Winter semester 2021/22

344.175 VL: Natural Language Processing
Compositional Representation Learning

Navid Rekab-saz
navid.rekabsaz@jku.at

Institute of Computational Perception

JOHANNES KEPLER
UNIVERSITY LINZ

Institute of
Computational
Perception

Agenda

- Compositional representations
- Sentence embedding with sent2vec
- An overview on BERT

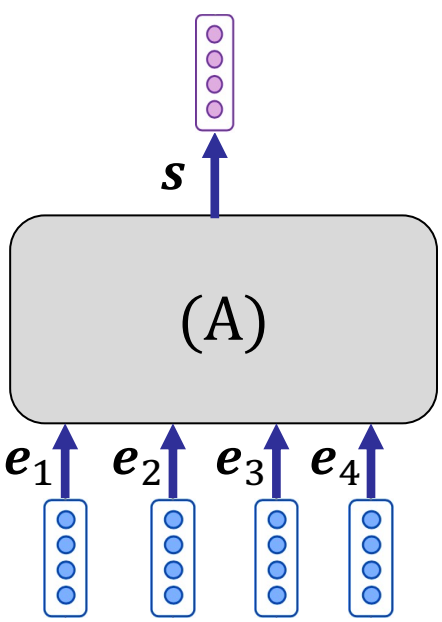
Agenda

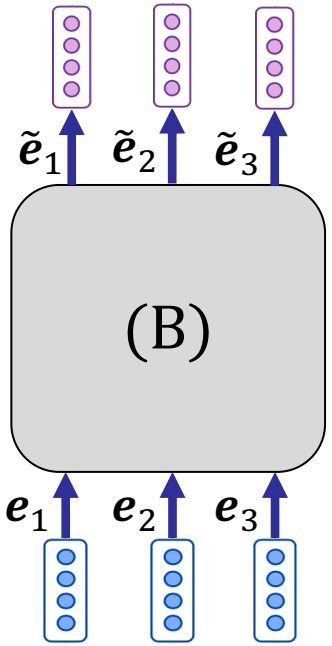
- **Compositional representations**
- Sentence embedding with sent2vec
- An overview on BERT

Compositional Representation

- Methods to compose (low-level) representations in order to create higher-level or richer representations

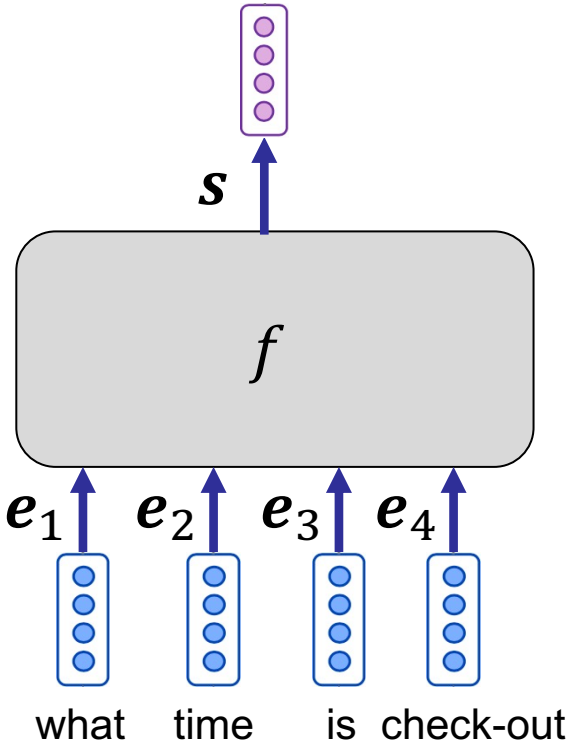
- Two common scenarios of compositional representation are …
  A. creating representations of high-level entities (i.e., n-gram, sentence, document) from lower-level entities (i.e., words, sub-words, characters)
  B. creating contextualized embeddings





Text embedding

- **Scenario A: composing a high-level embedding from low-level embeddings**
  - Directly applicable to many down-stream tasks, i.e., document classification, clustering, question/answering, information retrieval, machine translation
  - It can be seen as an *aggregation problem*



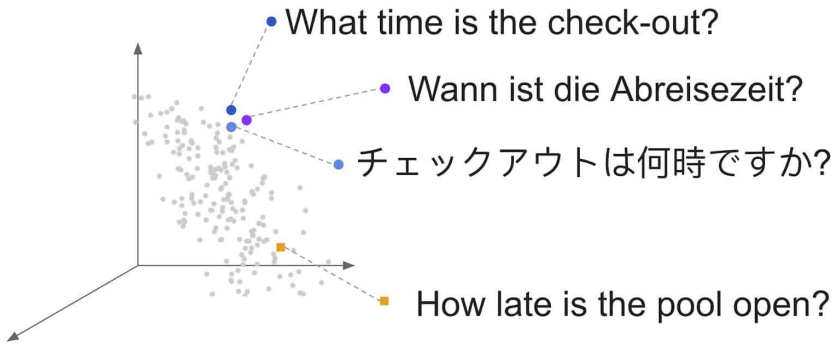


Image source: https://megagon.ai/blog/emu-enhancing-multilingual-sentence-embeddings-with-semantic-similarity/

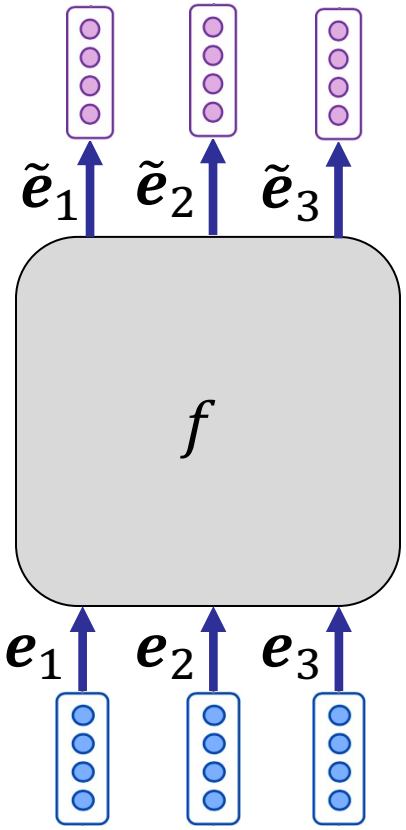
Contextualized Word Embeddings

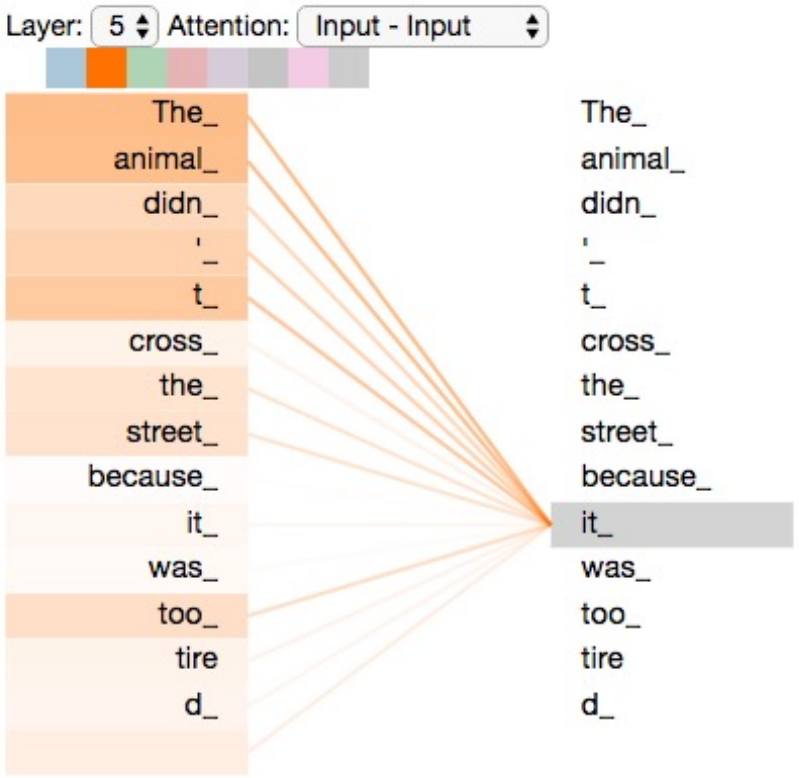
- Semantics of language entities (such as words) should be defined by considering the contexts, they appear in
  - Example of sense disambiguation when context is available:
    - *"eating an apple"* vs. "*share of the apple company*"

- Contextualized word embedding models create word/subword/token representations based on the semantics of the word as well as the context, the word appears in
  - The contextualized embedding of a word becomes different, depending on the context that the word appears in

Contextualized Embeddings

- **Scenario B: contextualizing embeddings**
  - Each embedding is enriched by looking at other embeddings in its context
  - The input is a list of word embeddings (in fact subword or token embeddings) and the output is a list of contextualized word embeddings, each provides the contextualized version of its corresponding input





Towards compositional embeddings

- Common architectural choices
  - Aggregating embeddings in feed-forward networks
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Transformers

- Common approach
  - Pre-training compositional embeddings using various language modeling objectives, like …
    - Predict the next word
    - Predict some words in the context
    - Predict the next sentence
  - Fine-tuning representations on down-stream tasks

Agenda

- Compositional representations
- **Sentence embedding with sent2vec**
- An overview on BERT

Sentence embedding

Problem definition

- Given a "sentence" S with length $|S|$, consisting of the words

$$v_1, v_2, \dots, v_{|S|}$$

with corresponding word vectors

$$\boldsymbol{e}_{v_1}, \boldsymbol{e}_{v_2}, \dots, \boldsymbol{e}_{v_{|S|}}$$

create the sentence embedding: $\boldsymbol{e}_S$

- "Sentence" here can refer to
  - Any sequence of words with any arbitrary length
  - An actual sentence in language

Sentence embedding

- First approach … simply average!

$$\boldsymbol{e}_S = \frac{1}{|S|} \sum_{v \in S} \boldsymbol{e}_v$$

  - As done in Assignment 2 and 3

- What are the possible limitations of this approach?

word2vec Skip-Gram – recap

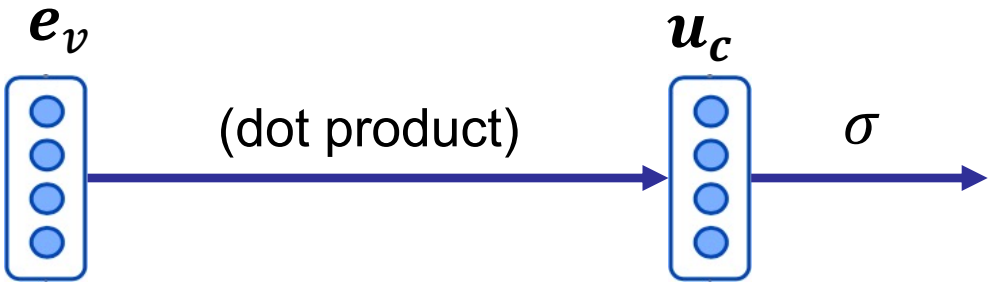
- Training data with a window size of 2 in the form of $(\text{word}, \text{context–word})$, namely (v, c) :

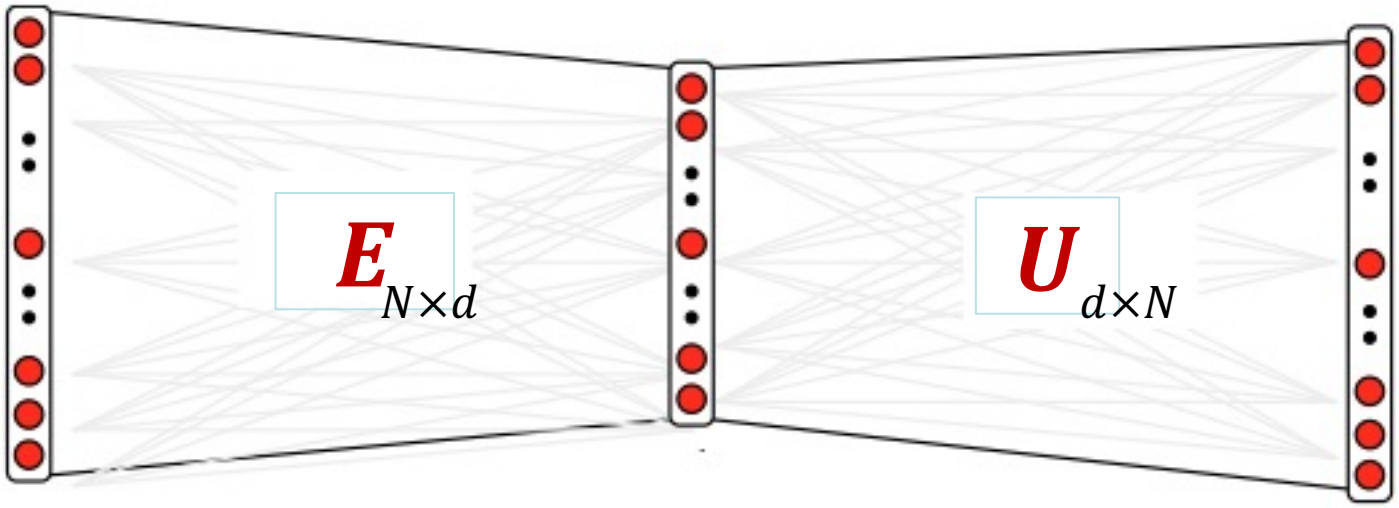
Some training data points in $\mathcal{D}$:

(Tesgüino , people)
(Tesgüino , drink)
(Tesgüino , while)
(Tesgüino , the)

word2vec Skip-Gram – recap

Training data: (v = *Tesgüino*, c = *drink*)



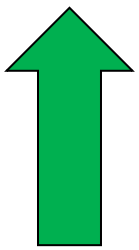


word2vec Skip-Gram – recap

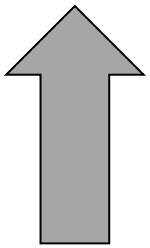
- Negative Sampling loss
  - increases $P(y = 1|v, c)$ probability for positive sample (v, c)
  - decreases $P(y = 1|v, \tilde{c})$ probability for k negative samples $(v, \tilde{c})$

- Loss function:

$$\mathcal{L} = -\mathbb{E}_{(v,c)\sim\mathcal{D}} \left[\log \sigma(\boldsymbol{e}_v \boldsymbol{u}_c) - \sum_{\substack{\tilde{c}\sim\tilde{\mathcal{D}} \\ k \text{ times}}} \log \sigma(\boldsymbol{e}_v \boldsymbol{u}_{\tilde{c}}) \right]$$



positive sample



k negative samples

sent2vec

- A simple and efficient method for creating sentence representations

- sent2vec (similar to word2vec) trains word embeddings, and then calculates a sentence embedding as the average of word embeddings:

$$\boldsymbol{e}_S = \frac{1}{|S|}\sum_{v \in S} \boldsymbol{e}_v$$

- The objective of sent2vec is to train effective sentence embeddings. It trains word embeddings ($\boldsymbol{E}$) in the way they can fulfill this objective

Pagliardini, Matteo, Prakhar Gupta, and Martin Jaggi. "Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features." *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2018.

Sent2vec – Training

- Parameters of sent2vec – similar to word2vec – consists of word embeddings ($\boldsymbol{E}$) and context-word embeddings ($\boldsymbol{U}$)

- Given a sentence S, a training data point is defined as the pair of:

(set of words in S while putting out one word v , the left-out word v)

$$(S \backslash \{v\}\,,\, v)$$

- During training, $\boldsymbol{e}_{S\backslash\{v\}}$, the sentence embedding without the left-out word, aims to predict the left-out word v

- The optimization is done with Negative Sampling

sent2vec – Example

- Training data is in the form of $(S\backslash\{v\}, v)$

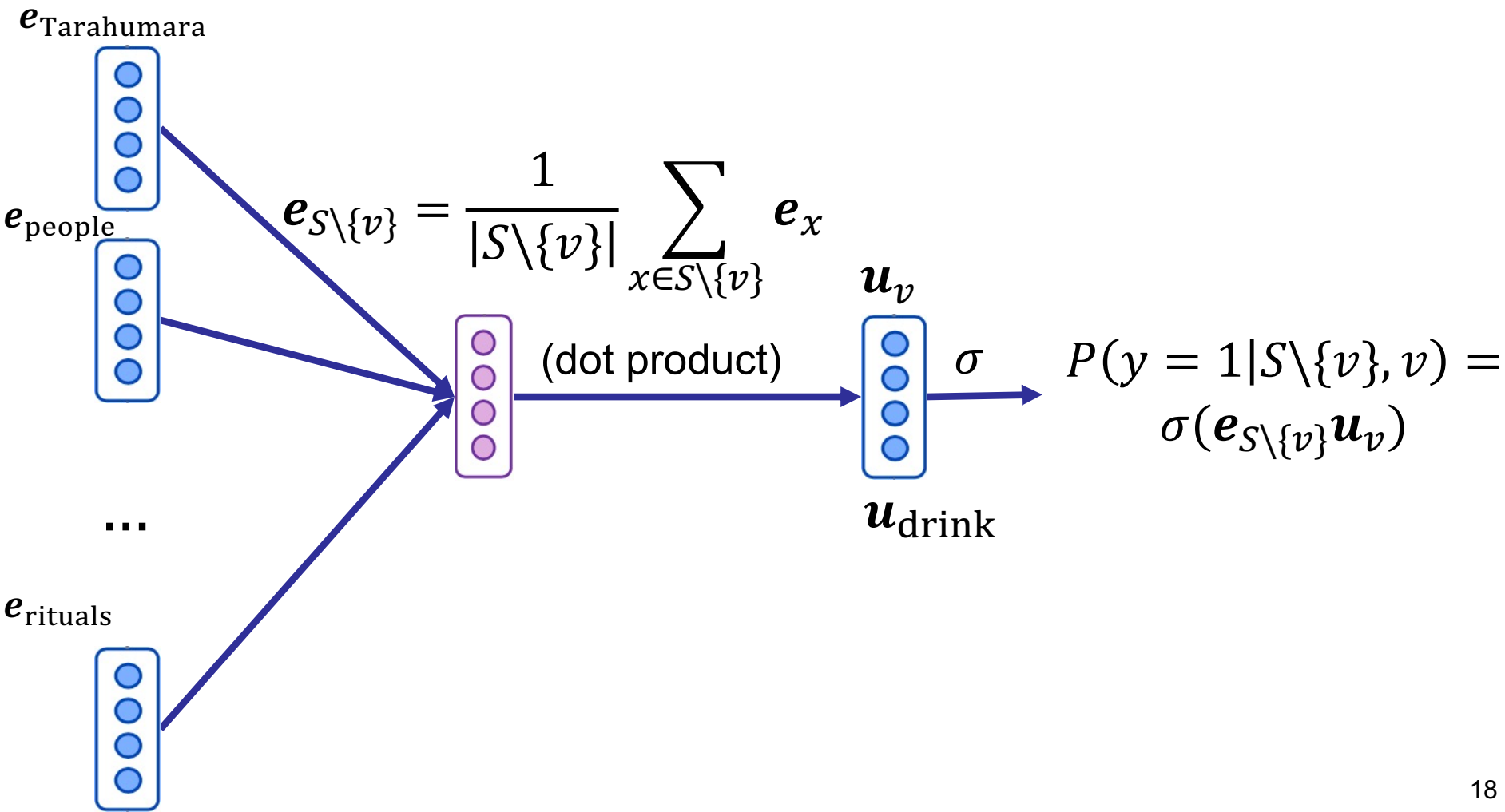
$S =$ `Tarahumara people drink Tesgüino during the rituals`

Some training data points in $\mathcal{D}$:

(people drink Tesgüino during the rituals , Tarahumara)
(Tarahumara drink Tesgüino during the rituals , people)
(Tarahumara people Tesgüino during the rituals , drink)
(Tarahumara people drink during the rituals , Tesgüino)
(Tarahumara people drink Tesgüino the rituals , during)
…

sent2vec – Architecture

Training data:
($S\backslash\{v\}$ = Tarahumara people Tesgüino during the rituals , v = *drink*)

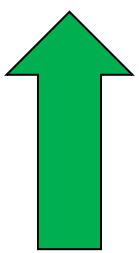


sent2vec – Negative Sampling loss function

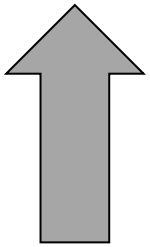
- Negative Sampling loss
  - increases $P(y = 1|S\backslash\{v\}, v)$ probability for positive sample $(S\backslash\{v\}, v)$
  - decreases $P(y = 1|S\backslash\{v\}, \tilde{v})$ probability for k negative samples $(S\backslash\{v\}, \tilde{v})$

- Loss function:

$$\mathcal{L} = -\mathbb{E}_{(S\backslash\{v\},v)\sim\mathcal{D}}\left[\log\sigma\left(\boldsymbol{e}_{S\backslash\{v\}}\boldsymbol{u}_v\right) - \sum_{\substack{\tilde{v}\sim\tilde{\mathcal{D}} \\ k \text{ times}}} \log\sigma\left(\boldsymbol{e}_{S\backslash\{v\}}\boldsymbol{u}_{\tilde{v}}\right)\right]$$



positive sample



k negative samples

Some results on supervised tasks

Data	Model	MSRP (Acc / F1)	MR	CR	SUBJ	MPQA	TREC	Average
Unordered Sentences: (Toronto Books; 70 million sentences, 0.9 Billion Words)	SAE	**74.3** / 81.7	62.6	68.0	86.1	76.8	80.2	74.7
	SAE + embs.	70.6 / 77.9	73.2	75.3	89.8	86.2	80.4	79.3
	SDAE	**76.4 / 83.4**	67.6	74.0	89.3	81.3	77.7	78.3
	SDAE + embs.	**73.7** / 80.7	74.6	78.0	90.8	**86.9**	78.4	80.4
	ParagraphVec DBOW	72.9 / 81.1	60.2	66.9	76.3	70.7	59.4	67.7
	ParagraphVec DM	73.6 / **81.9**	61.5	68.6	76.4	78.1	55.8	69.0
	Skipgram	69.3 / 77.2	73.6	77.3	89.2	85.0	82.2	78.5
	C-BOW	67.6 / 76.1	73.6	77.3	89.1	85.0	82.2	79.1
	Unigram TFIDF	73.6 / 81.7	73.7	79.2	90.3	82.4	**85.0**	80.7
	Sent2Vec uni.	72.2 / 80.3	75.1	**80.2**	90.6	**86.3**	83.8	**81.4**
	Sent2Vec uni. + bi.	72.5 / 80.8	**75.8**	**80.3**	**91.2**	85.9	**86.4**	**82.0**
Ordered Sentences: Toronto Books	SkipThought	73.0 / **82.0**	**76.5**	**80.1**	**93.6**	**87.1**	**92.2**	**83.8**
	FastSent	72.2 / 80.3	70.8	78.4	88.7	80.6	76.8	77.9
	FastSent+AE	71.2 / 79.1	71.8	76.7	88.8	81.5	80.4	78.4
2.8 Billion words	C-PHRASE	72.2 / 79.6	**75.7**	78.8	**91.1**	86.2	78.8	80.5

Details: https://www.aclweb.org/anthology/N18-1049.pdf

Some results on unsupervised tasks

Model	STS 2014 News	Forum	WordNet	Twitter	Images	Headlines	SICK 2014 Test + Train	Average
SAE	.17/.16	.12/.12	.30/.23	.28/.22	.49/.46	.13/.11	.32/.31	.26/.23
SAE + embs.	.52/.54	.22/.23	.60/.55	.60/.60	.64/.64	.41/.41	.47/.49	.50/.49
SDAE	.07/.04	.11/.13	.33/.24	.44/.42	.44/.38	.36/.36	.46/.46	.31/.29
SDAE + embs.	.51/.54	.29/.29	.56/.50	.57/.58	.59/.59	.43/.44	.46/.46	.49/.49
ParagraphVec DBOW	.31/.34	.32/.32	.53/.50	.43/.46	.46/.44	.39/.41	.42/.46	.41/.42
ParagraphVec DM	.42/.46	.33/.34	.51/.48	.54/.57	.32/.30	.46/.47	.44/.40	.43/.43
Skipgram	.56/.59	.42/.42	.73/**.70**	**.71**/.74	.65/.67	.55/.58	.60/.69	.60/.63
C-BOW	.57/.61	**.43/.44**	.72/.69	**.71/.75**	.71/.73	.55/.59	.60/.69	.60/.65
Unigram TF-IDF	.48/.48	.40/.38	.60/.59	.63/.65	.72/.74	.49/.49	.52/.58	.55/.56
Sent2Vec uni.	**.62/.67**	**.49/.49**	**.75/.72**	.70/**.75**	**.78/.82**	**.61/.63**	**.61/.70**	**.65/.68**
Sent2Vec uni. + bi.	**.62/.67**	**.51/.51**	.71/.68	.70/**.75**	**.75/.79**	.59/.62	**.62/.70**	**.65/.67**
SkipThought	.44/.45	.14/.15	.39/.34	.42/.43	.55/.60	.43/.44	.57/.60	.42/.43
FastSent	.58/.59	.41/.36	**.74/.70**	.63/.66	.74/.78	.57/.59	**.61/.72**	.61/.63
FastSent+AE	.56/.59	.41/.40	.69/.64	.70/.74	.63/.65	.58/.60	.60/.65	.60/.61
Siamese C-BOW[4]	.58/.59	.42/.41	.66/.61	**.71**/.73	.65/.65	**.63/.64**	–	–
C-PHRASE	**.69/.71**	**.43**/.41	.76/.73	.60/.65	**.75/.79**	**.60/.65**	.60/**.72**	**.63/.67**

Details: https://www.aclweb.org/anthology/N18-1049.pdf

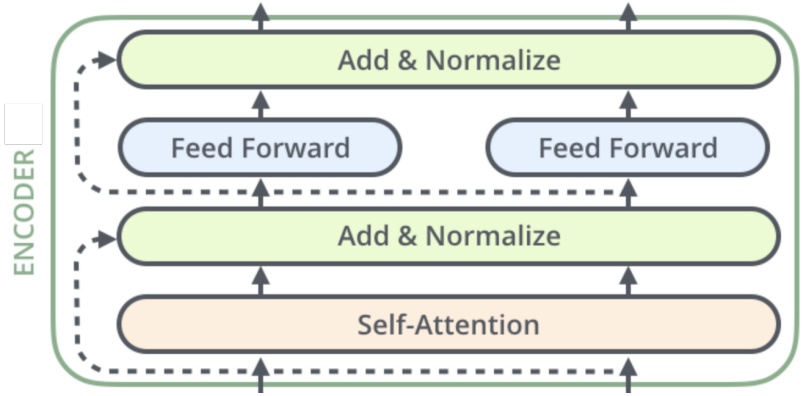
Agenda

- Compositional representations
- Sentence embedding with sent2vec
- **An overview on BERT**

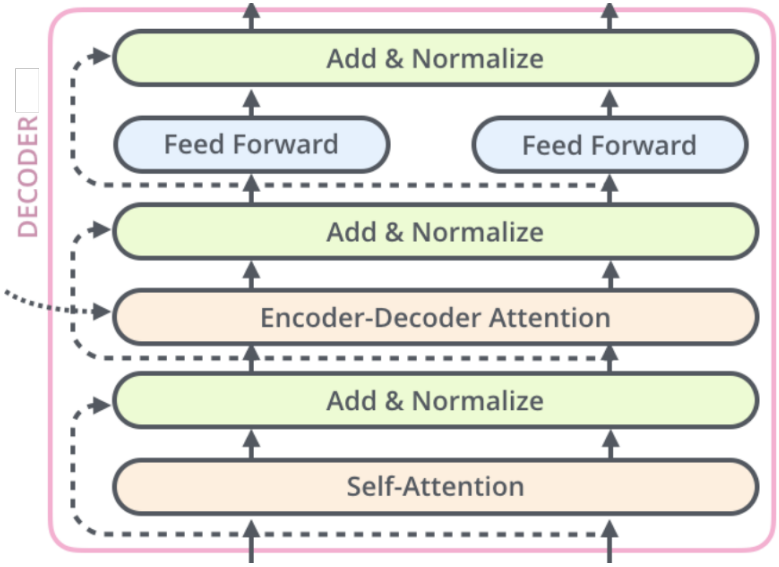
Transformers

- Transformers are originally introduced for machine translation, and since then are widely used in almost any task for sequence encoding and sequence decoding

Transformer encoder



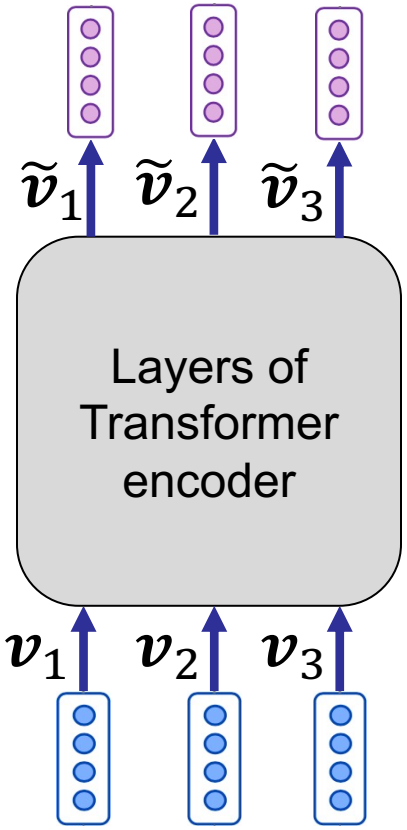
Transformer decoder

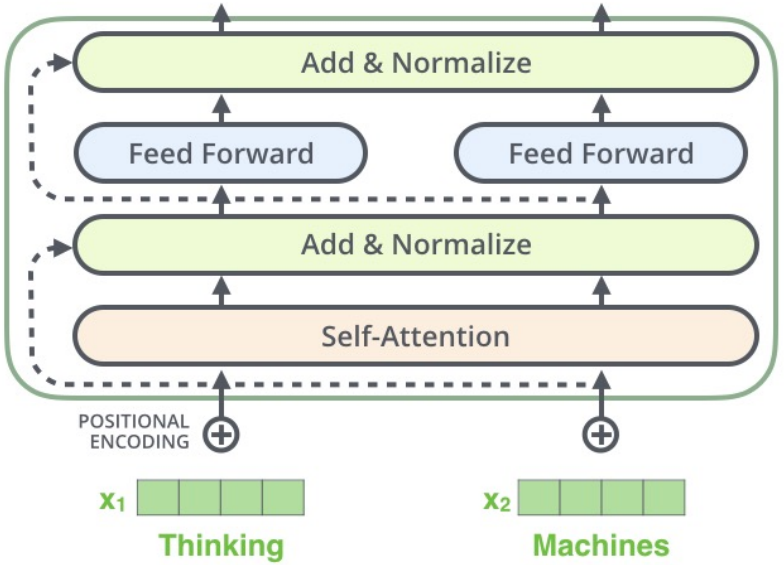


Attention is all you need. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Polosukhin, I. (2017). In *NeurIPS*.
Figure source: http://jalammar.github.io/illustrated-transformer/

Contextualized word embeddings with Transformer

- Each encoded vector is the contextual embedding of the corresponding input vector
  - $\tilde{\boldsymbol{v}}_i$ is the contextual embedding of $\boldsymbol{v}_i$





Contextualized word embeddings with LM objective

All of these models are Transformer models

ELMo
Oct 2017
Training:
800M words
42 GPU days

GPT
June 2018
Training
800M words
240 GPU days

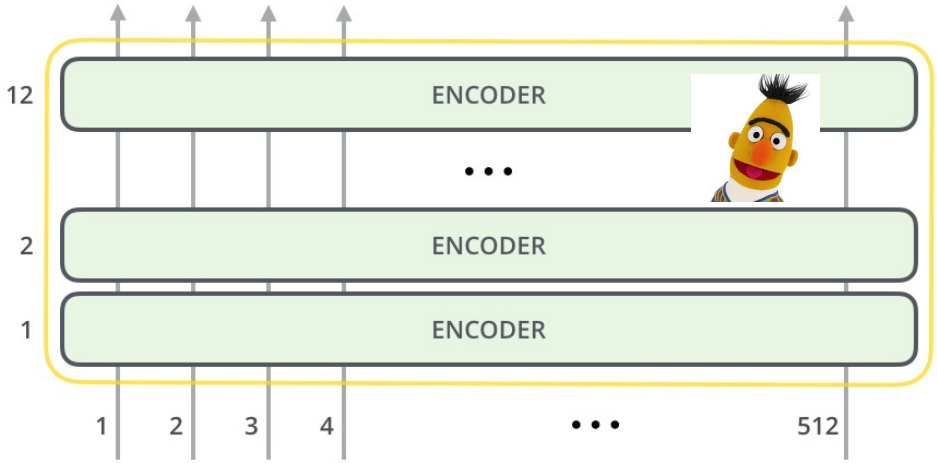
BERT
Oct 2018
Training
3.3B words
256 TPU days
~320–560
GPU days

GPT-2
Feb 2019
Training
40B words
~2048 TPU v3
days according to
a reddit thread

XL-Net,
ERNIE,
Grover
RoBERTa, T5
July 2019—

Source: http://web.stanford.edu/class/cs224n/

BERT
Bidirectional Encoder Representation from Transformers



- BERT is a pre-trained language model, composed of multi-layers of Transformer encoder
- BERT …
  - provides contextualized word embeddings
  - uses WordPiece for tokenization
  - uses sentence pair encoding which provides …
    - sequence (sentence/document/etc.) embedding
    - an embedding for relation estimation between two sequences

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL-HLT (2019)*

WordPiece – recap

- WordPiece is a descendent of BPE
- WordPiece with MaxMatch decoding is used in BERT
- WordPiece indicates internal subwords with “##” special symbol

Examples:

*“**unavoidable**”*

[*“**un**”, “**##avoid**”, “**##able**”*]

*“**natural language processing**”*

First pre-tokenization:

[*“**natural**”, “**language**”, “**processing**”*]

Then subword tokenization:

[*“**natural**”, “**lang**”, “**##uage**”, “**process**”, “**##ing**”*]

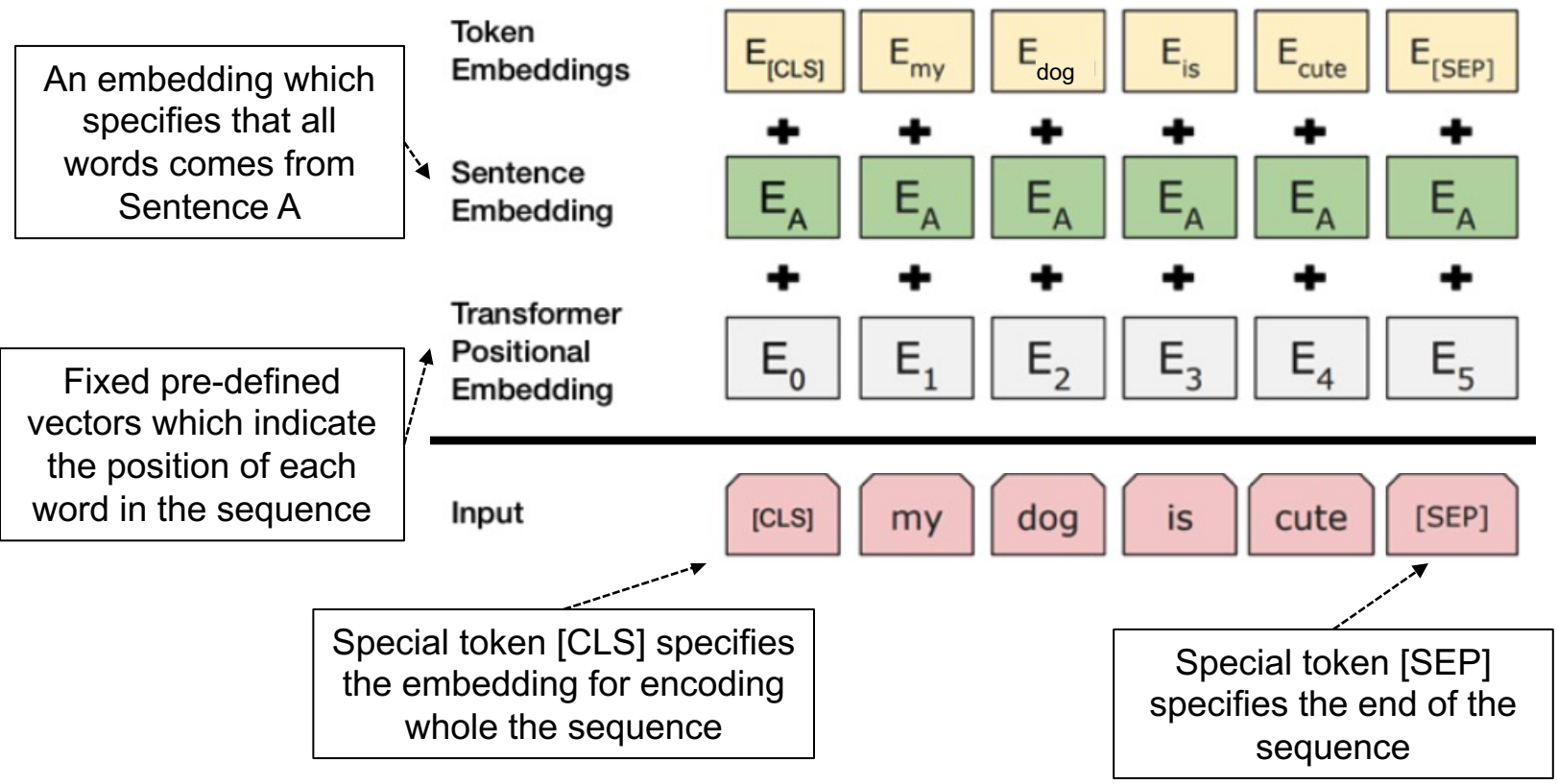
Training

- Training objective is Masked Language Model (MLM)
  - MLM masks a word inside a sequence and then tries to predict it (instead of predicting the next word in common LM objectives)
  - Trained on Wikipedia + BookCorpus
  - Fairly expensive for training from scratch – takes four days on 4 to 16 TPUs
- Dictionary size is only ~30K tokens – due to WordPiece subword tokenization
- Specs of some provided pre-trained models:
  - BERT-Tiny: 2-layer, 128-hidden, 2-head, ~4M parameters*
  - BERT-Mini: 4-layer, 256-hidden, 4-head, ~11M parameters*
  - BERT-Base: 12-layer, 768-hidden, 12-head, ~110M parameters*
  - BERT-Large: 24-layer, 1024-hidden, 16-head, ~340M parameters*
- Some resources:
  - https://github.com/google-research/bert
  - Library to have BERT models in PyTorch: https://huggingface.co/transformers/

* For comparison, a (static) word embedding like word2vec with vocabulary size 200K and vector size 768 has 153M parameters

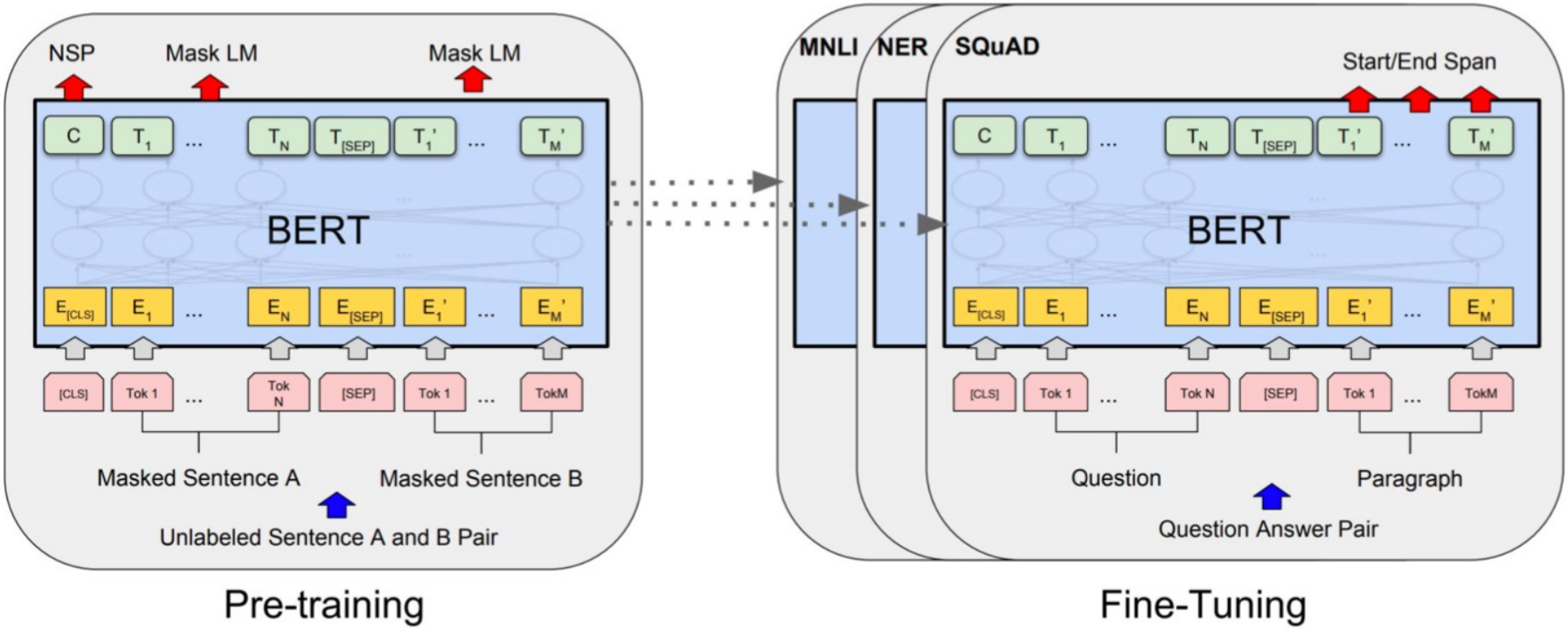
Input to BERT

- The input sequence to BERT
- The input embeddings to BERT are in fact the sum of three types of embeddings



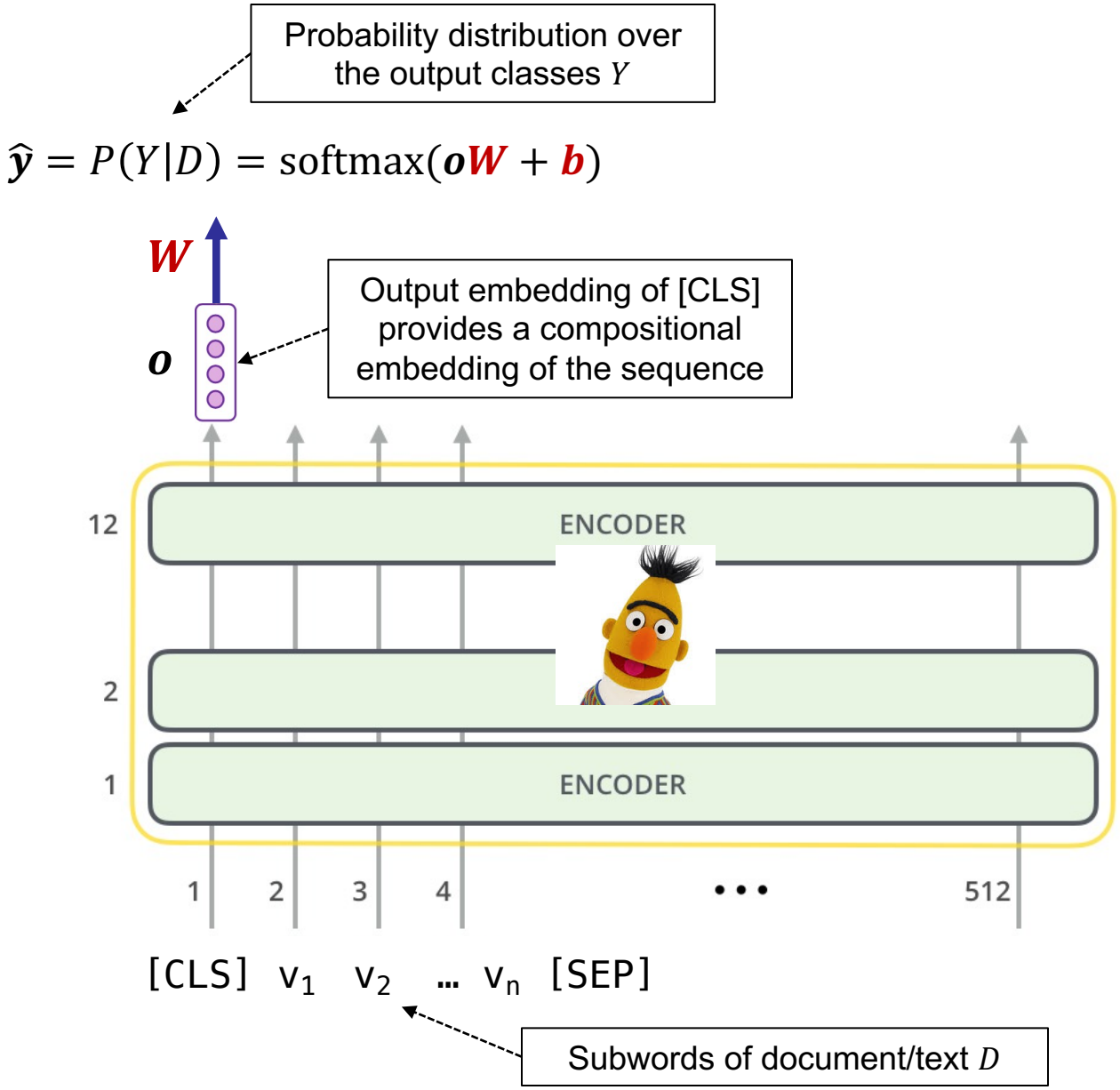
Fine-tuning

- To exploit BERT in down-stream tasks, the output embeddings are used for the corresponding predictions, and then all (or some) of parameters are updated end-to-end. This process is called fine-tuning.



Source: http://web.stanford.edu/class/cs224n/

BERT Fine-tuning for text classification

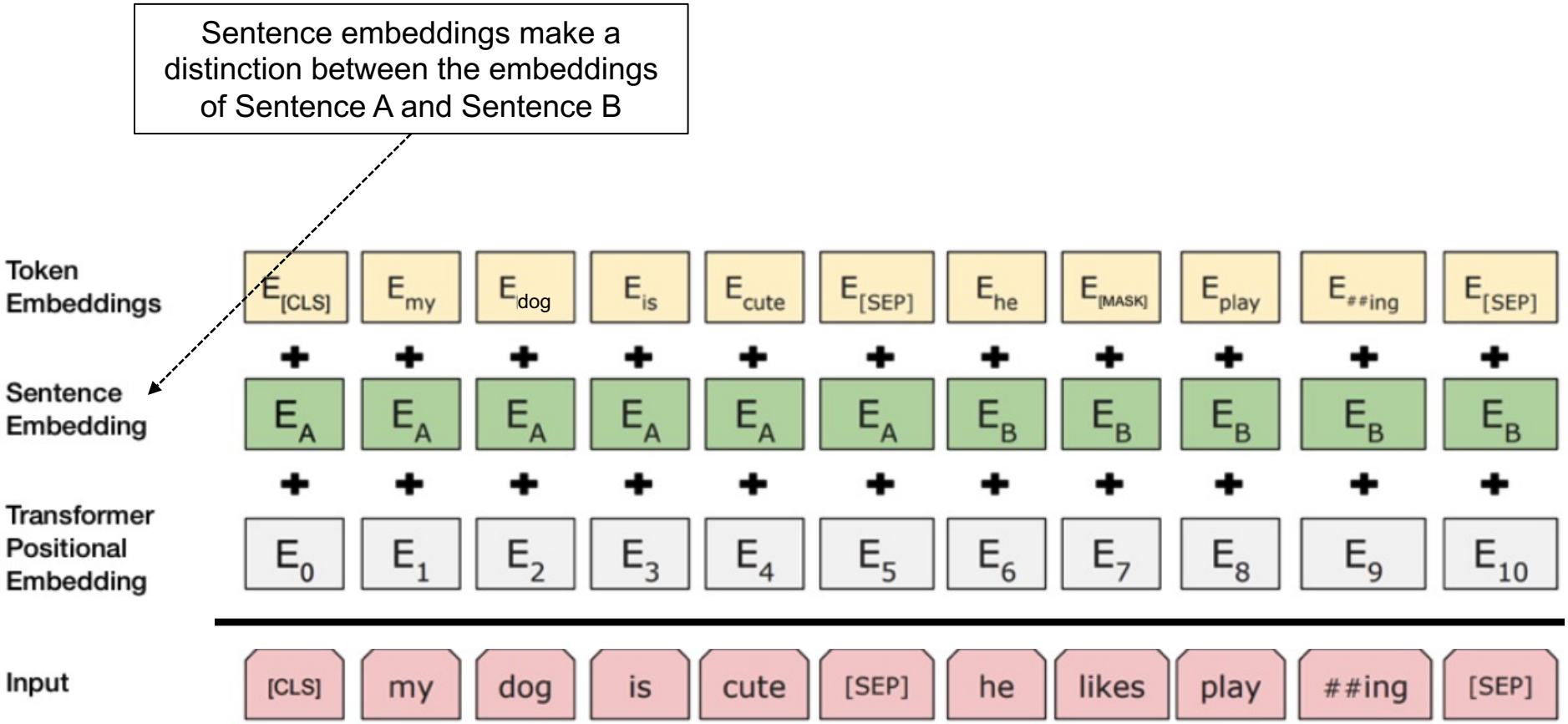


Sentence pair encoding

- Many NLP tasks require an estimation of the relation between two sequences
  - E.g., question answering, information retrieval, natural language inference, paraphrasing, etc.

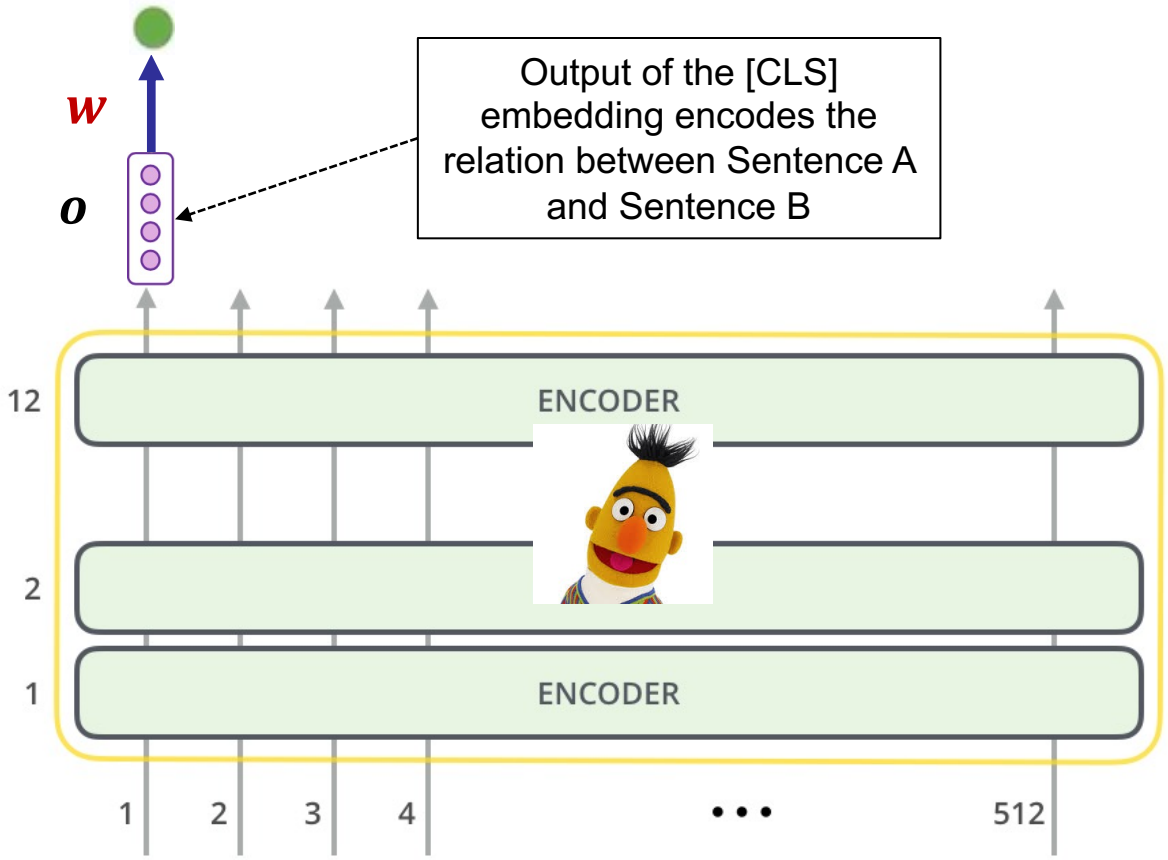
- During training, BERT learns the relationships between two sentences using an additional binary classifier objective
  - The binary classifier predicts whether Sentence B is the actual sentence that proceeds Sentence A, or a random sentence
  - The classifier is optimized based on the output embedding of [CLS]
  - The binary classifier is jointly optimized with the MLM objective

Input to BERT – two sequences



BERT Fine-tuning for Text Matching/Similarity tasks

$$\text{similarity}\left(D^{(1)}, D^{(2)}\right) = \boldsymbol{o}\boldsymbol{w}$$



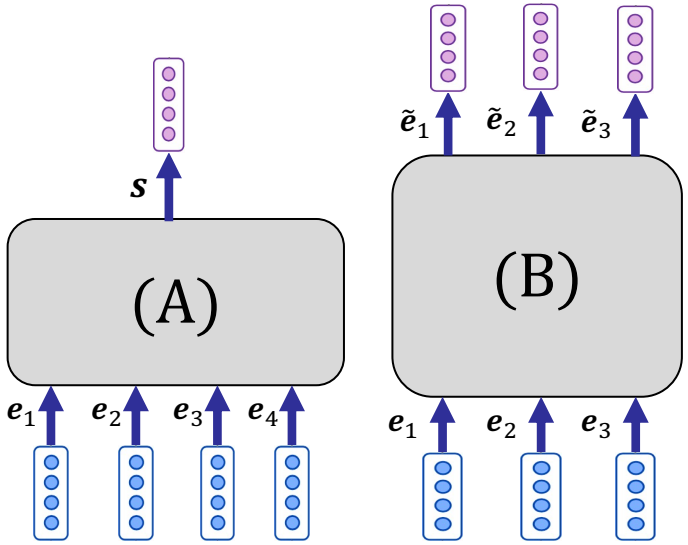
Some results

- A generic, deep, pre-trained model that can simply be plugged in (almost) any NLP task!

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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Recap

- Compositional representations
  A. Sequence encoding
  B. Contextualized word embeddings



- Efficient sentence embedding with sent2vec

- Word/sentence/sequence pair encoding with BERT