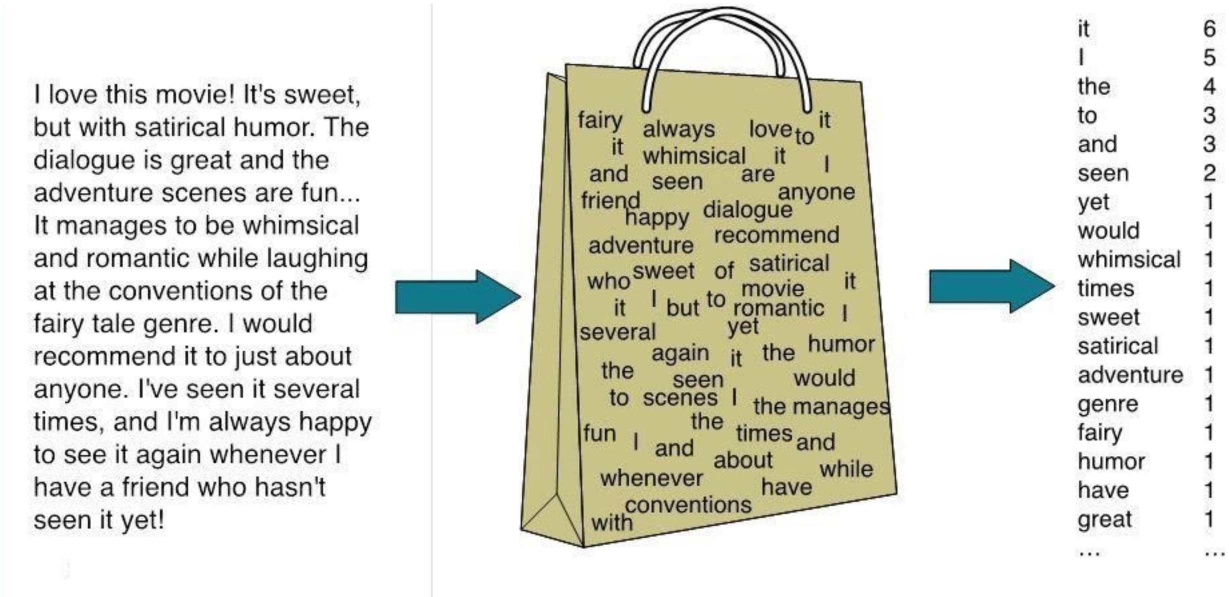
  For instance, "I have a dog" can be represented as a matrix on the right

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

➢ Representing sentences as a matrix is cumbersome, especially when the sentences are long

Bag-of-Words (BOW)

- Bag-of-Word (BOW) discards the word order information, only recording the count of each word in a document



- Cons: Each word is deemed as equal importance. However, *the amount of information conveyed by different words is not equal*

  For example, words like ‘the’, ‘a’ are much less informative than words like ‘classroom’, ‘football’ *etc.*

TFIDF

- Term Frequency–Inverse Document Frequency (TFIDF) attempts to reflect the unequal importance of different words
- Specifically, it is computed as the product of two statistics, *i.e.*,

$$\boxed{tfidf(w, d, \mathcal{D}) = tf(w, d) \times idf(w, \mathcal{D})}$$

  – w denote the word; d denotes the d-th document, $\mathcal{D}$ denotes the corpus (a collection of documents)

  – $tf(w, d)$ denotes the frequency of word w in document d

$$tf(w, d) = \frac{\#\, of\ word\ w\ in\ d}{\#\, of\ all\ words\ in\ d}$$

  – $idf(w, \mathcal{D})$ the log inverse fraction of the documents containing word w in the corpus $\mathcal{D}$

$$idf(w, \mathcal{D}) = \log \frac{\#\, of\ documents\ in\ \mathcal{D}}{\#\, of\ documents\ containing\ word\ w\ in\ \mathcal{D}}$$

- $idf(w, \mathcal{D})$ measures how much information the word w contains. The larger the value $idf(w, \mathcal{D})$ is, the more informative the word w is

- Thus, TFIDF feature $tfidf(w, d, \mathcal{D}) = tf(w, d) \times idf(w, \mathcal{D})$ accounts for both of a word's frequency in a document and its informativeness in the corpus

$$\texttt{tfidf}(t, d, D) = \texttt{tf}(t, d) \cdot \texttt{idf}(t, D)$$

		blue	bright	can	see	shining	sky	sun	today
Document 1	1	**0.301**	0	0	0	0	0.151	0	0
Document 2	2	0	0.0417	0	0	0	0	0.0417	**0.201**
Document 3	3	0	0.0417	0	0	0	**0.100**	0.0417	0
Document 4	4	0	0.0209	**0.100**	**0.100**	**0.100**	0	0.0417	0

Each row means the TFIDF feature of a document

Limitations

- The limitations of BOW and TFIDF document representations
  - Unable to reflect the *semantic similarities* of words
  - Do not contain any *word-order information*
  - The dimension is very *high*
  - It is very *sparse*

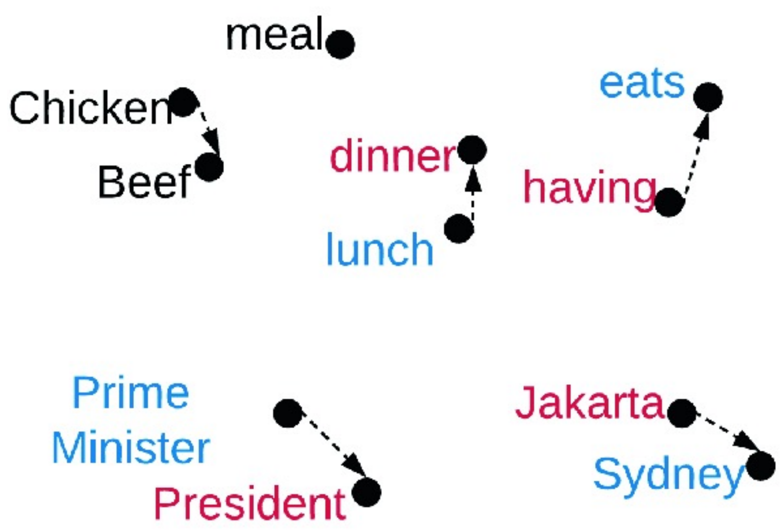
But much better than one-hot representatons

Outline

- One-hot Representation
- Word2Vec Representation
- BERT-Based Representation
- Fine-tuning BERT for Downstream Tasks

Word2Vec Word Embedding

- Goal: Represent words by real-valued vectors which have the characteristics
  1) being *dense* (low dimensional)
  2) reflecting the *semantic similarities* among words



- Sentence embedding can be constructed from word embedding subsequently

How to Learn Word2Vec Embedding?

- Observation: Semantic similarities of words are implicitly reflected in sentences existing in the world

  For instance, if two words often appear simultaneously, they may preserve similar meaning

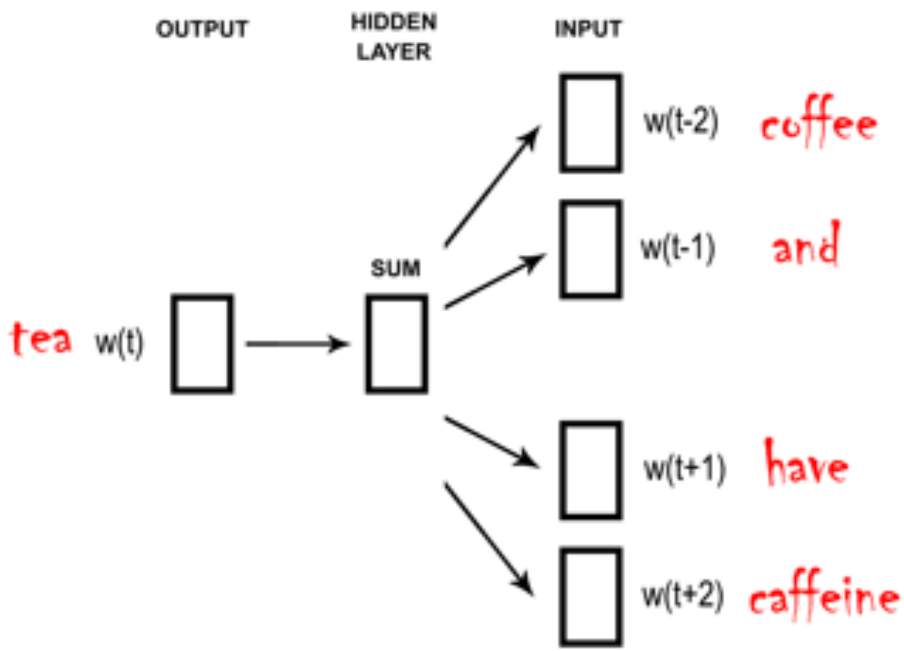
A cup of **tea**
A cup of **coffee**
Tea or **coffee**?
Coffee and **tea** have caffeine
Let's go for a **coffee**
Let's get a **tea**
Coffee vs **Tea**: Which is Best?
I avoid adding sugar to my **tea**
I drink **coffee** with two spoons of sugar

- Thus, *semantics-preserving* embedding could be learned from the huge amount of sentences existing in the world

Skip-gram

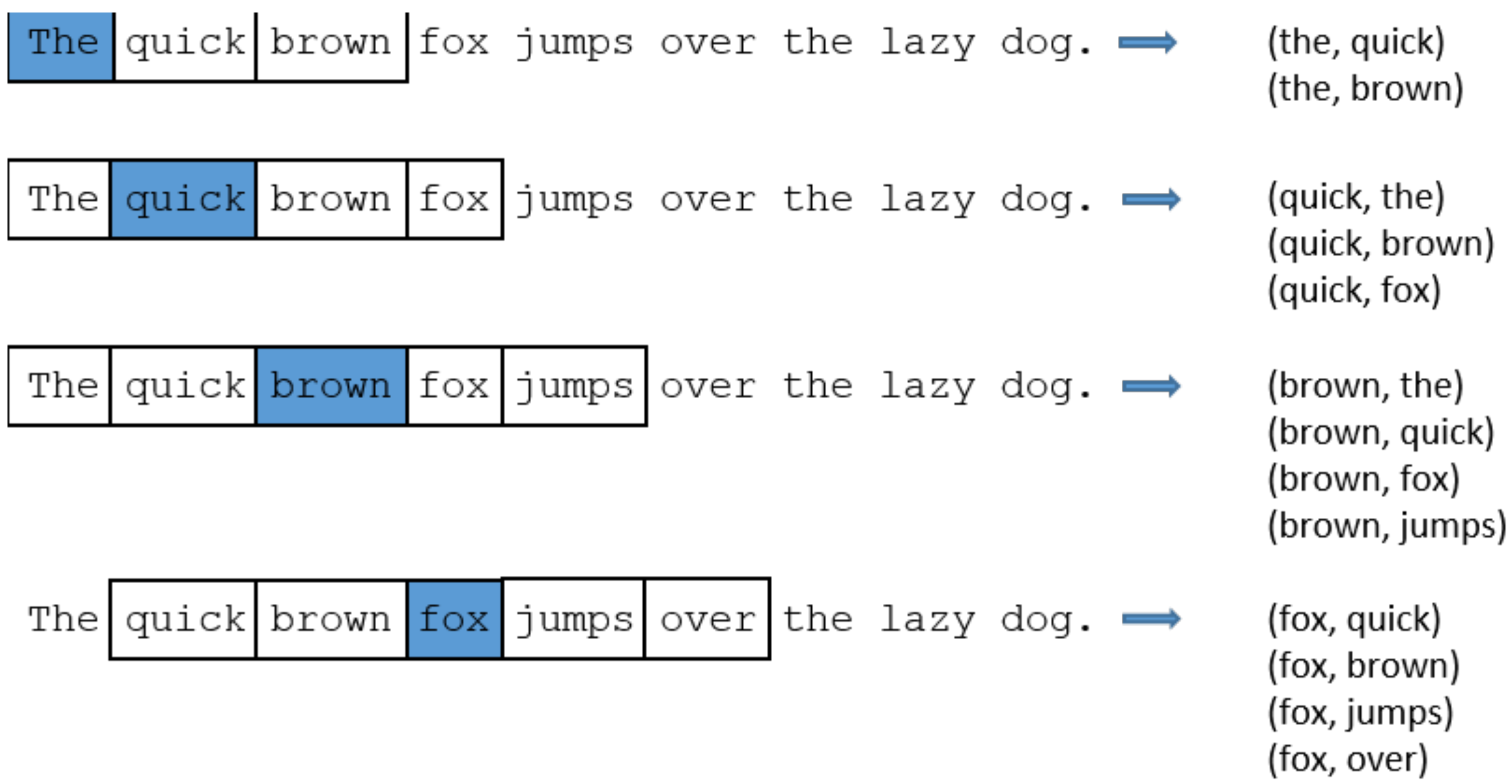
- Two basic methods
  1) Skip-grams
  2) Continuous Bag-of-Words (CBOW)

- Skip-gram: predicting the context using the center word



- Initialize the embedding of each word by a random vector $u \in \mathbb{R}^m$
- Then, using the t-th word w_t to predict its left and right words. The objective function is

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

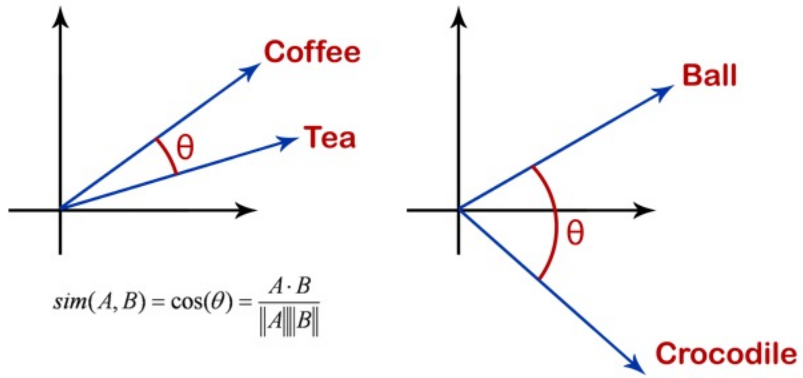


- The prediction probability $P\left(w_{t+j} \mid w_t; \theta\right)$ is modeled as

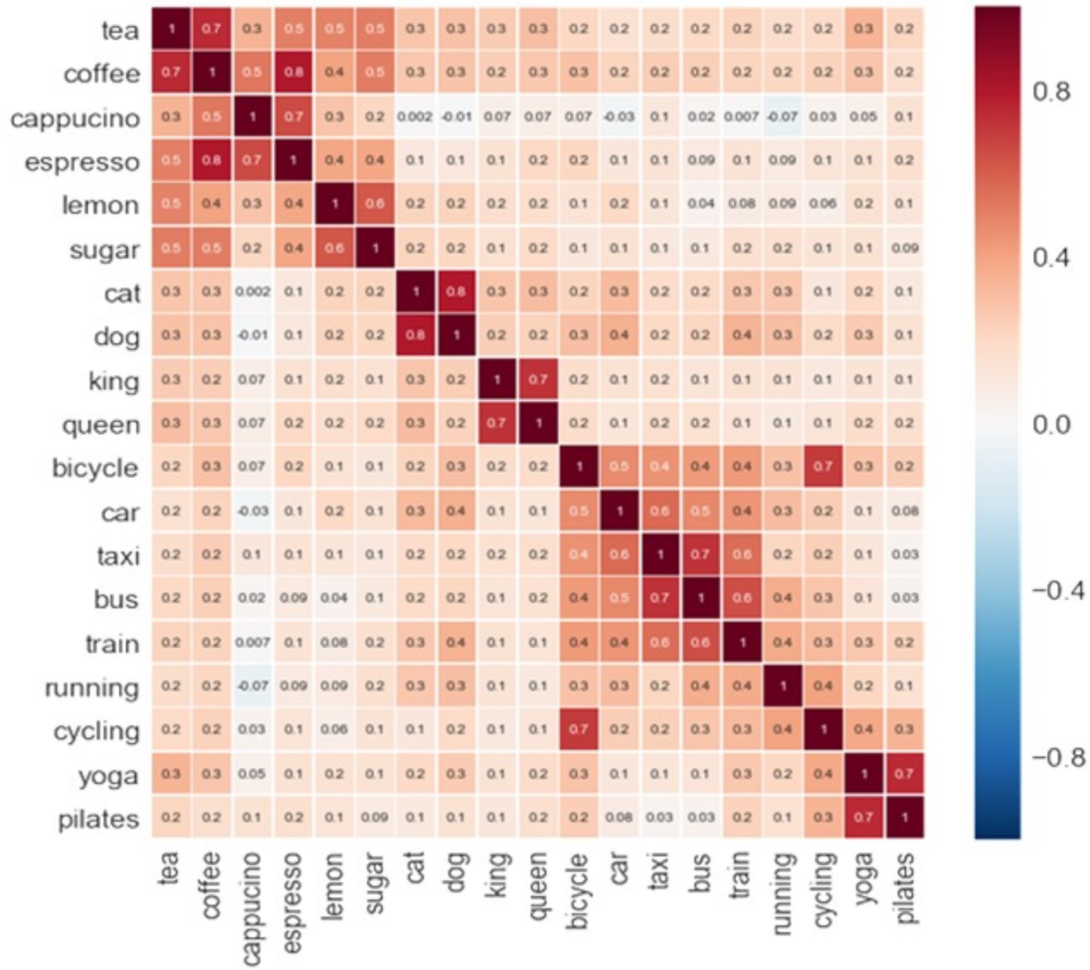
$$P(w_{t+j} \mid w_t; \theta) = \frac{\exp\left(u_{w_t}^T u_{w_{t+j}}\right)}{\sum_{w \in \mathcal{V}} \exp\left(u_{w_t}^T u_w\right)}$$

  – $\mathcal{V}$ is the set of vocabulary words
  – u_w denotes the embedding of word w, which is to be optimized

- After training on a huge corpus, *semantic similarities of words are reflected in the cosine similarities of their embeddings*

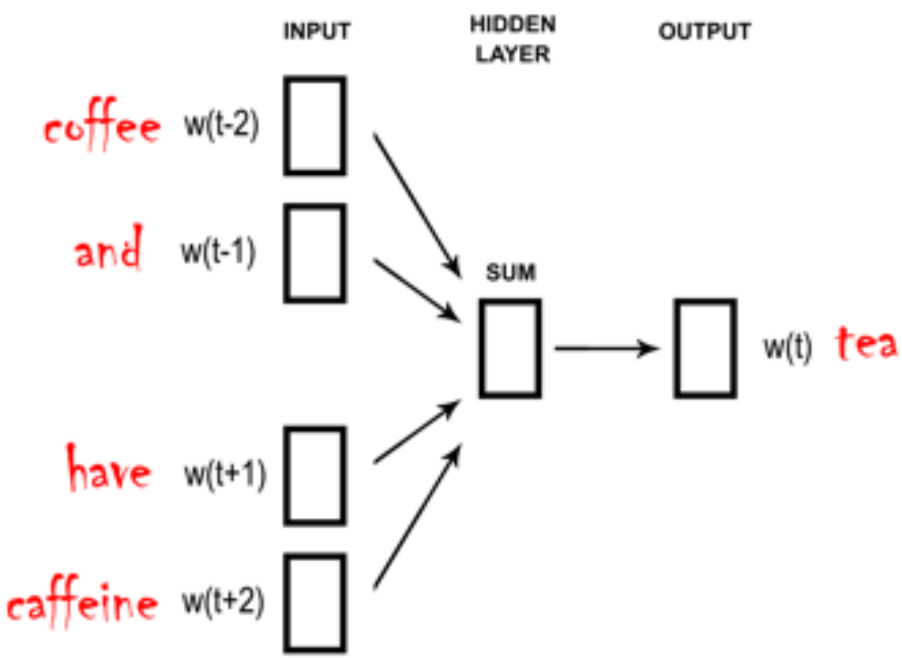


- Example of cosine similarities evaluated on word embeddings that are trained on a large corpus



Continuous Bag-of-Words

- Different from the skip-grams, CBOW predicts the center word using left and right words



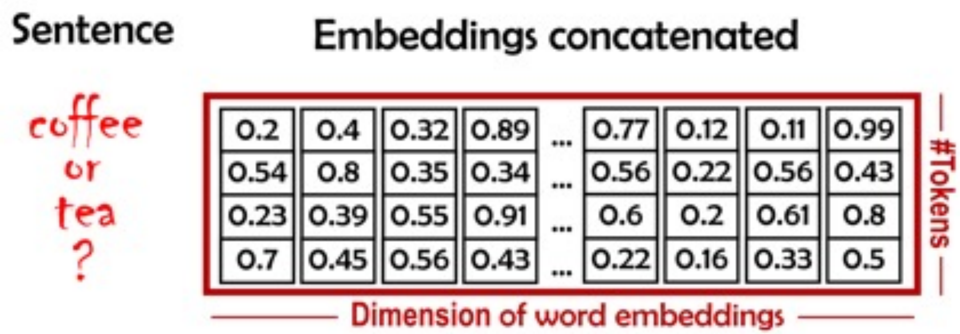
Sentence Embedding

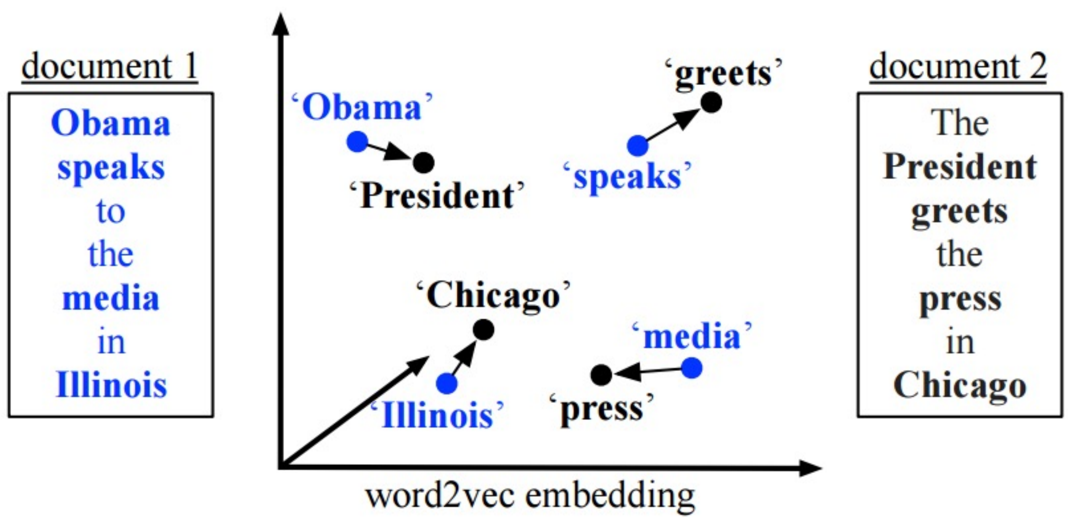
- Sentence embedding can be obtained from word emebddings in many different ways

1) Average

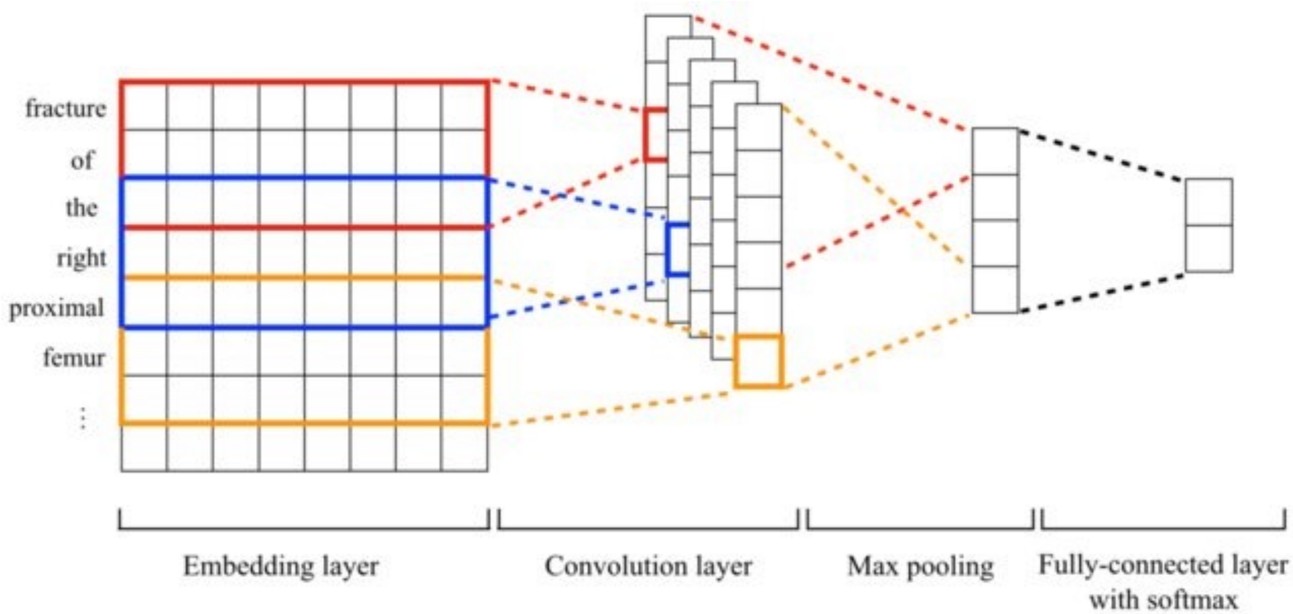
Sentence	Embeddings								
coffee	0.2	0.4	0.32	0.89	...	0.77	0.12	0.11	0.99
or	0.54	0.8	0.35	0.34	...	0.56	0.22	0.56	0.43
tea	0.23	0.39	0.55	0.91	...	0.6	0.2	0.61	0.8
?	0.7	0.45	0.56	0.43	...	0.22	0.16	0.33	0.5
Average	0.42	0.51	0.45	0.64	...	0.54	0.17	0.4	0.68

2) Concatenation



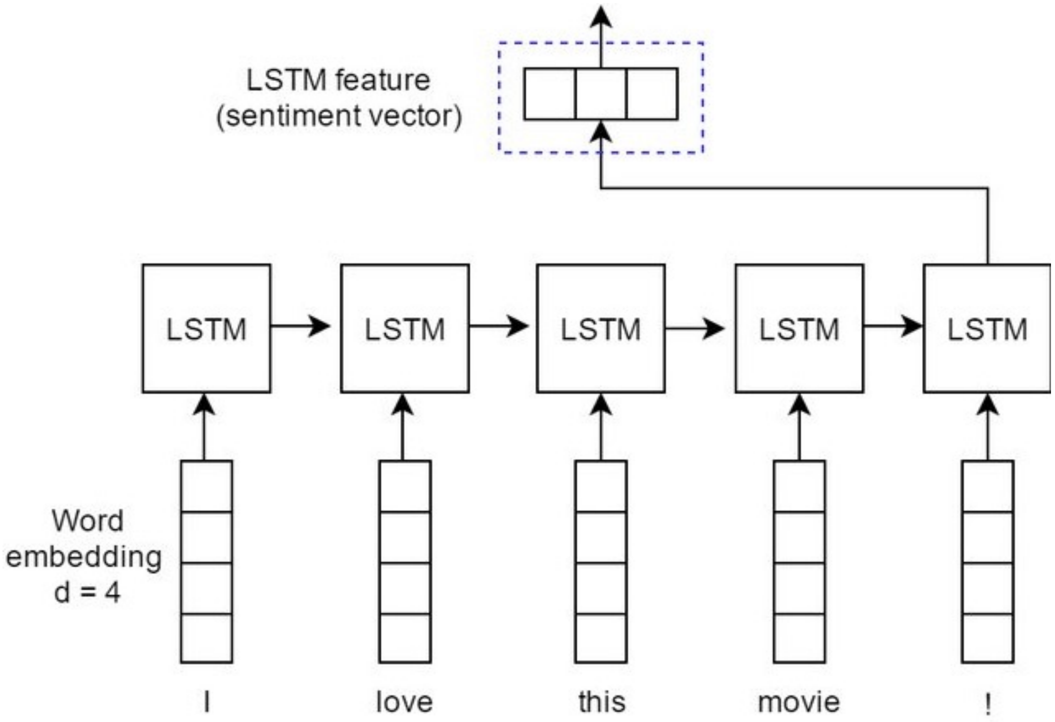


3) Extracting sentence embedding with CNNs (Text CNN)



The parameters of CNN are optimized using a downstream task

4) Extracting sentence embedding with RNN



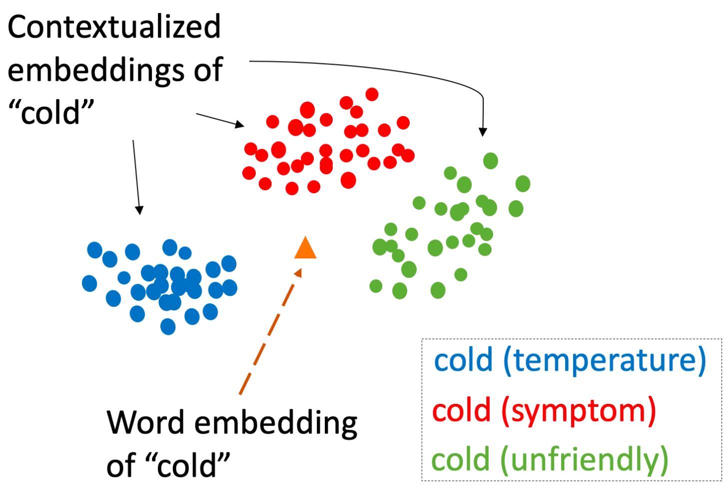
The parameters of RNN are optimized using a downstream task

Contextualized Word Embedding

- Issues of Word2Vec embedding
  - Word2Vec assigns every word with a fixed embedding
  - However, words often exhibit different meanings when placed under different contexts, *e.g.*,

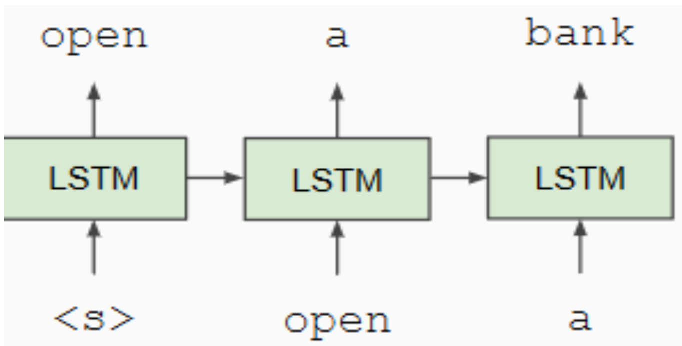
    1) open a *bank* account 2) on the river *bank*

- The embedding of a word should *vary w.r.t. the context* under which it is placed

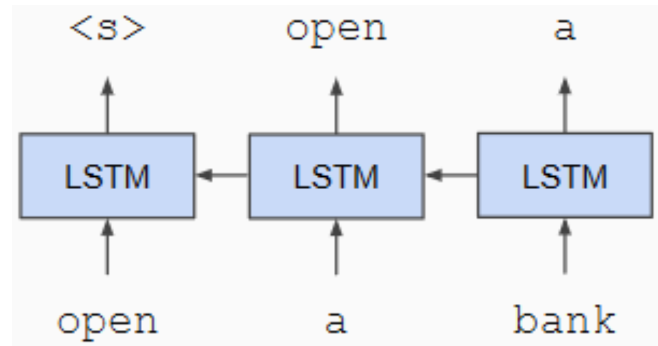


- ELMo

1) First, train a bidirectional RNN on a large corpus
2) Then, pass interested sentences through the pre-trained RNN

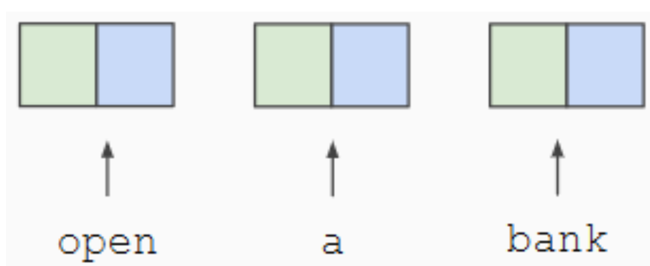


Left-to-right



Right-to-left

3) The final word embeddings are obtained as the concatenation of hidden states of the pre-trained RNN



Outline

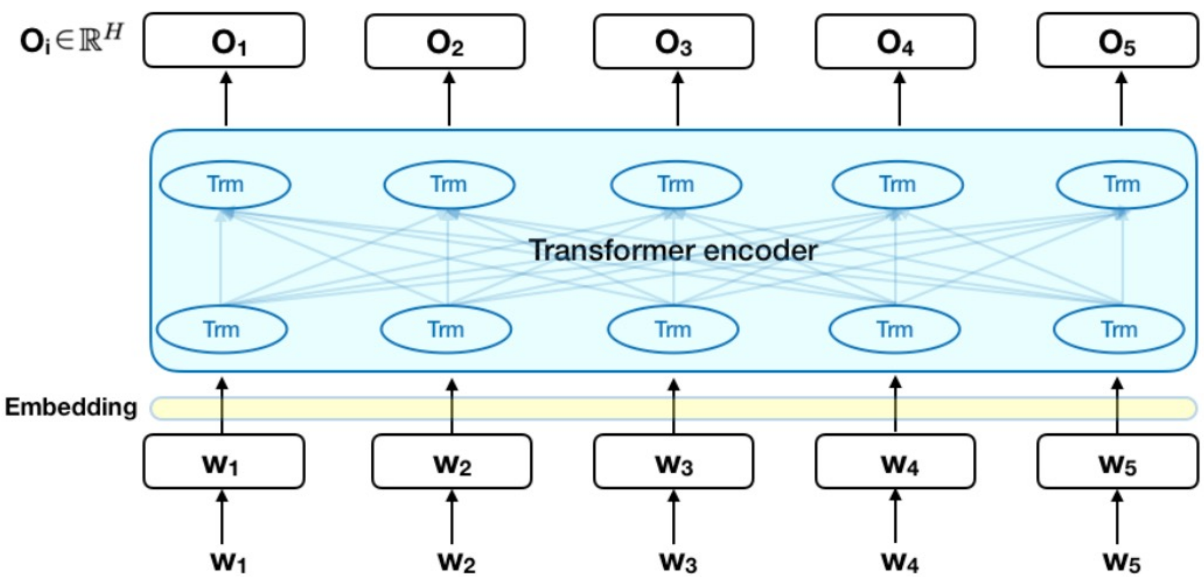
- One-Hot Representation
- Word2Vec Representation
- BERT-Based Representation
- Fine-tuning BERT for Downstream Tasks

- Issues of ELMo embedding

  1) RNN is difficult to capture long-term dependencies of words in documents

  2) RNN only admits sequential execution, making it difficult to exploit the *parallel computation resources* in GPUs

- Instead of using left-to-right or right-to-left sequential structures, Bidirectional Encoder Representations from Transformers (BERT) adopts a transformer-based 'fully-connected' structure



Pre-Training Tasks

- The model is pre-trained on an extremely large corpus

1) Pre-Training Task 1: Masked Language Model

   - For every input sequence, randomly select 15% of tokens
     - Replace 80% of the selected tokens with the [MASK] token

       went to the store $\longrightarrow$ went to the [MASK]

     - Replace 10% of the selected tokens with a random word

       went to the store $\longrightarrow$ went to the running

     - Keep the rest 10% of the selected token untouched

       went to the store $\longrightarrow$ went to the store

- Objective: training the BERT model to predict the true tokens at the positions of masked tokens

store gallon

the man went to [MASK] to buy a [MASK] of milk

2) Pre-Training Task 2: Next Sentence Prediction (NSP)

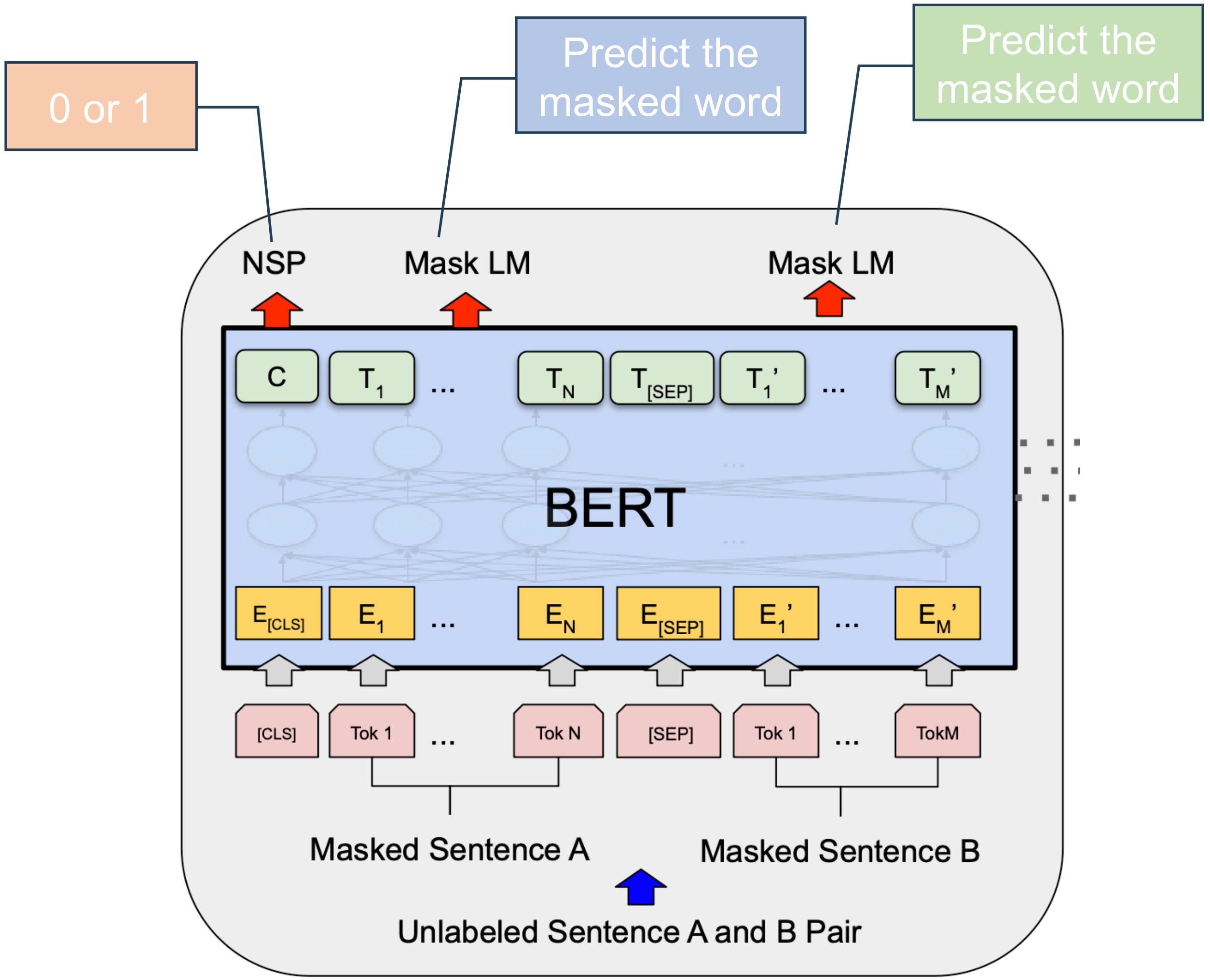
Predict whether sentence B is the next sentence of sentence A

[CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]

Sentence A Sentence B

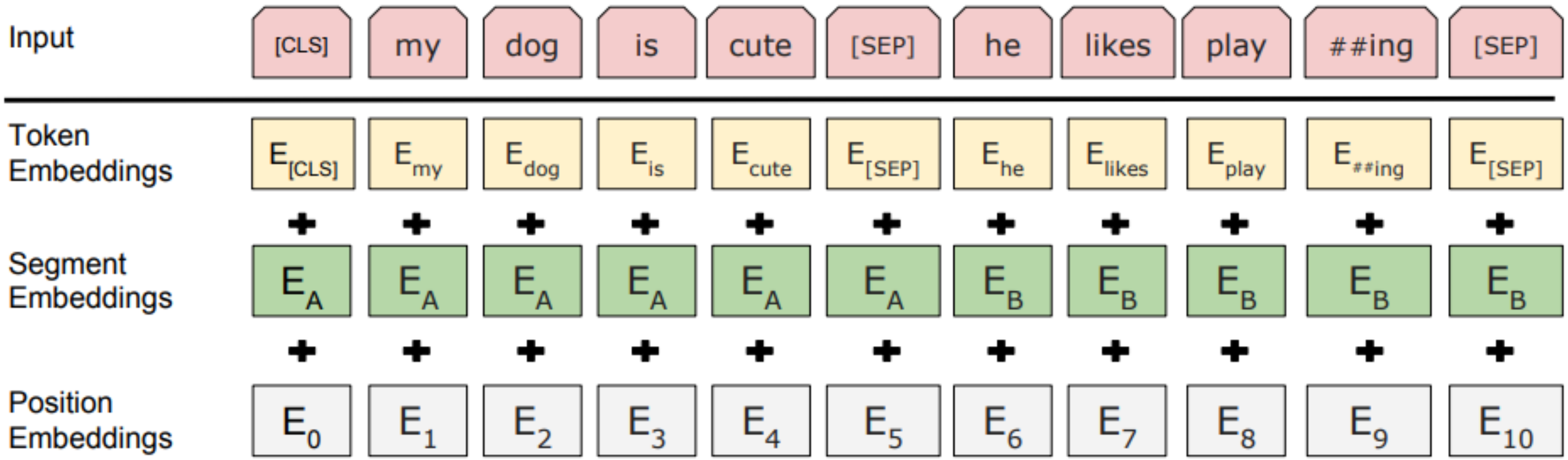
- 50% chance of letting sentence B to be the true next sentence of sentence A

- Putting the pre-training tasks together



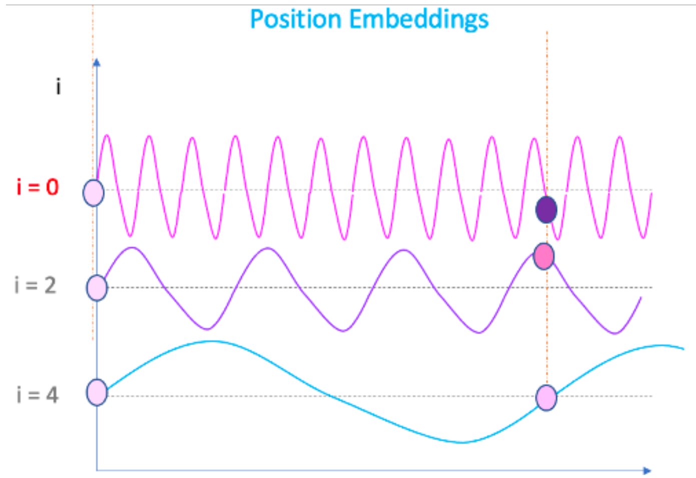
- Input to the BERT model

Summing the three token embedding as the final embedding input to the model



- Position embedding

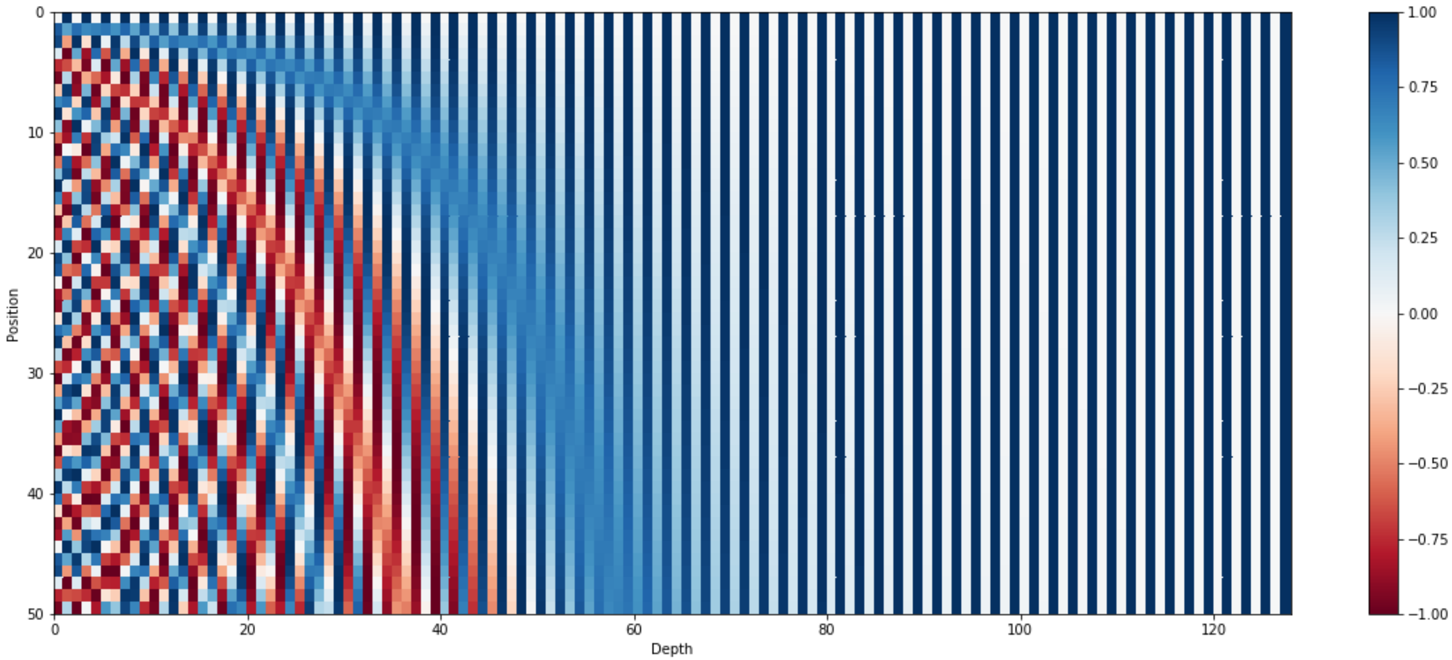
$$PE(pos, i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$



Position of word in a sentence

i

Word Position

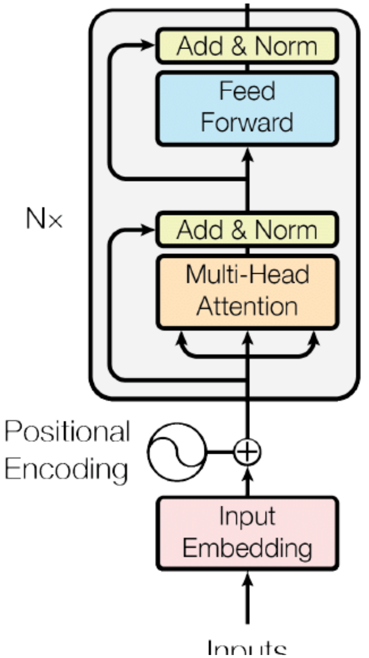


- Training details
  - Training Corpus: Wikipedia (2.5B) + BooksCorpus (0.8B)
  - Training sequence length: 512 word pieces
  - Batch size: 128K
  - Trained for 1M steps

- Size of BERT models

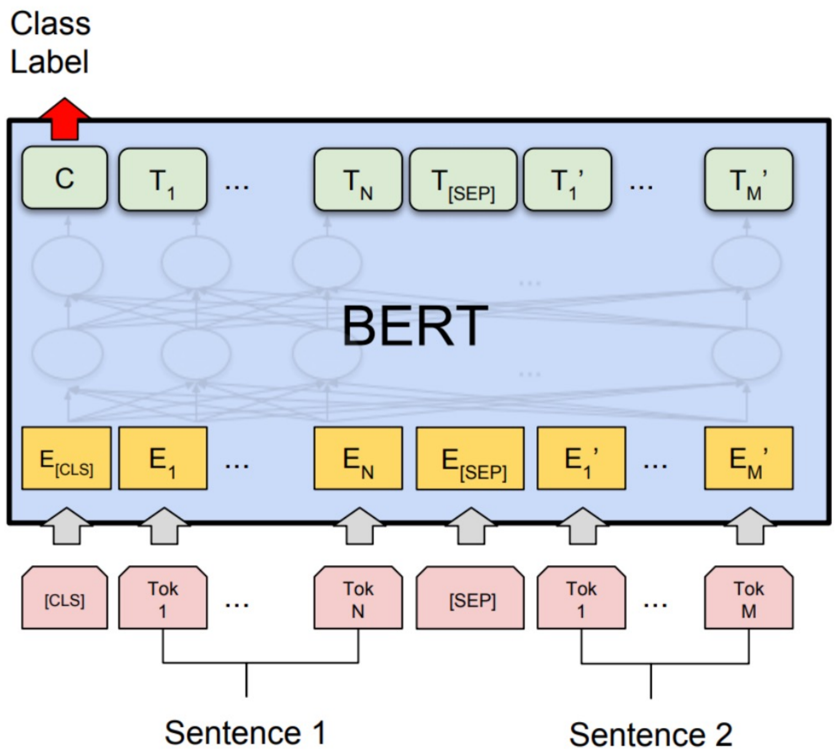
  - BERT-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters

  - BERT-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters



Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100 GPU
GPT-3	96	12,288	96	175B	694GB	?
Gopher	80	16,384	128	280B	10.55 TB	4096x TPUv3 (38 days)

- To obtain word and sentence embedding, pass the interested sentences through the pre-trained BERT model



- Outputs from BERT are the contextualized embeddings of words and sentences

Outline

- One-Hot Representation
- Word2Vec Representation
- BERT-Based Representation
- Fine-tuning BERT for Downstream Tasks

GLUE: general language understanding evaluation benchmark, including 6 sentence pair and 2 single-sentence tasks

- Sentence-level tasks

MNLI

Premise: A soccer game with multiple males playing

Hypothesis: Some men are playing a sport

(entailment, contradiction, neutral)

QQP

Q1: Where can I learn to invest in stocks?

Q2: How can I learn more about stocks?

(duplicate, not duplicate)

SST2

Rich veins of funny stuff in this movie

(positive, negative)

- Token-level tasks

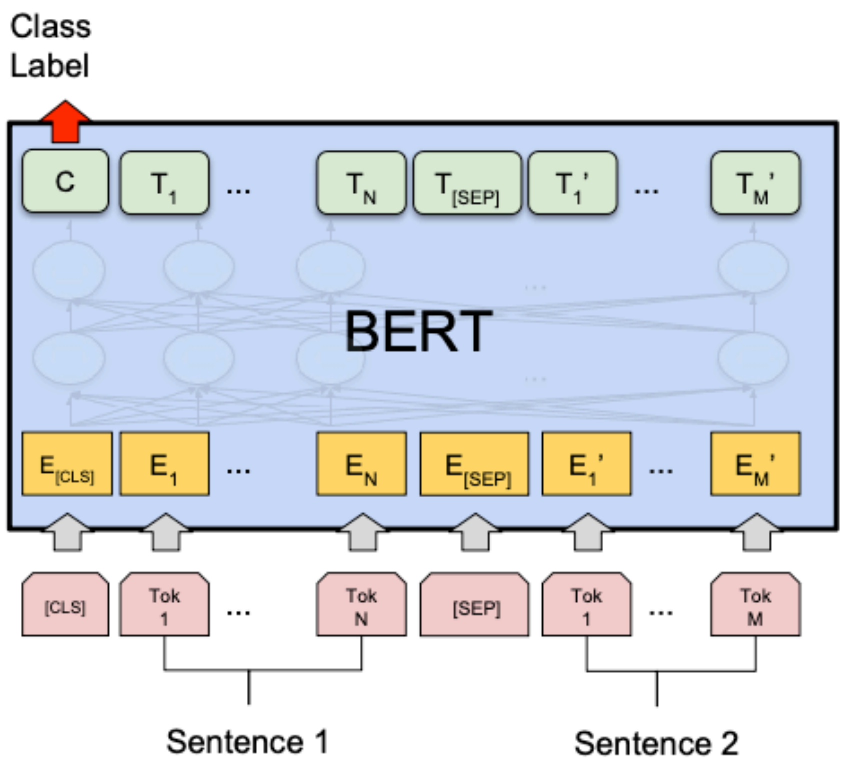
  - Question answering (SQuAD)

```
Question: The New York Giants and the New York Jets play
at which stadium in NYC ?
Context: The city is represented in the National Football
League by the New York Giants and the New York Jets ,
although both teams play their home games at MetLife
Stadium in nearby East Rutherford , New Jersey , which
hosted Super Bowl XLVIII in 2014 .
                                    (Training example 29,883)
```

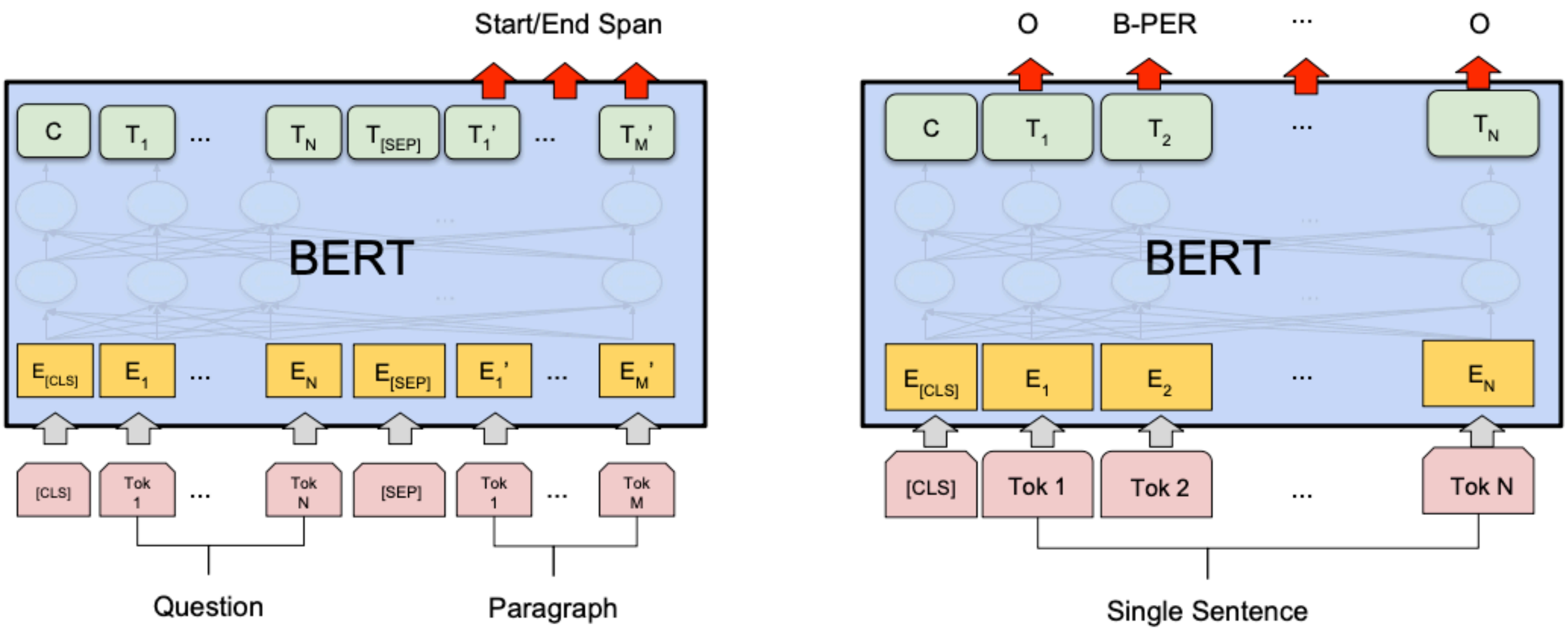
  - Named entity recognition

John	Smith	lives	in	New	York
↓	↓	↓	↓	↓	↓
B-PER	I-PER	O	O	B-LOC	I-LOC

- Fine-tuning BERT
  - For sentence pair tasks, use [SEP] to separate the two segments with segment embeddings
  - Add a linear classifier on top of [CLS] representation



- For token-level prediction tasks, add linear classifier on top of hidden representations



- Experimental results: GLUE

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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**

- Ablation study

layers | hidden size | # of heads

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

- Larger model always leads to better performance

BERT Variant

- RoBERTa (Liu et al. 2019)
  - Trained on 10x data & longer time, no the NSP pretraining task
  - Better performance than BERT

- ALBERT (Lan et al. 2019)
  - Increasing model sizes by sharing model parameters across layers
  - Less storage, much stronger performance but runs slower

- Multilingual BERT
  - Trained single model on 104 languages from Wikipedia. Shared 110k WordPiece vocabulary
- BERT extended to different domains
  - SciBERT, BioBERT, FinBERT, ClinicalBERT……
- Making BERT smaller to use
  - DistillBERT, TinyBERT

Thanks!