Machine Learning and NLP: Advances and Applications
Day 3: Advanced techniques and applications

1/24/2020

Yoshi Suhara

Recap: Course Overview

- Day 1: Machine Learning Basics
- Day 2: NLP Basics
- Day 3: Advanced Techniques and Applications

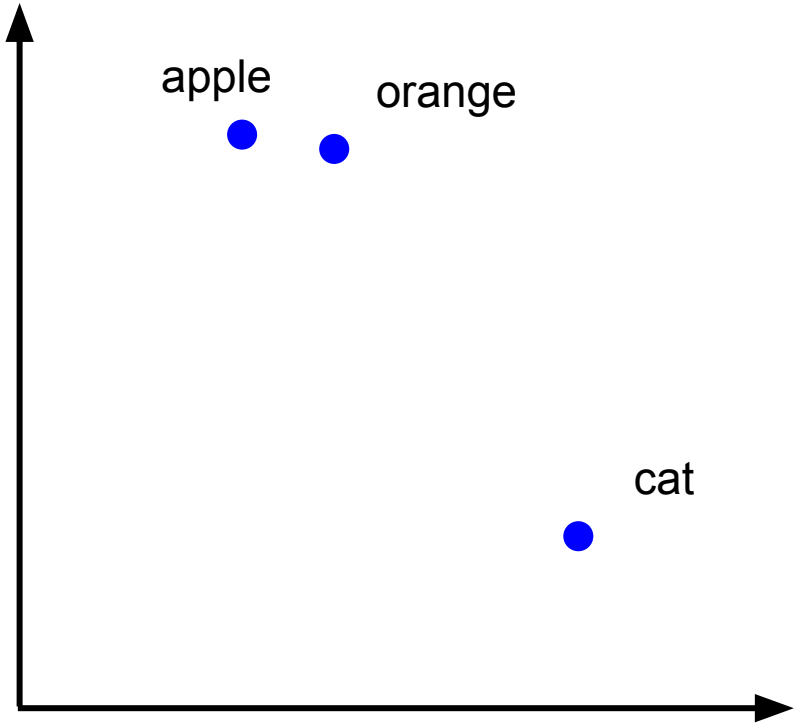
Recap: Course Overview

- Day 1: Machine Learning Basics
  - **Hands-on material 1**
- Day 2: NLP Basics
  - **Hands-on material 2**
- Day 3: Advanced Techniques and Applications
  - **Hands-on material 3**

Day 1 **Day 2** **Day 3**

Finally!

Everything is Vector Today



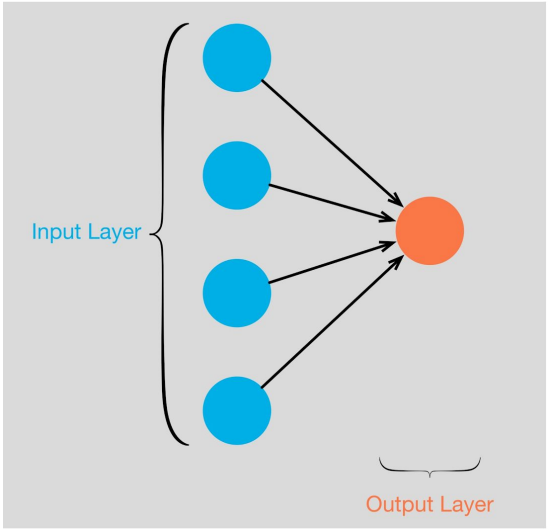
Day 3

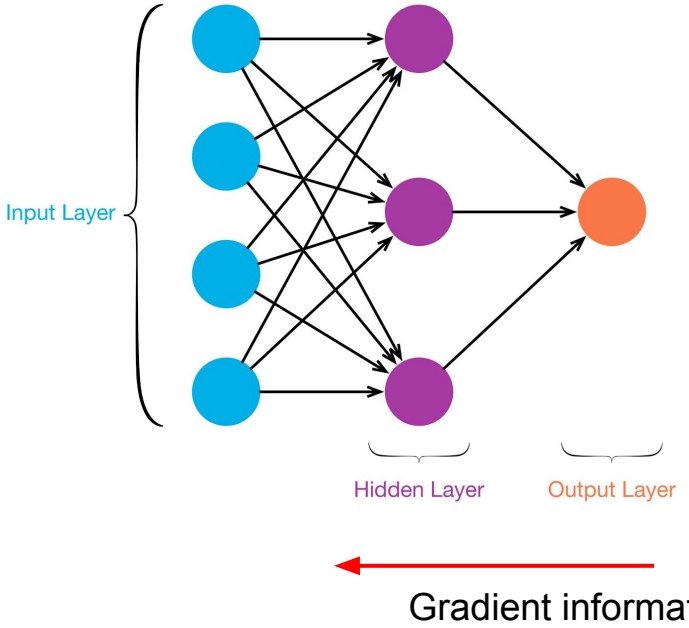
- Deep Learning Basics
- Word embeddings
- Neural Network Models for NLP
- Pre-training Methods for NLP
- Hands-on Part

Let's Learn Deep Learning

Recap: Neural Networks

- Node output = $\mathbf{a(w^T x + b)}$
- Training model parameters through backpropagation

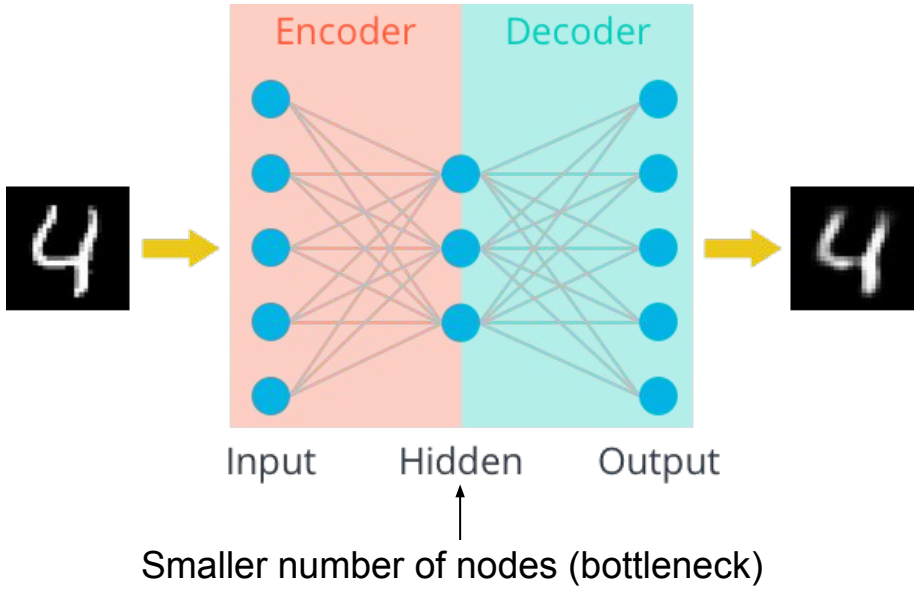




Rise of Representation Learning: Huge Success in Computer Vision

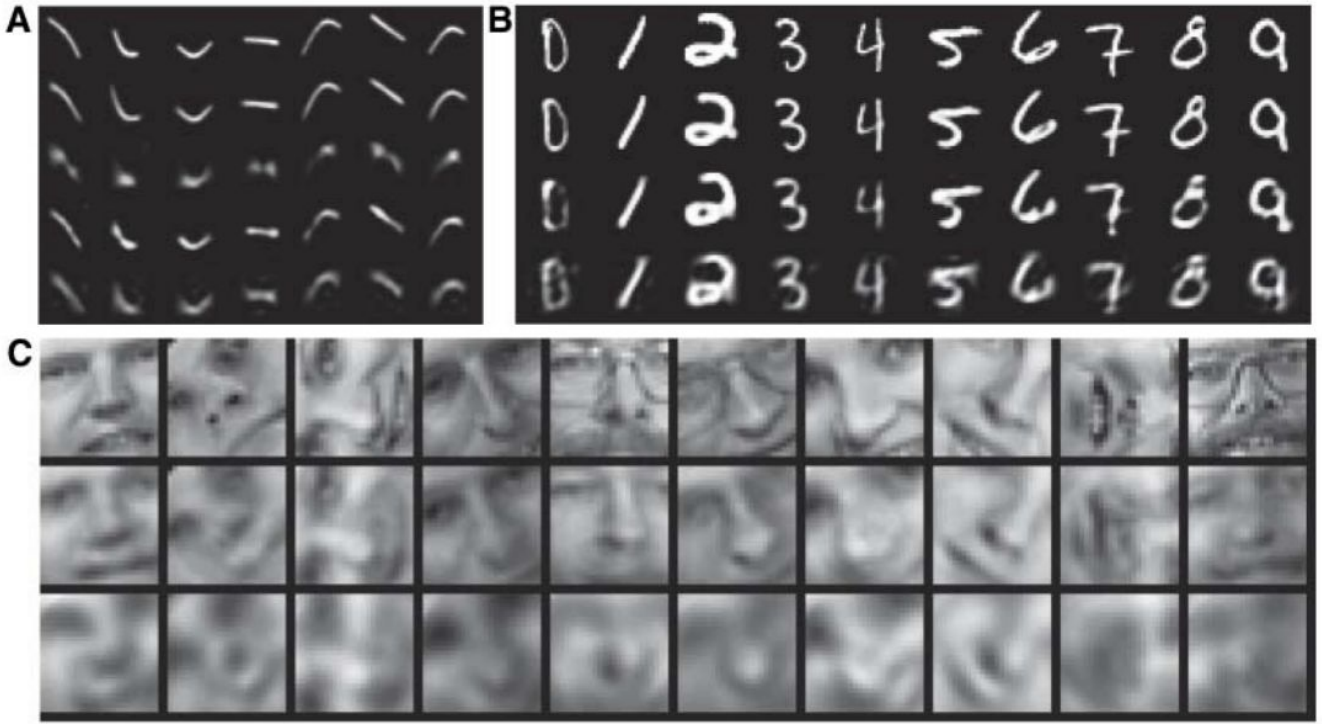
Key Technique 1: Autoencoder (2006)

- Aims to build a **dense representation** for each input by learning a dense representation that can reconstruct the original input



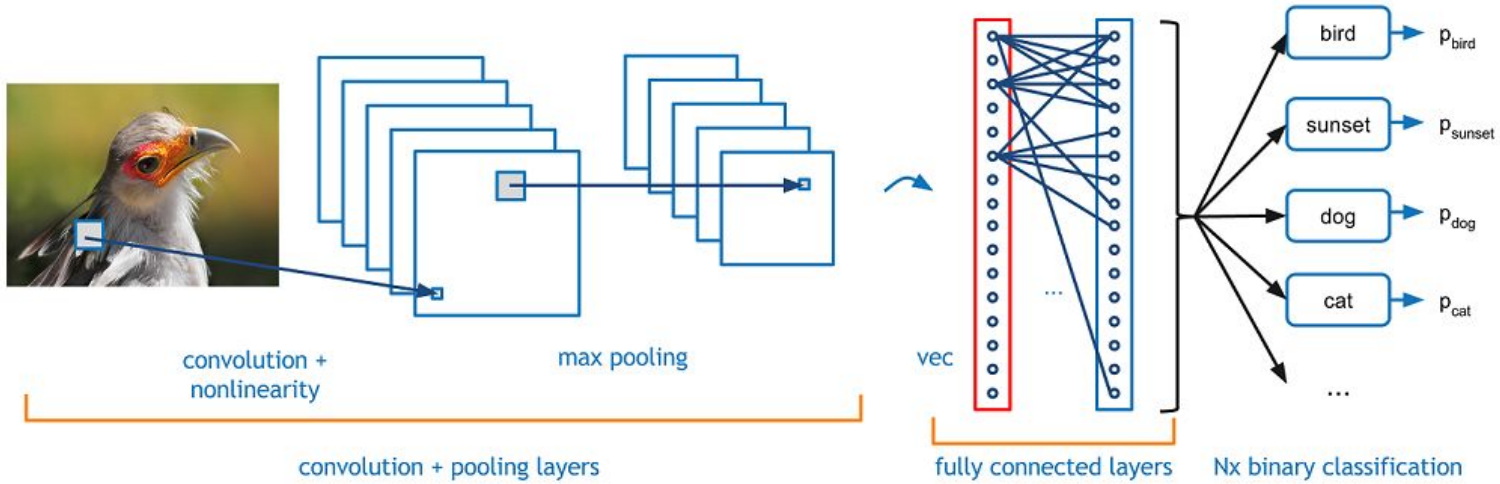
G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks", Science, 2006.

Learning Patterns in a Fully Unsupervised Manner



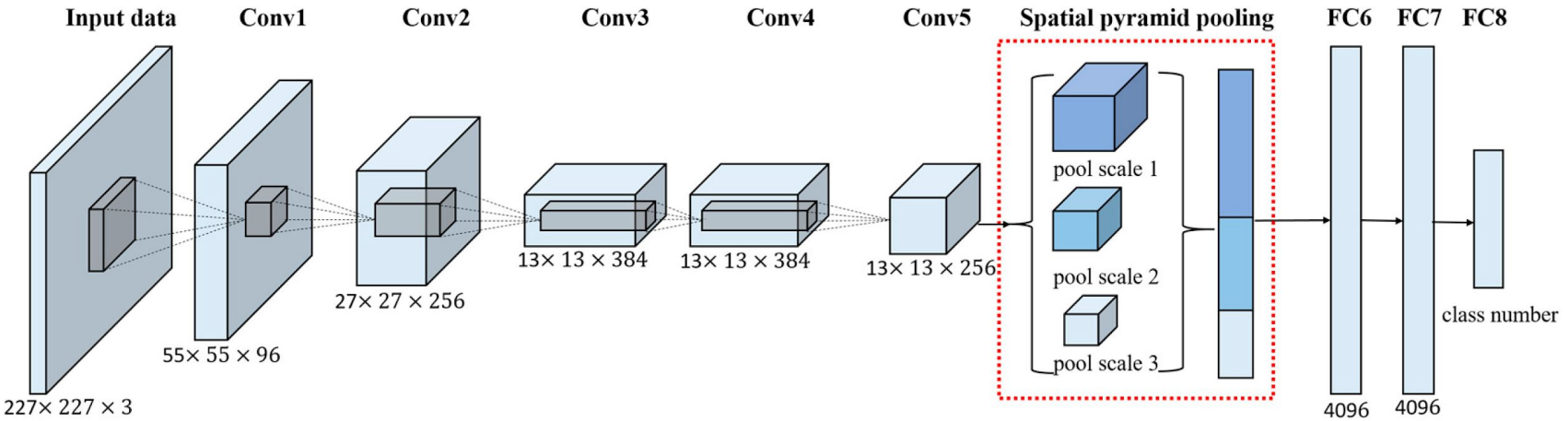
Key Technique 2: Convolutional NN (1980's~)

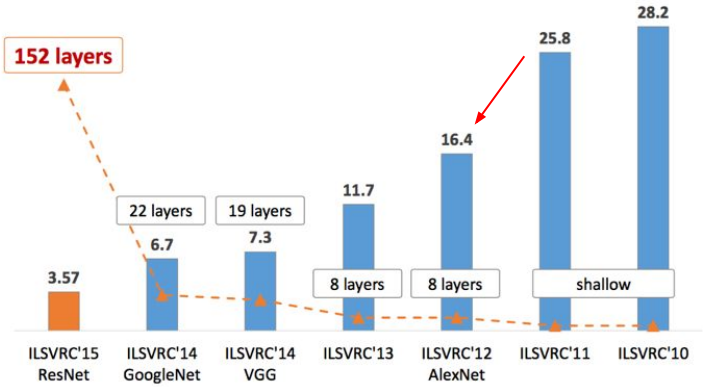
- Incorporating multiple convolution and pooling layers to capture "shift-invariant" sub-image information



Y. LeCun, L Bottou, Y Bengio, P Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE 86, 1998.
K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4): 93-202, 1980.

CNN meets GPU: AlexNet (2012)

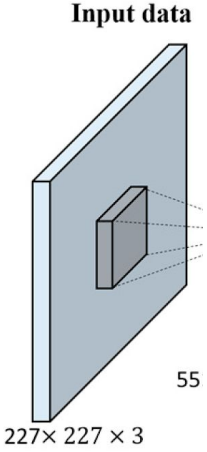


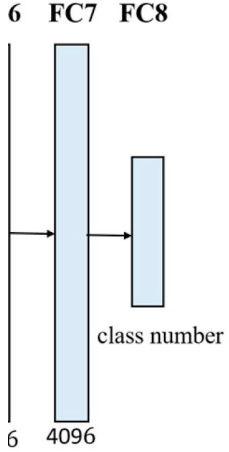


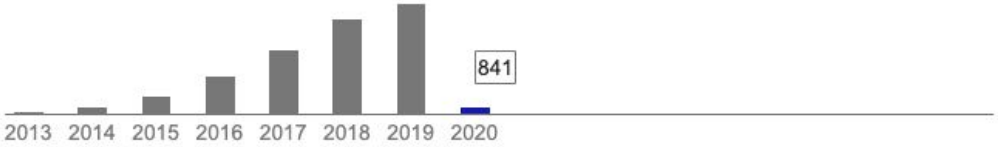
CNN meets GPU: AlexNet (2012)

Imagenet classification with deep convolutional neural networks

Authors	Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton
Publication date	2012
Conference	Advances in neural information processing systems
Pages	1097-1105
Description	We trained a large, deep convolutional neural network to classify the 1.3 million high-resolution images in the LSVRC-2010 ImageNet training set into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 39.7\% and 18.9\% which is considerably better than the previous state-of-the-art results. The neural network, which has 60 million parameters and 500,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and two globally connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of convolutional nets. To reduce overfitting in the globally connected layers we employed a new regularization method that proved to be very effective.
Total citations	Cited by 54737



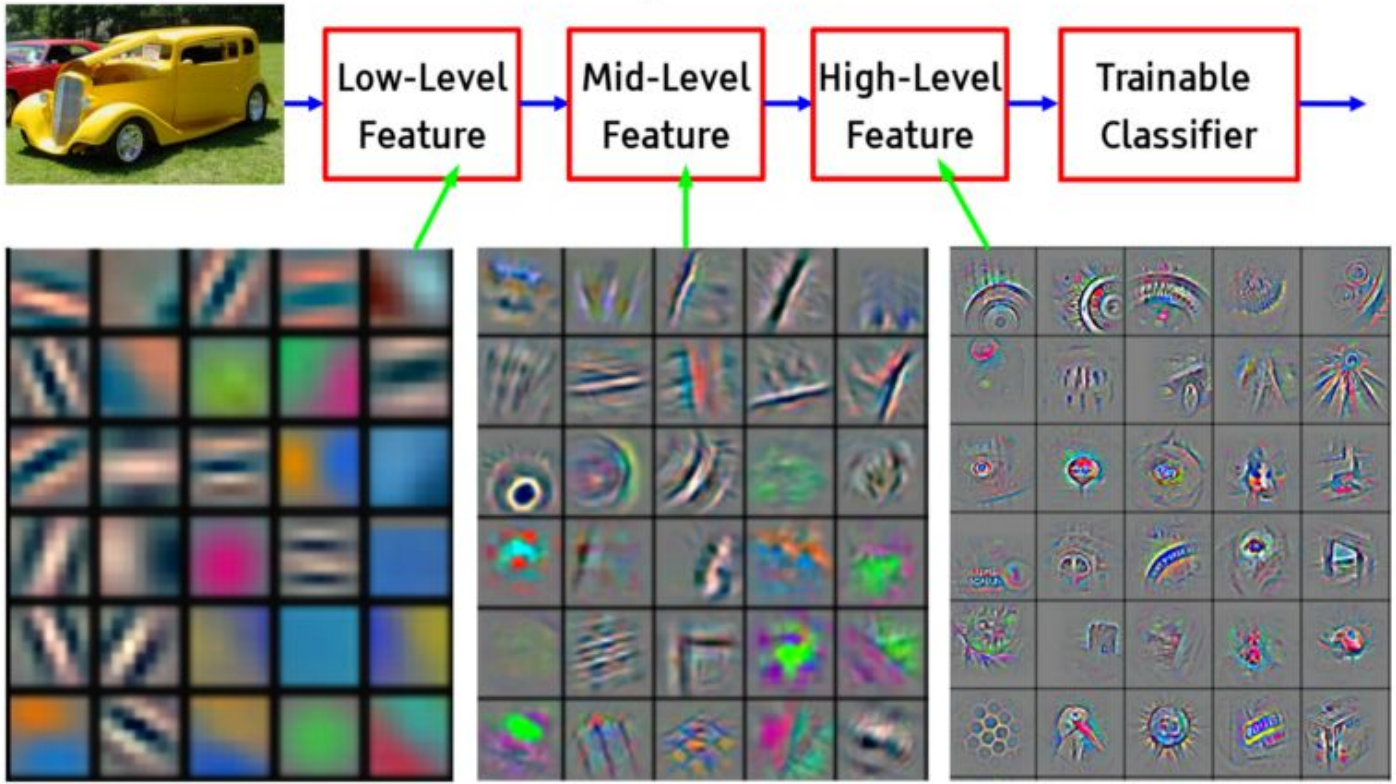




Deep Learning = Learning Hierarchical Representations

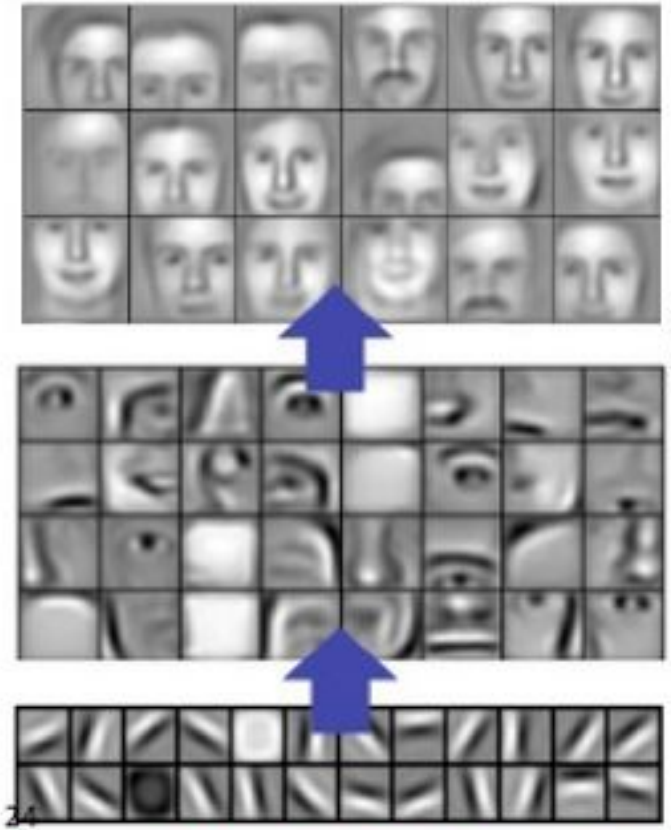
Y LeCun

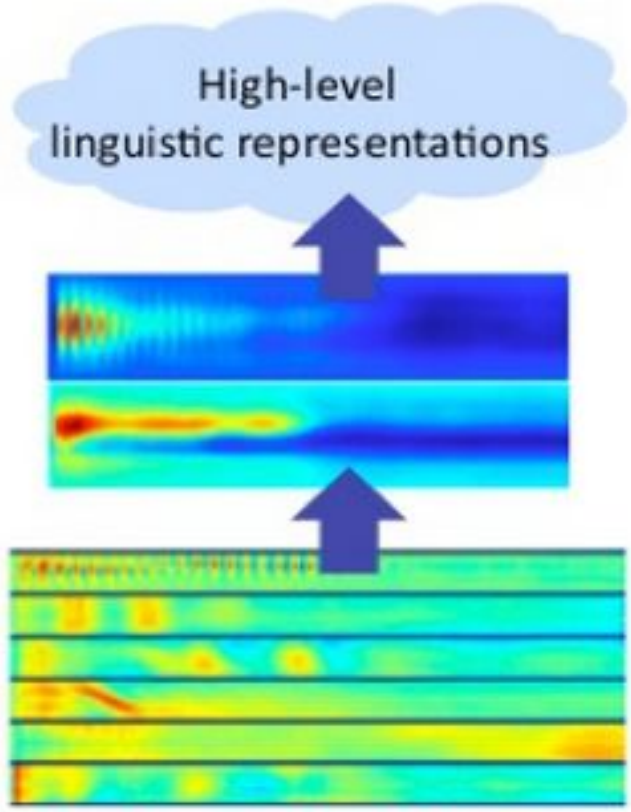
- It's deep if it has more than one stage of non-linear feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Successive model layers learn deeper intermediate representations





Prior: underlying factors & concepts compactly expressed w/ multiple levels of abstraction

Other Key Techniques include but not limited to

- ReLU activation function (2011)
- Dropout (2012)
- Batch Normalization (2015)
- ...

Major deep Learning frameworks implement those functions with high-level interface so we don't have to implement them from scratch

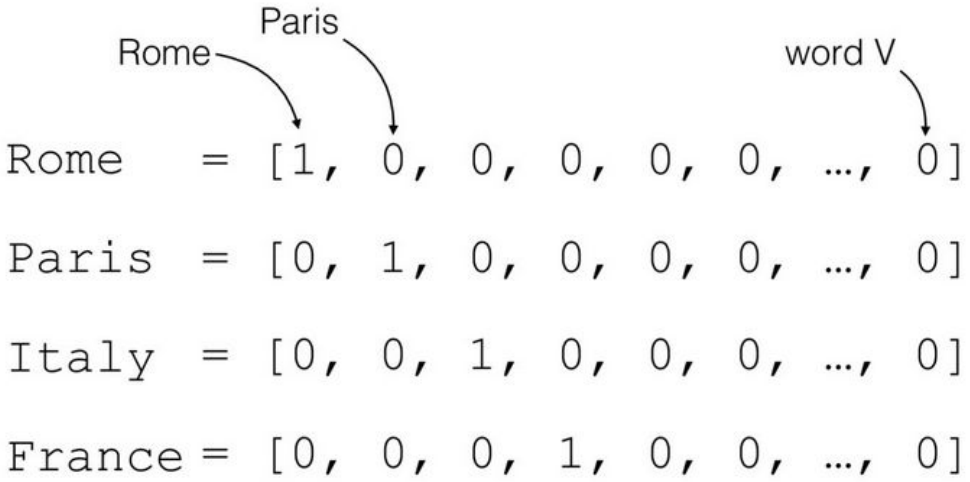
Deep Learning techniques for NLP

- Word embeddings
- Neural Network Models for NLP
  - CNN
  - RNN
  - Encoder-decoder
  - Self-attention (Transformer)

Word Embeddings

Motivation: Limitation of one-hot encoding (BoW)

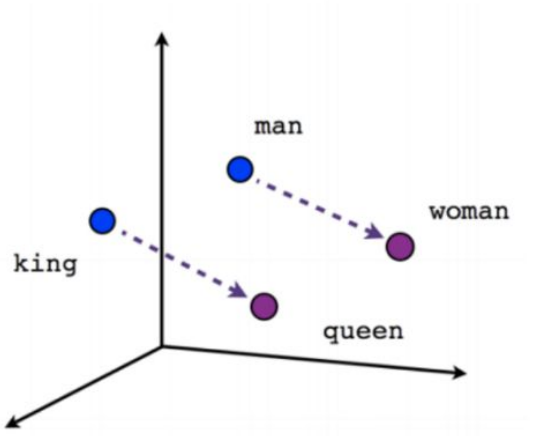
- Different words are assigned different IDs (= equally different)



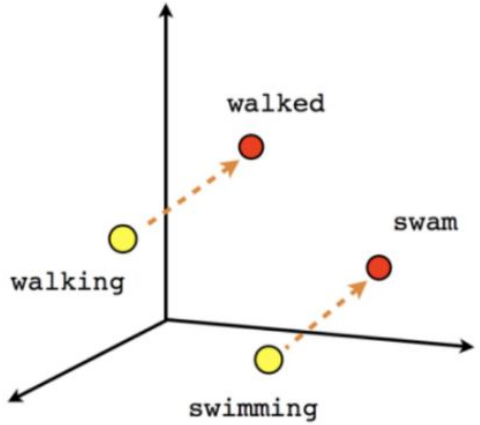
Can we build dense vectors that properly capture the semantic similarity?

word2vec (2013)

- A technique that creates a dense vector representation for each word
  - The vector representation has interesting property
  - e.g., v("king") - v("man") + v("woman") =~ v("queen")

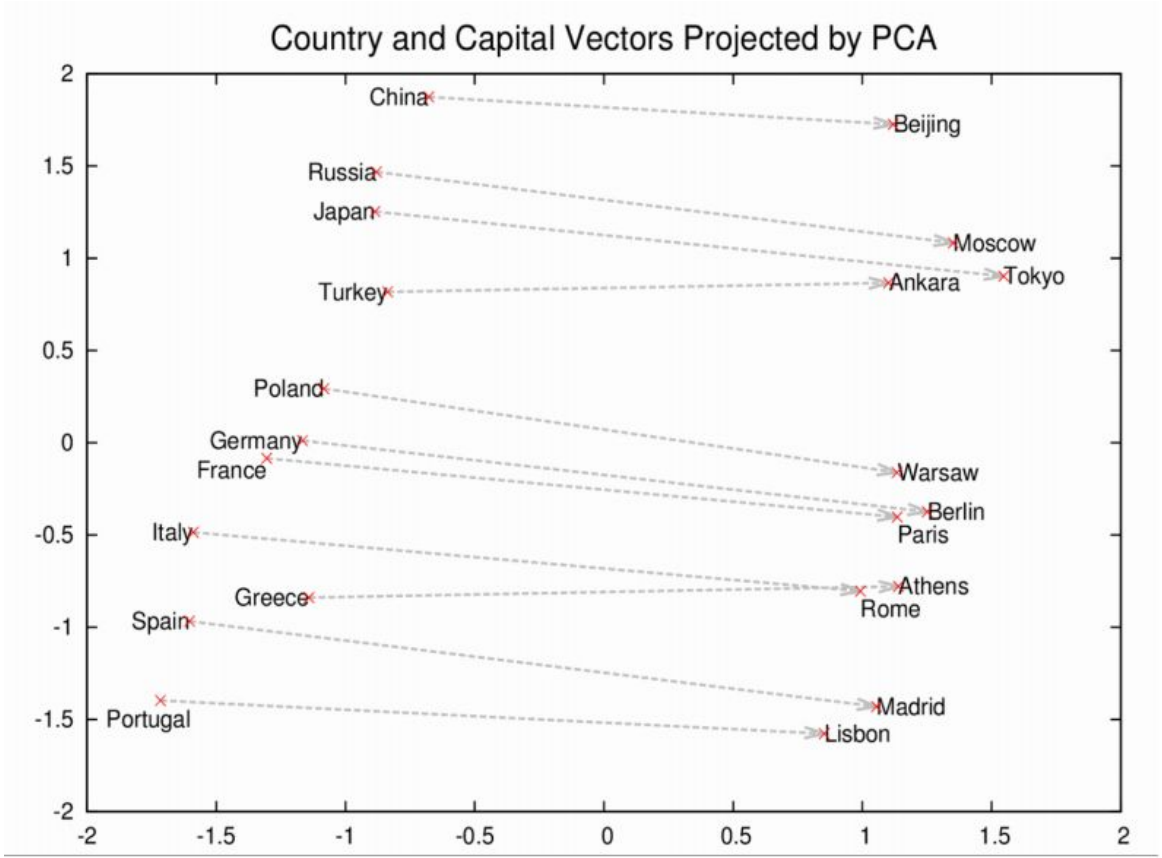


Male-Female



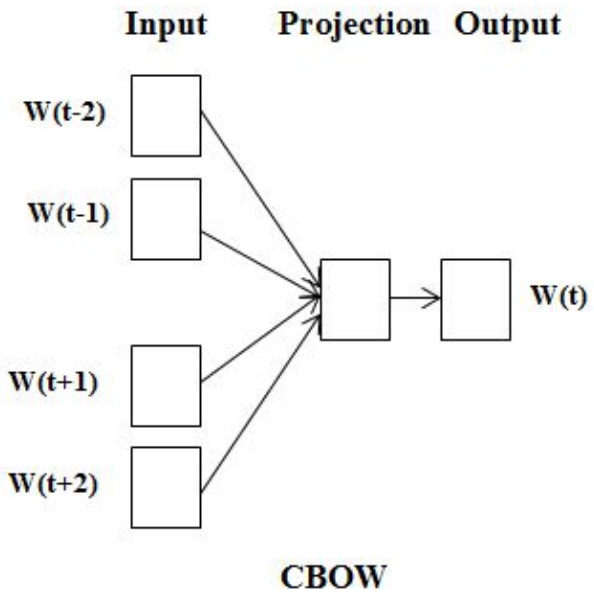
Verb tense

word2vec: country-capital relationships

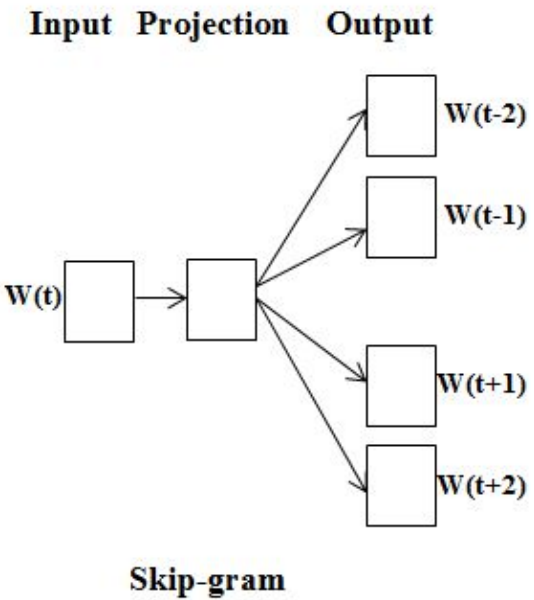


Training word2vec Model: CBOW or Skip-gram

- A) CBOW: Neighbor words → target word
- B) Skip-gram: Target word → neighbor words

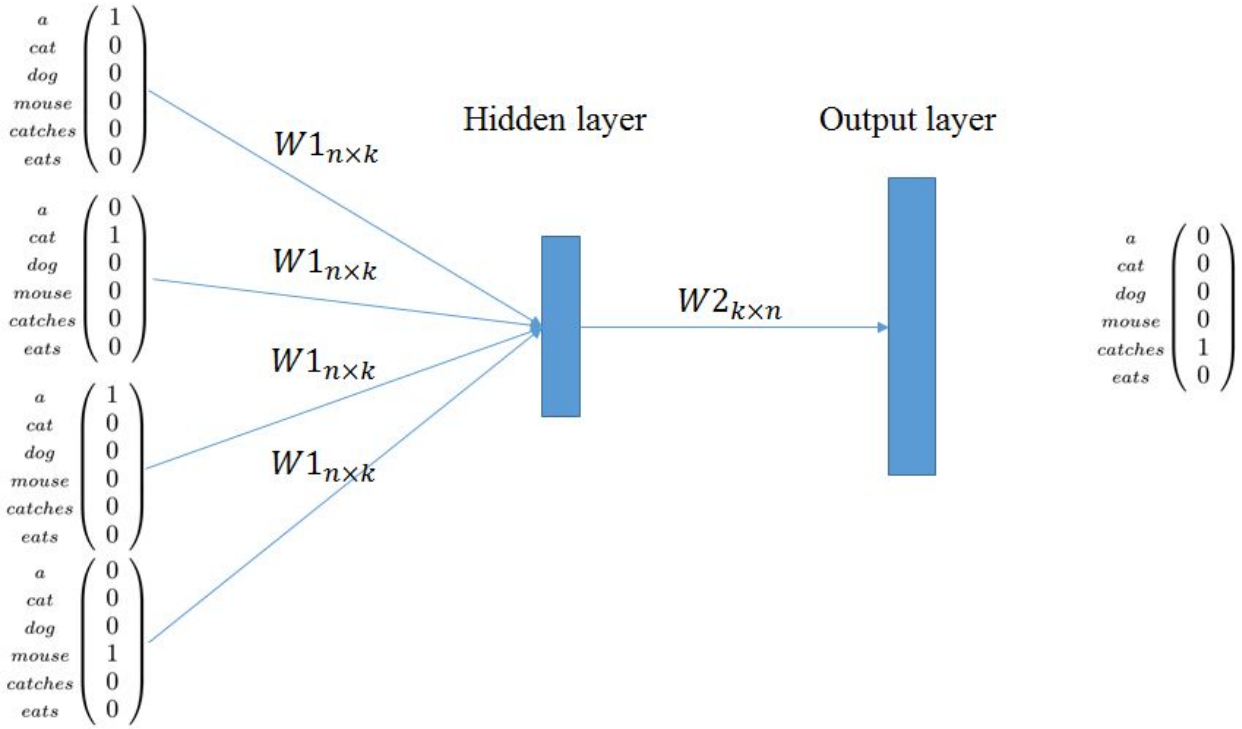


CBOW

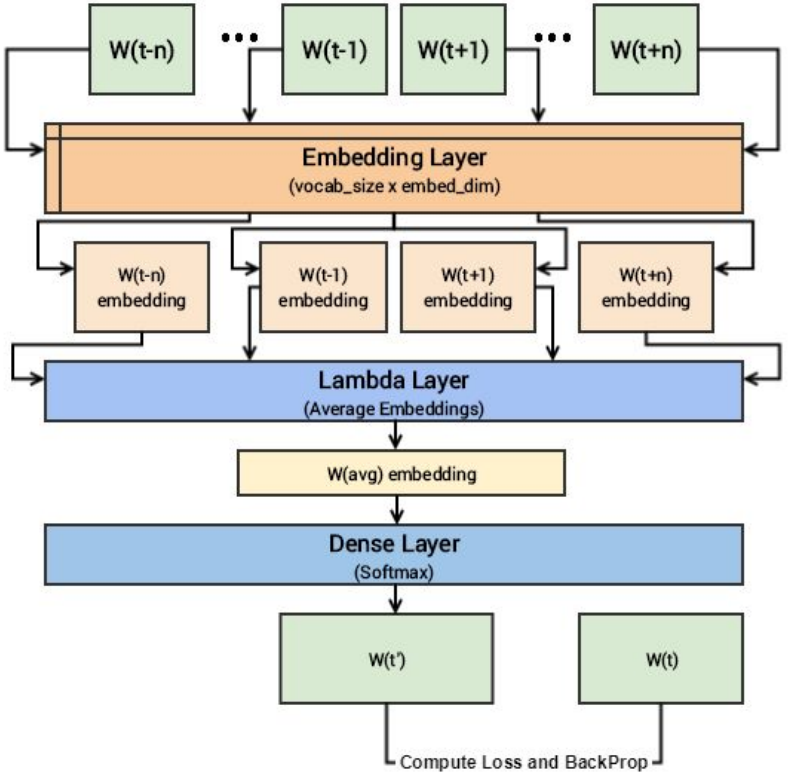


Skip-gram

word2vec: CBOW



word2vec: CBOW

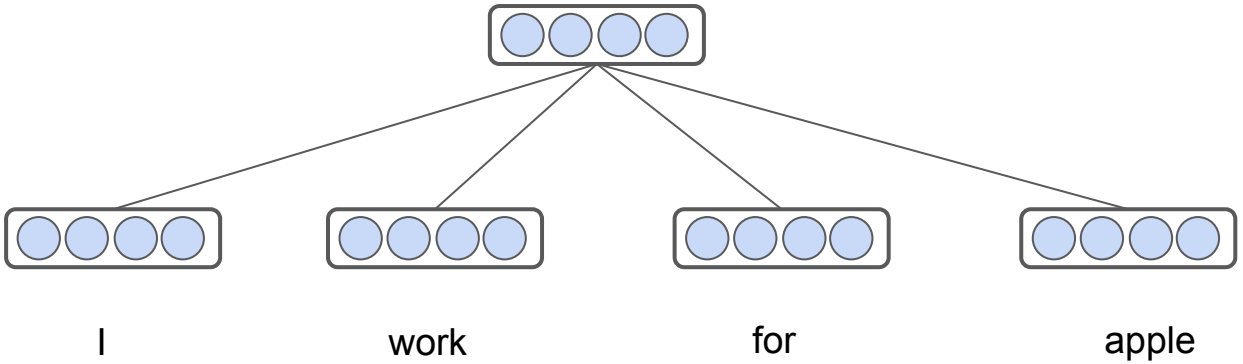


Sentence Embeddings

- 1) Simple Average
- 2) Smooth Inverse Frequency
- 3) Supervised methods (e.g., InferSent)

Simple Average

- Average word embeddings



Smooth Inverse Frequency (SIF)

- Simple Average+

Published as a conference paper at ICLR 2017

A SIMPLE BUT TOUGH-TO-BEAT BASELINE FOR SENTENCE EMBEDDINGS

Sanjeev Arora, Yingyu Liang, Tengyu Ma
Princeton University
{arora,yingyul,tengyu}@cs.princeton.edu

Algorithm 1 Sentence Embedding

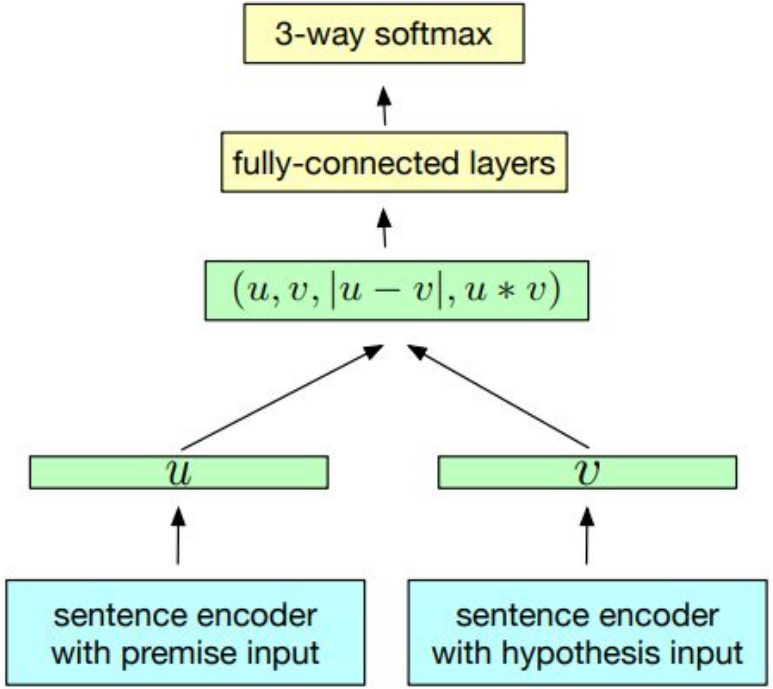
Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.

Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$

1: **for all** sentence s in $\mathcal{S}$ **do**
2: $\quad v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$ "IDF-like" weighting method
3: **end for**
4: Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector
5: **for all** sentence s in $\mathcal{S}$ **do**
6: $\quad v_s \leftarrow v_s - uu^\top v_s$ "common component removal"
7: **end for**

Supervised Methods: InferSent (2017)

- Training a sentence embedding model based on Textual Entailment Task

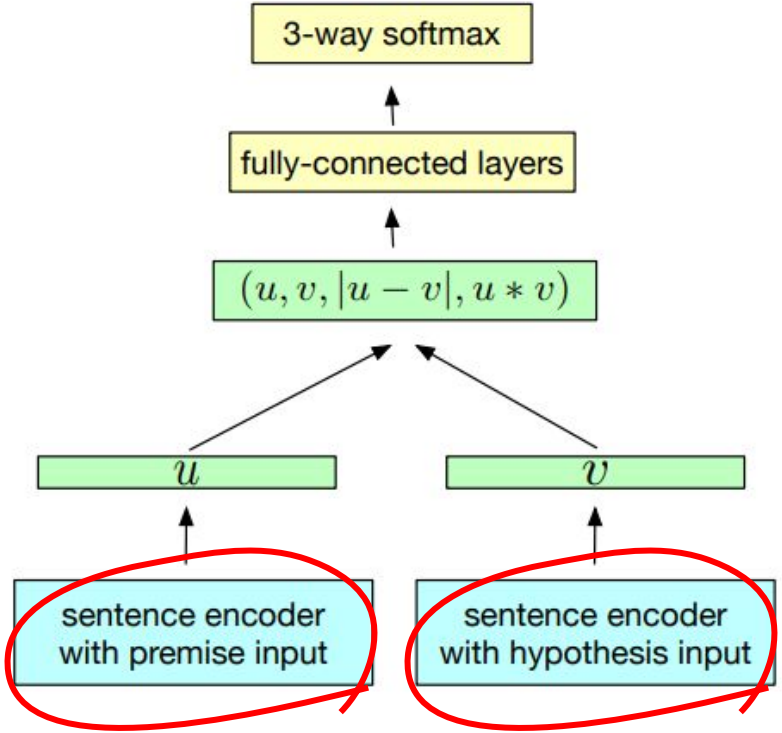


T: The carmine cat devours the mouse in the garden.
H: The red cat killed the mouse.

Label: Entailed (H is entailed by T)

Supervised Methods: InferSent (2017)

- Training a sentence embedding model based on Textual Entailment Task



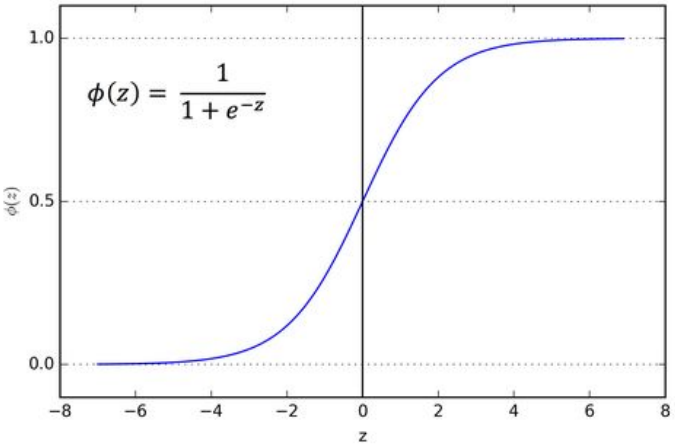
T: The carmine cat devours the mouse in the garden.
H: The red cat killed the mouse.

Label: Entailed (H is entailed by T)

Neural Network Models for NLP

Recap: Logistic Regression

- Linear model + logistic sigmoid function: $y = \sigma(\boldsymbol{w}^T \boldsymbol{x}_i + b)$

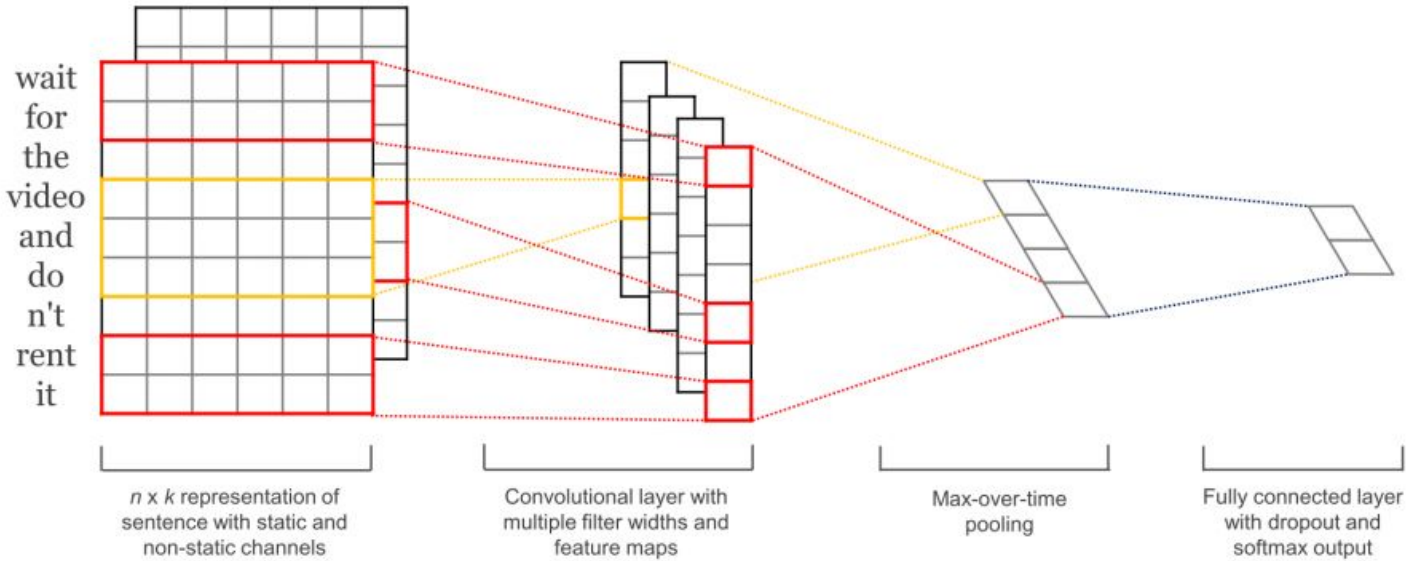


$$P(y = 1|x) = 1/\{1 + \exp(-w^T x)\}$$

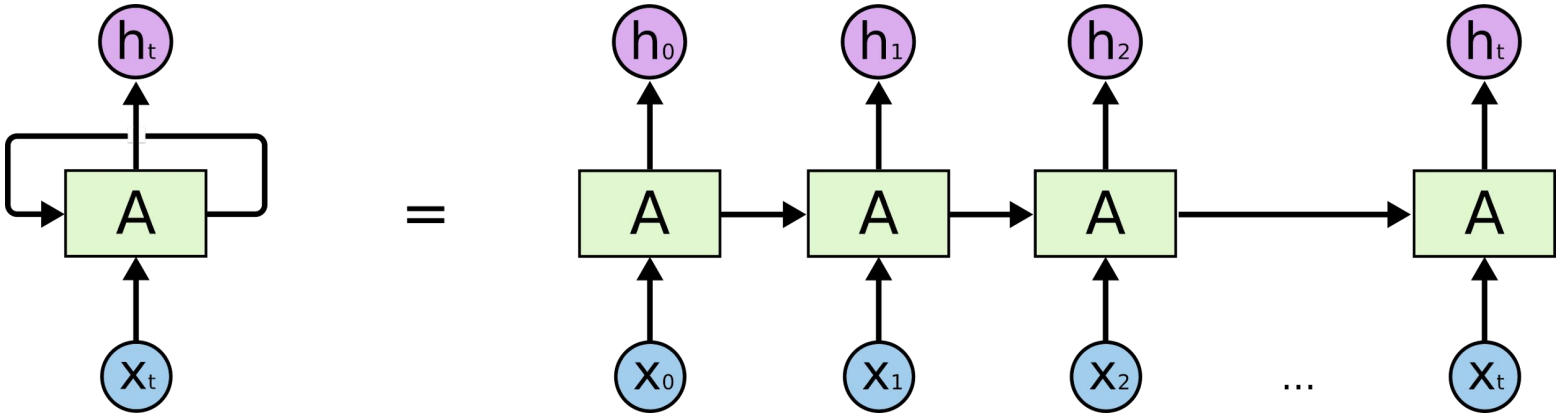
$$L(D; w) = \prod_{i=1}^{N} P(y = 1|x_i)^{y_i} \cdot (1 - p(y = 0|x_i)^{(1-y_i)}$$

Easy extension to multi-class classification using softmax function

Convolutional Neural Networks (CNN)

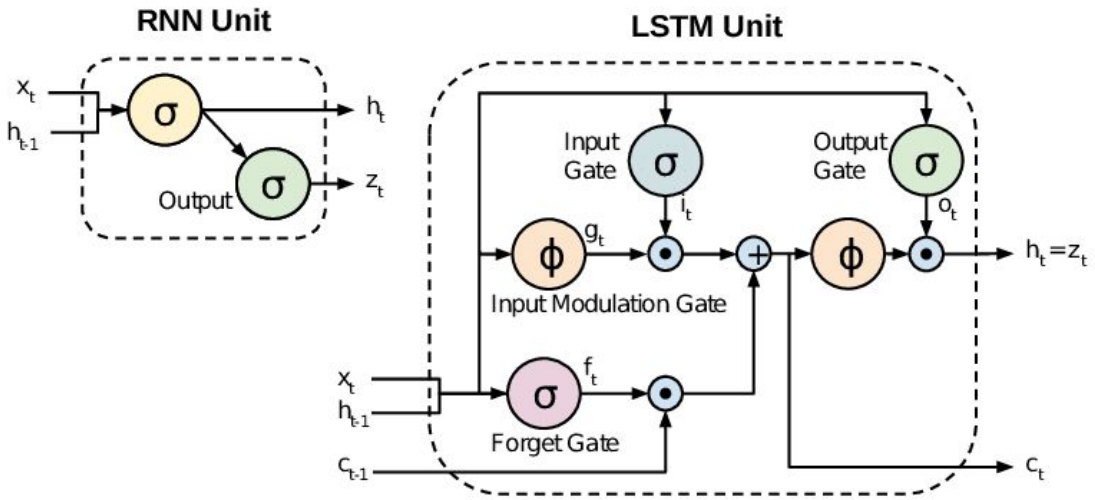


Recurrent Neural Networks (RNN)

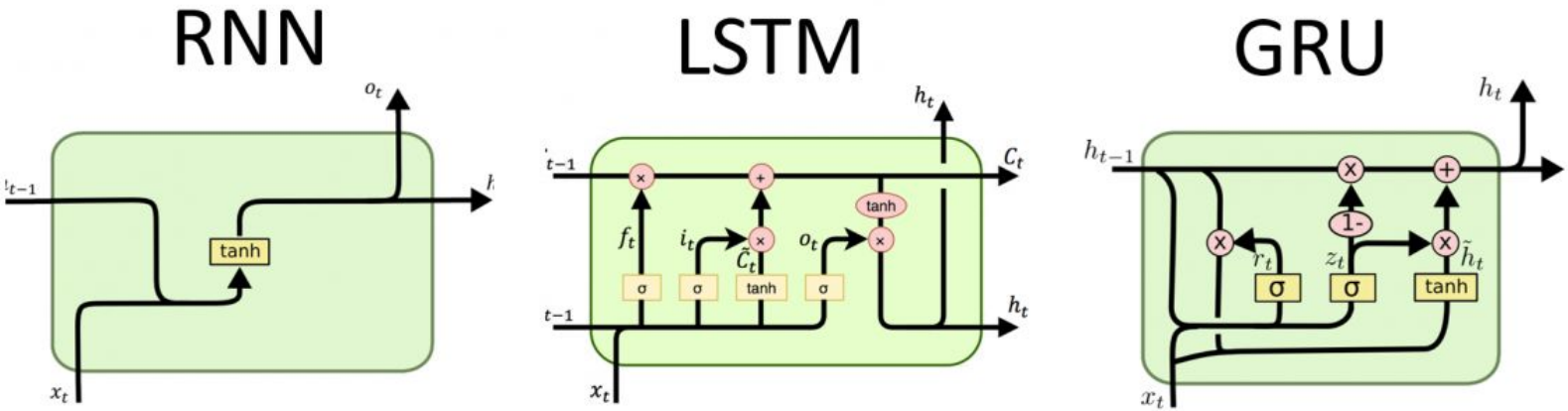


Return of Long Short Term Memory (LSTM) (1997)

- Separate context variable helps to propagate long-distant information



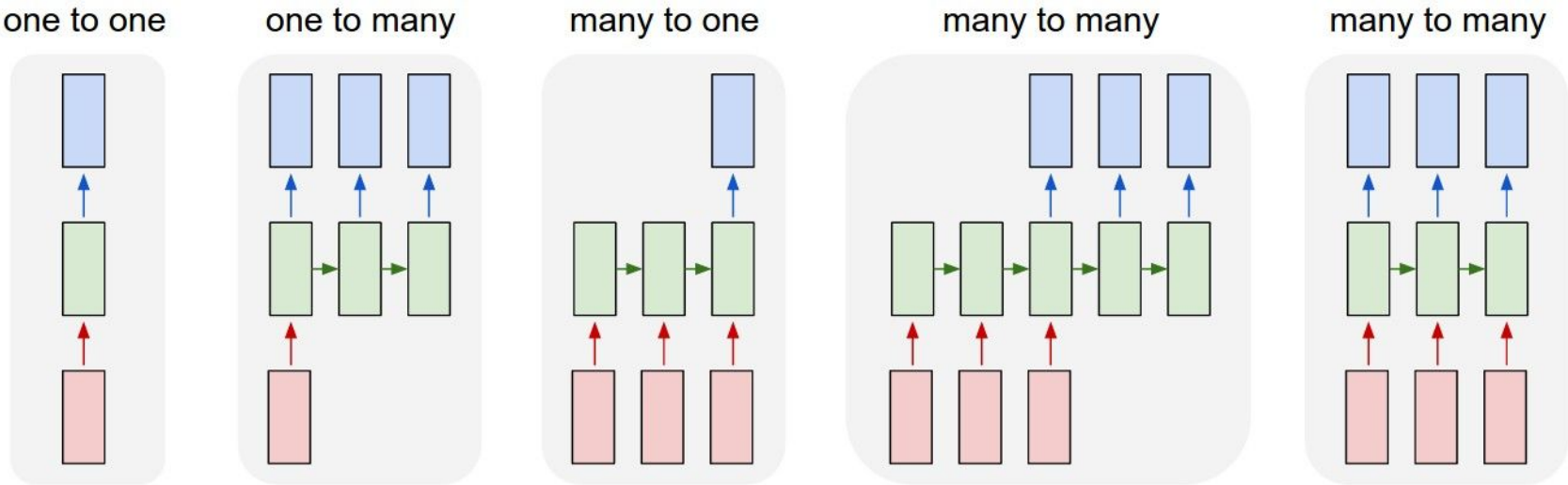
LSTM vs GRU



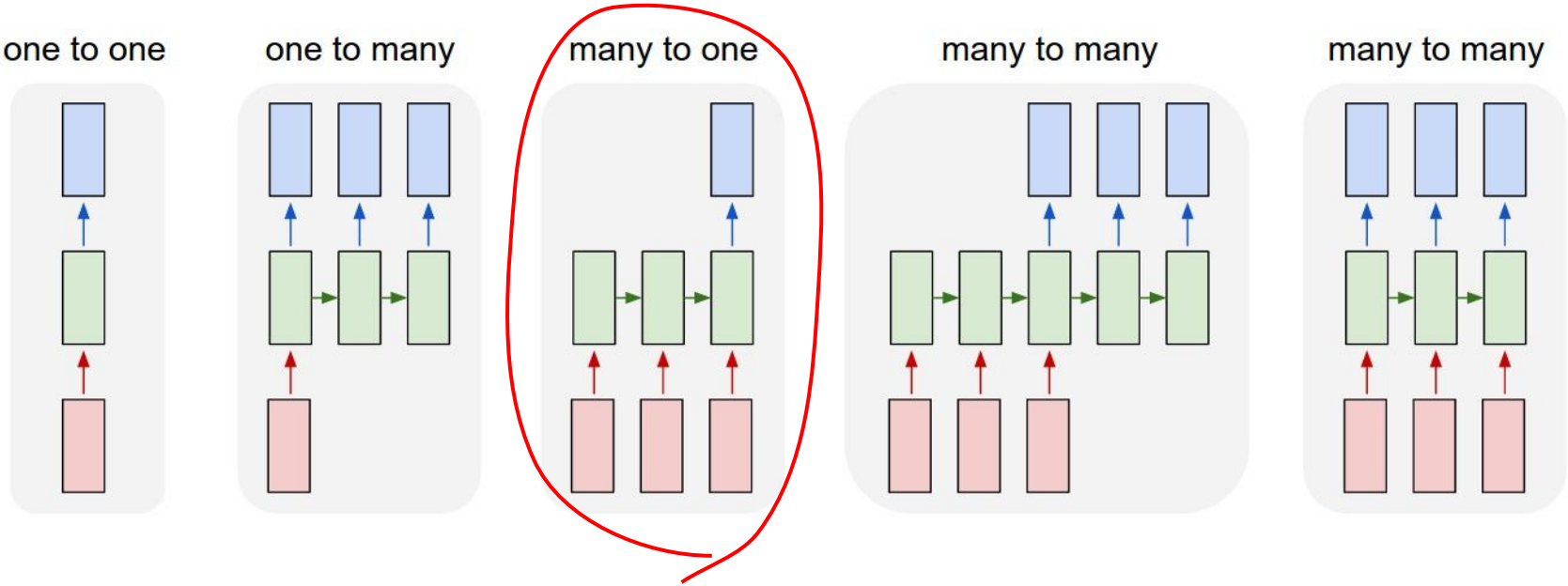
More powerful

More efficient

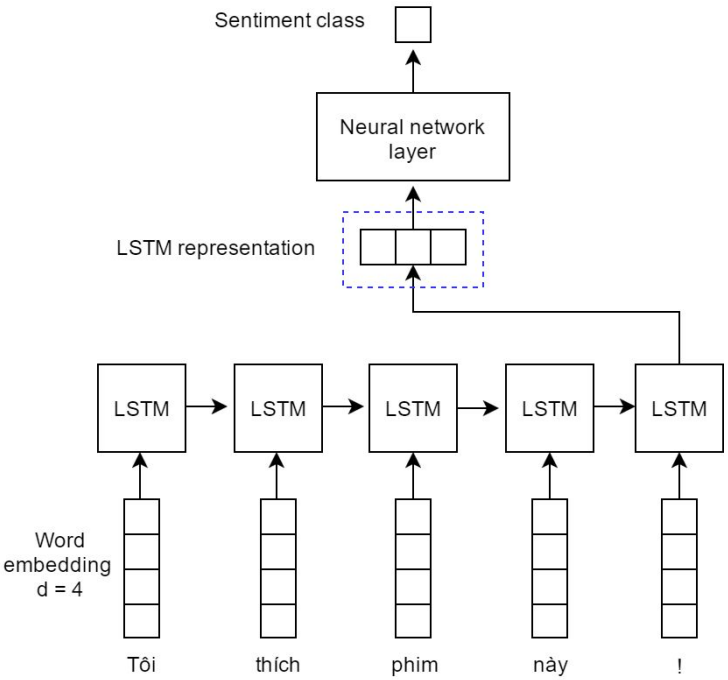
RNN Problem Formulations



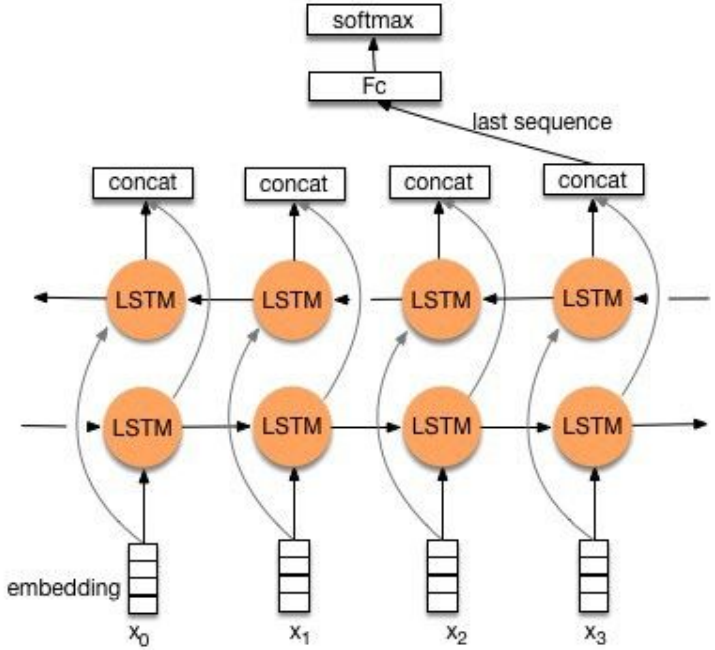
RNN for Text classification



LSTM-RNN for Text Classification

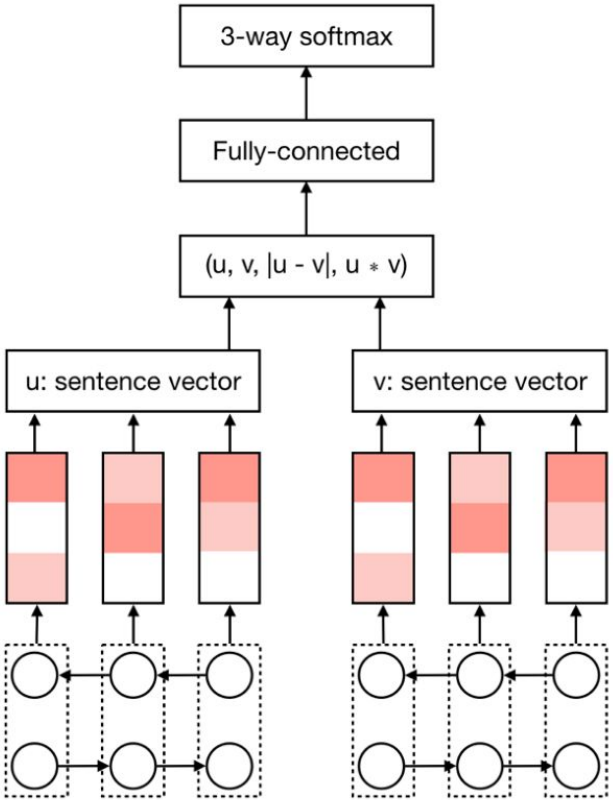


LSTM RNN

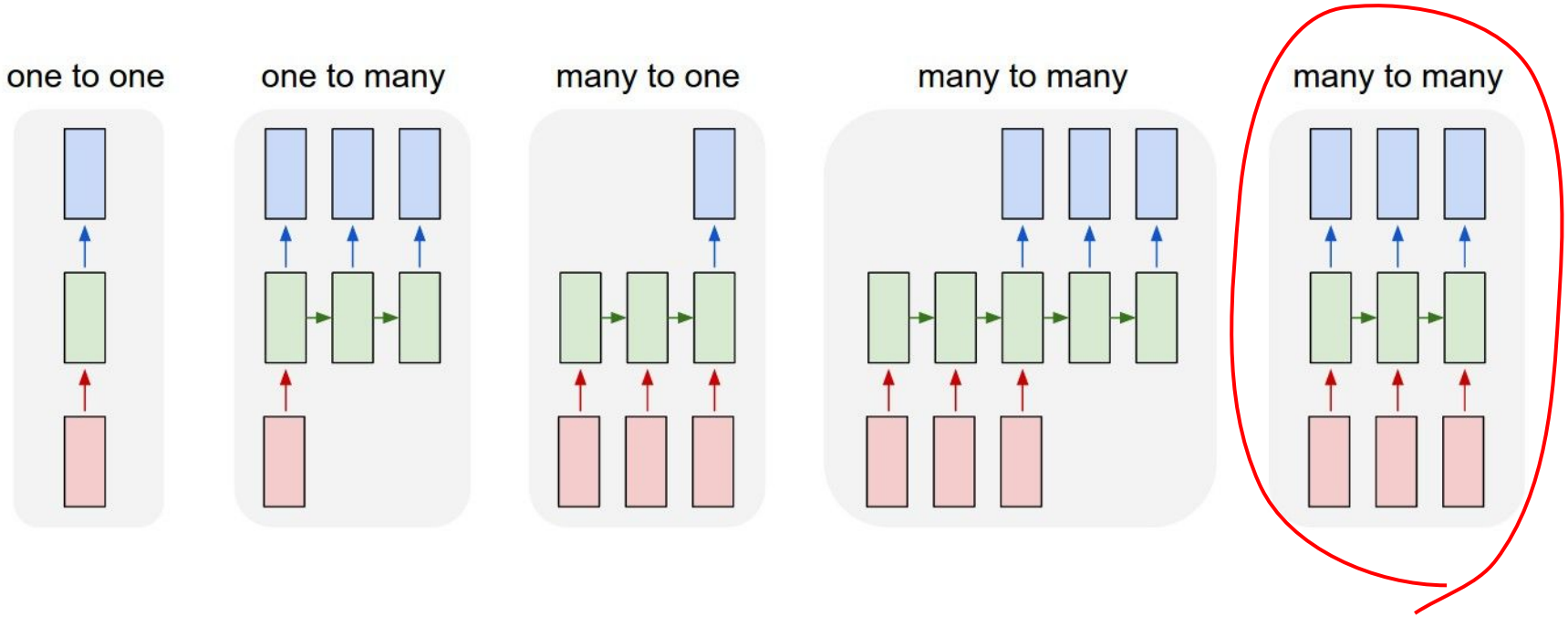


"Bi-LSTM" RNN

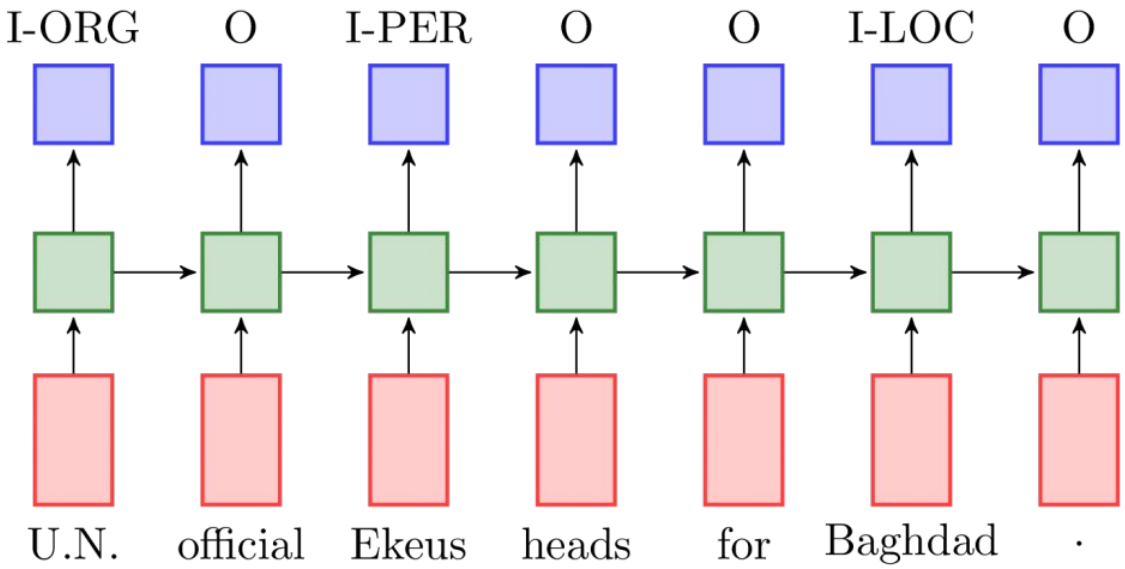
RNN x 2 for Textual Entailment tasks



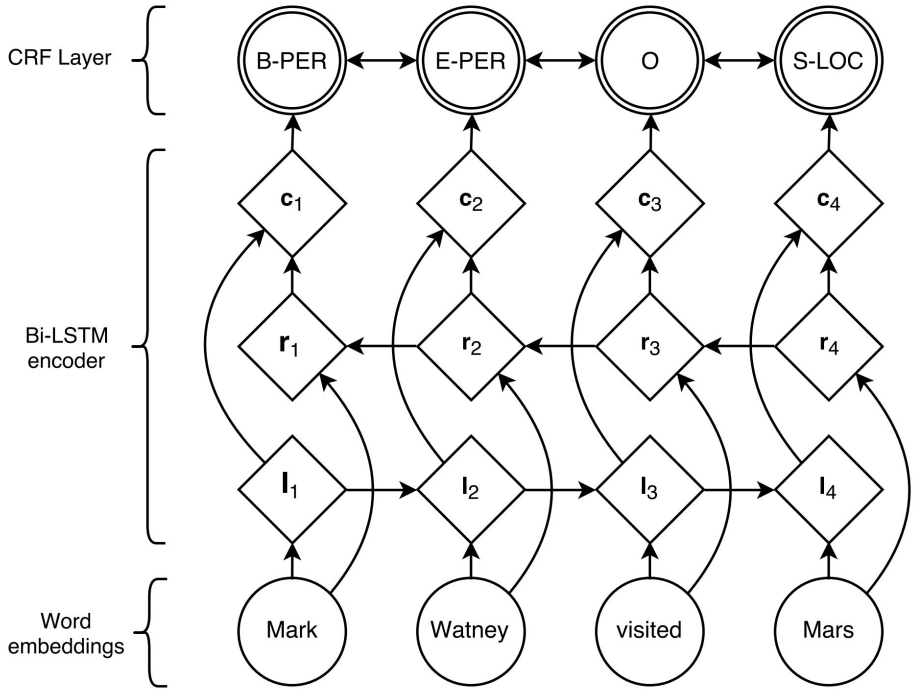
RNN for Sequential Tagging



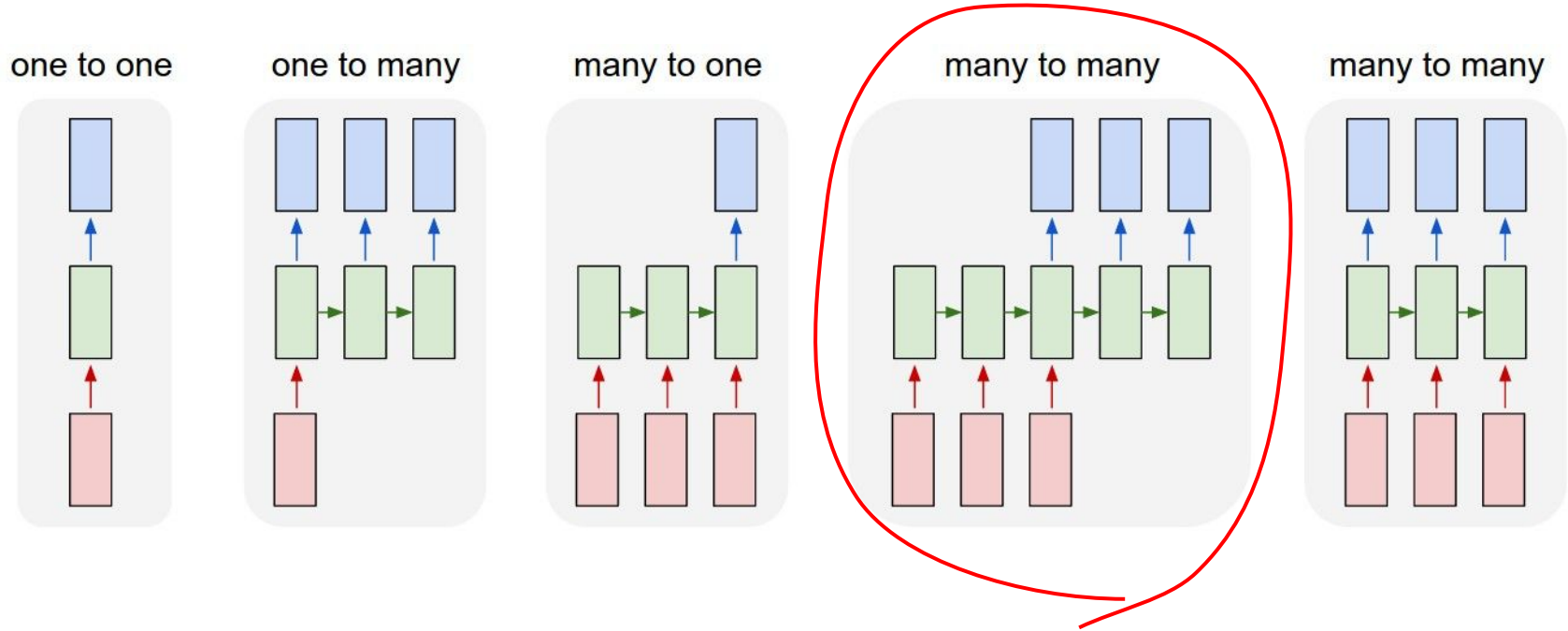
Many-to-many LSTM-RNN for sequential tagging



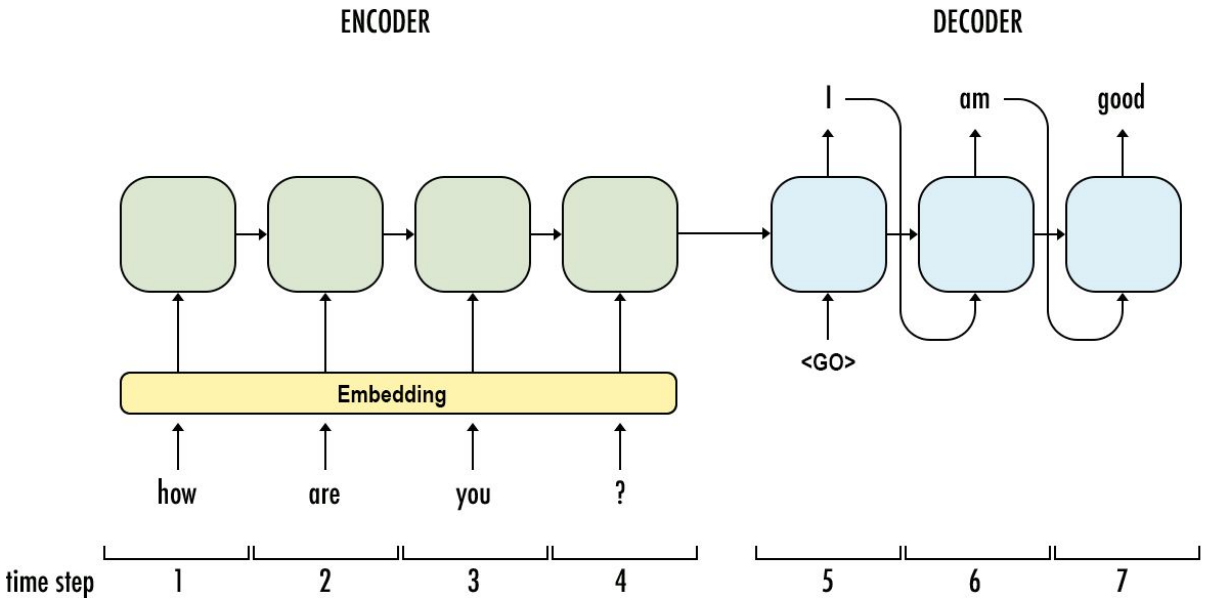
Bi-LSTM CRF (2018)

- 

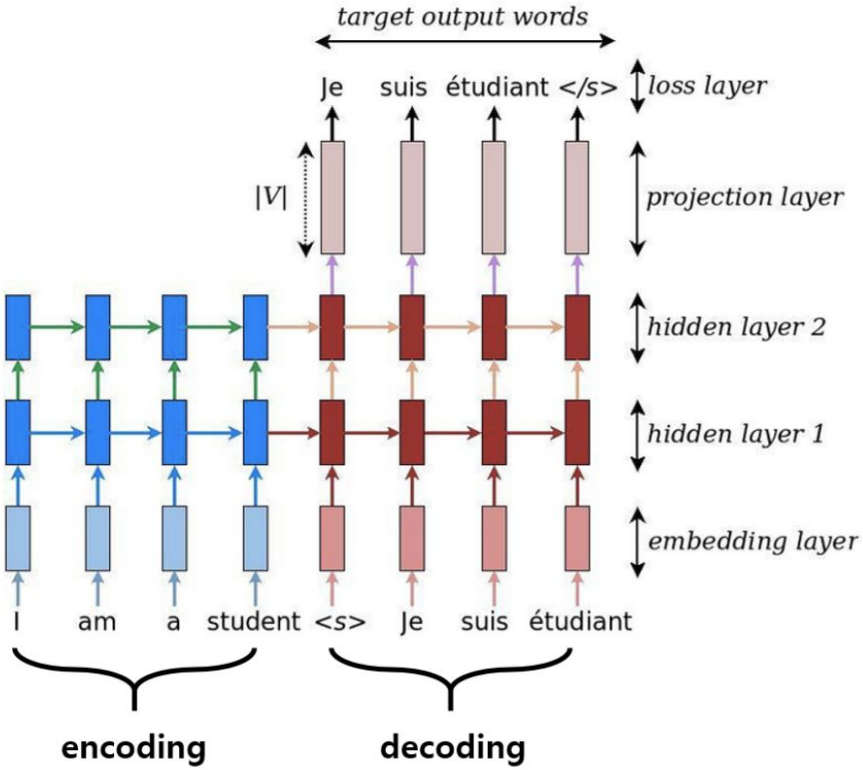
Sequence-to-sequence generation



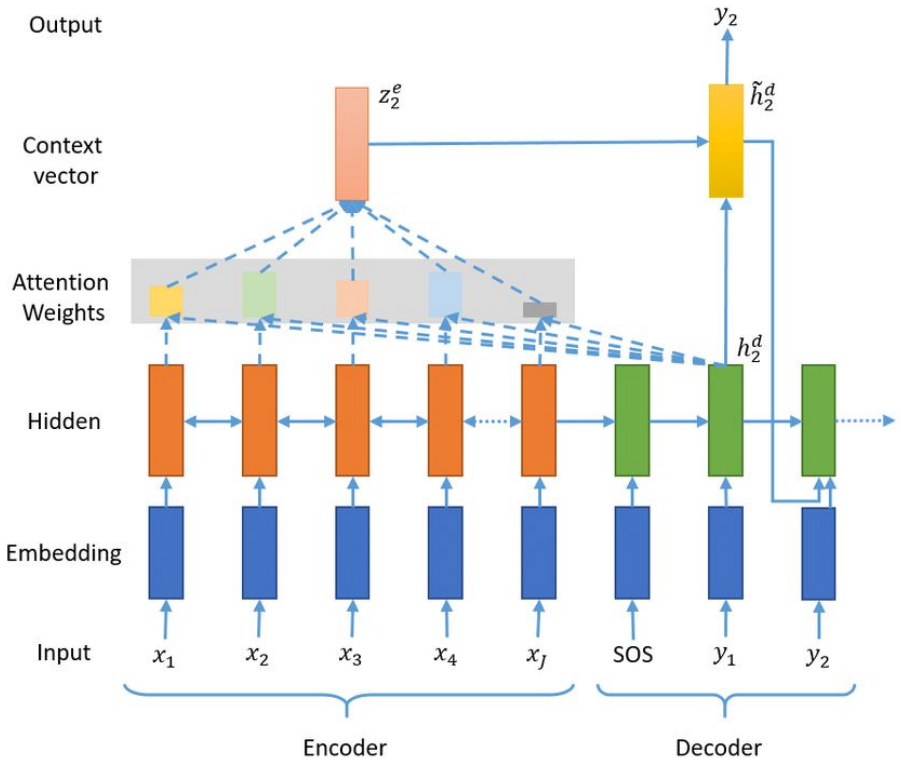
Sequence-to-sequence generation (seq2seq)



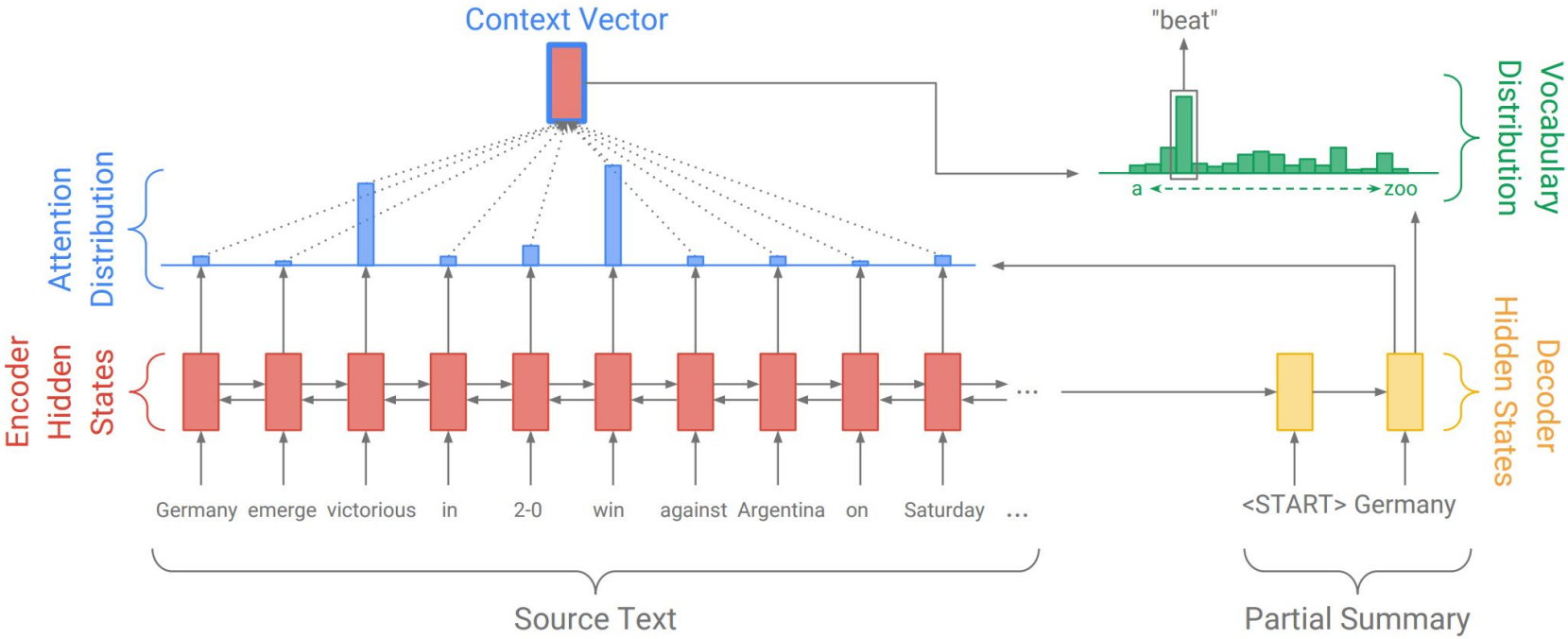
seq2seq Application: Machine Translation



Attention Mechanism (2014)



seq2seq Application: Text Summarization



Break

Advanced DL techniques for NLP

Pre-training models: What and Why?

- Deep learning =~ **Representation learning** + Logistic Regression (linear model)

- **Pre-training** is a technique that learns (hopefully) good representations using (a large amount of) data from different tasks to transfer knowledge to the target task

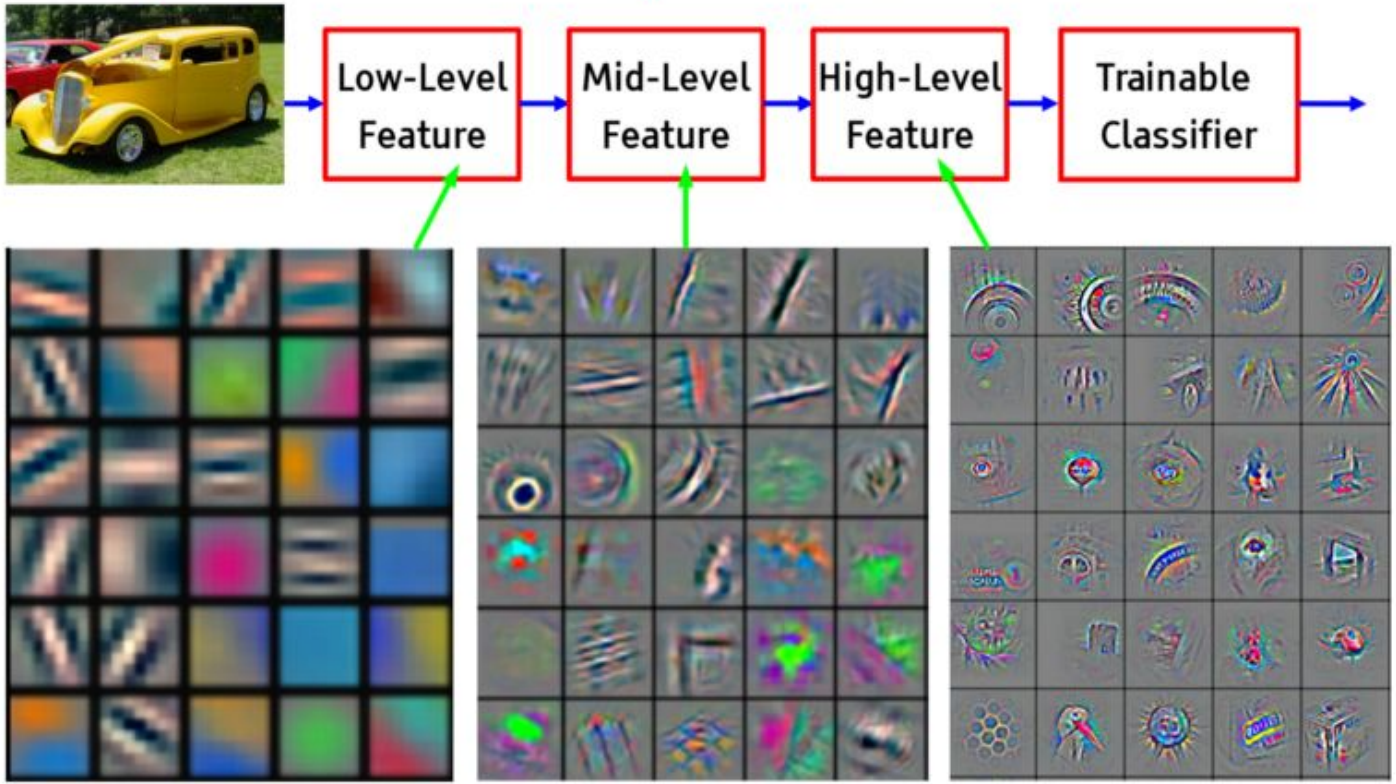
- Pre-training models are very useful to improve many Deep Learning applications (especially in image recognition)
  - And, NLP is not an exception!

Huge success in Image Recognition

Deep Learning = Learning Hierarchical Representations

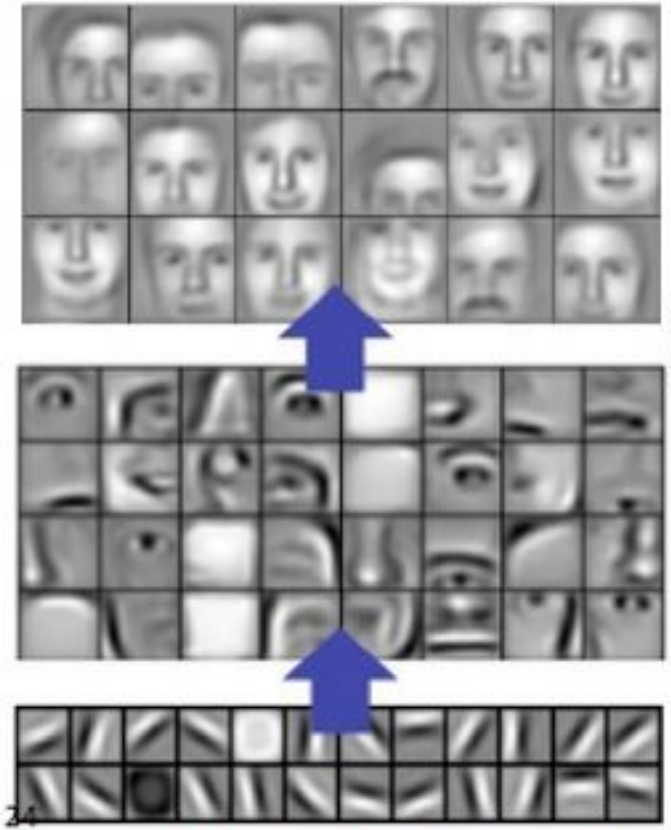
Y LeCun

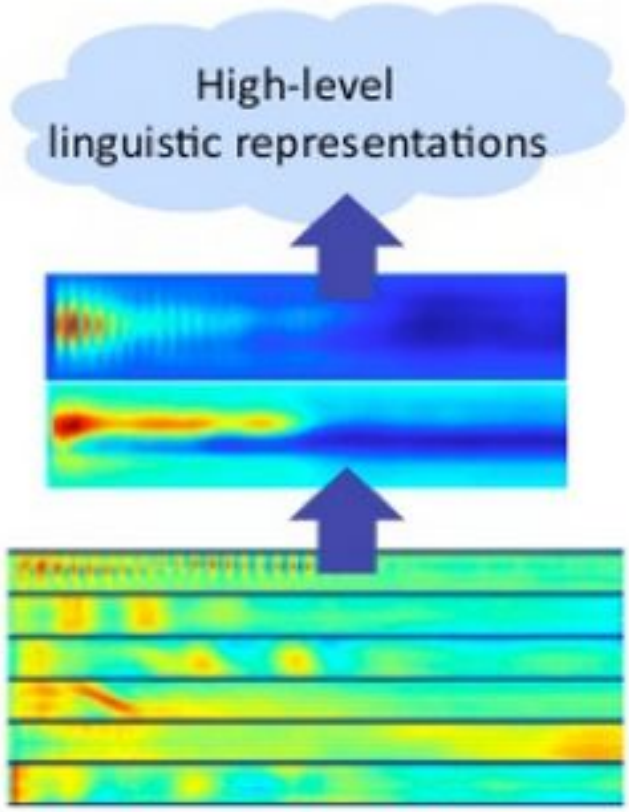
- It's deep if it has more than one stage of non-linear feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

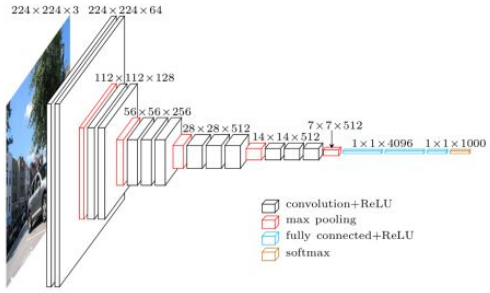
Successive model layers learn deeper intermediate representations

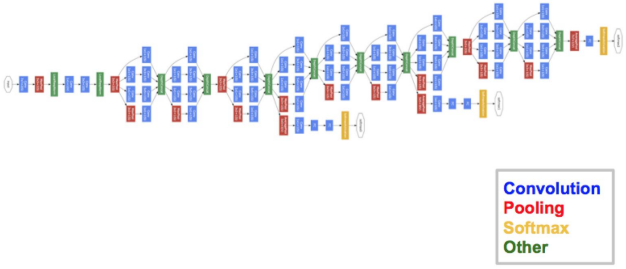


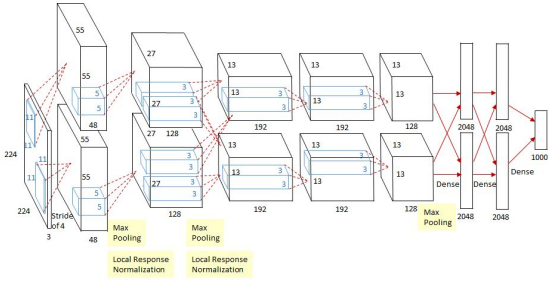


Prior: underlying factors & concepts compactly expressed w/ multiple levels of abstraction


IMAGENET


224×224×3
224×224×64
112×112×128
56×56×256
28×28×512
14×14×512
7×7×512
1×1×4096
1×1×1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

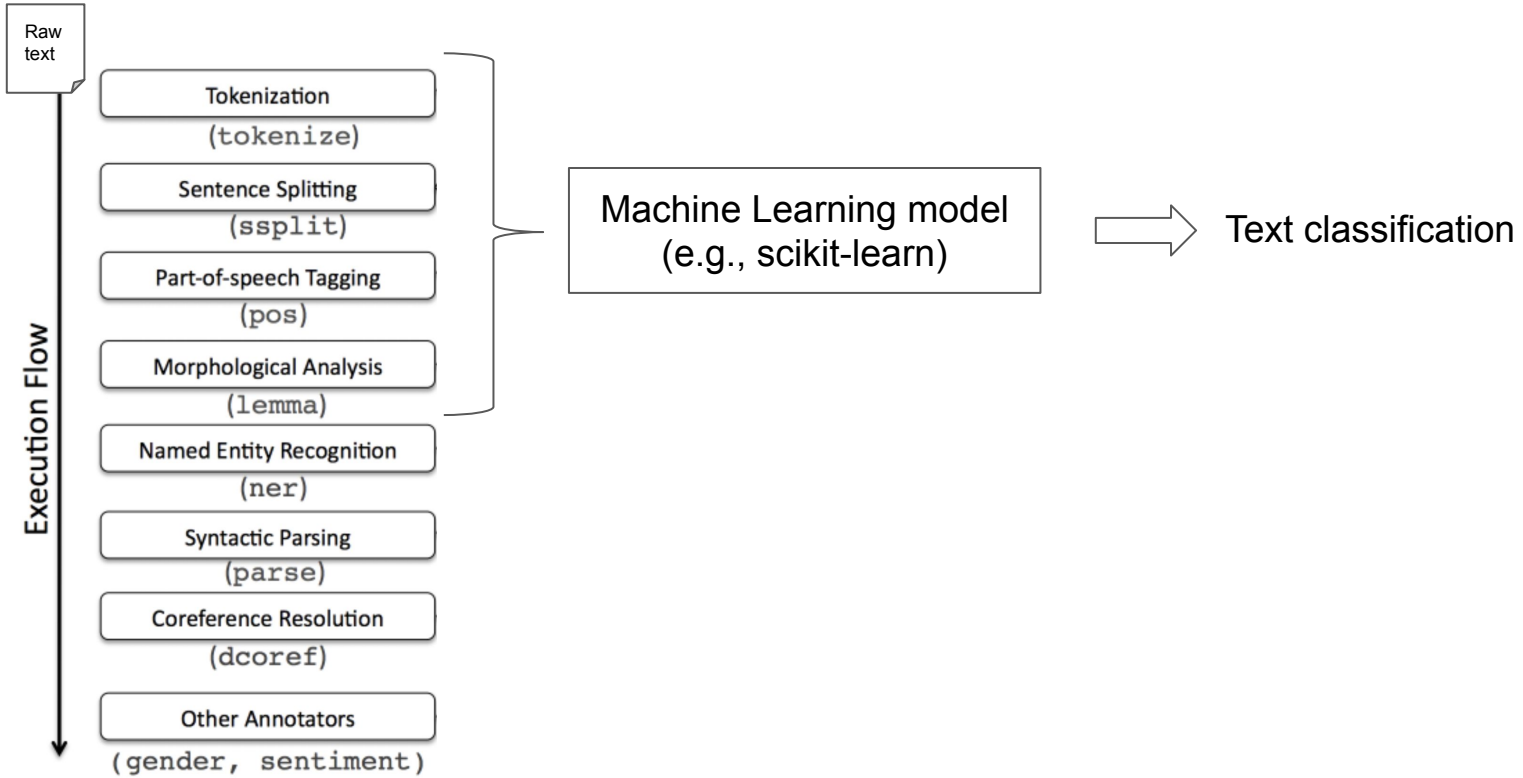

Convolution
Pooling
Softmax
Other


Max Pooling
Local Response Normalization
Dense

Conventional NLP Pipeline

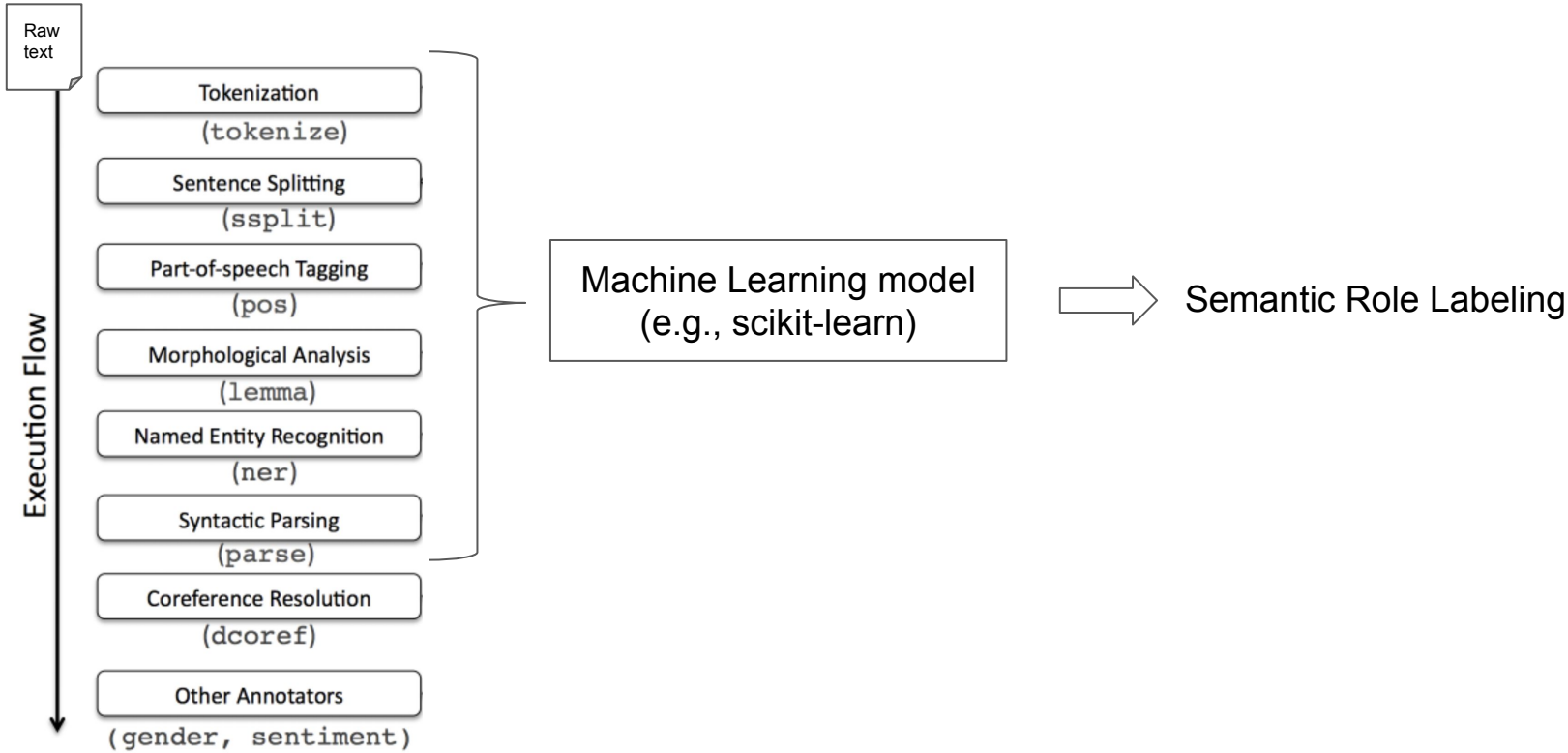
Conventional NLP pipeline

NLTK, spaCy, Stanford CoreNLP etc.

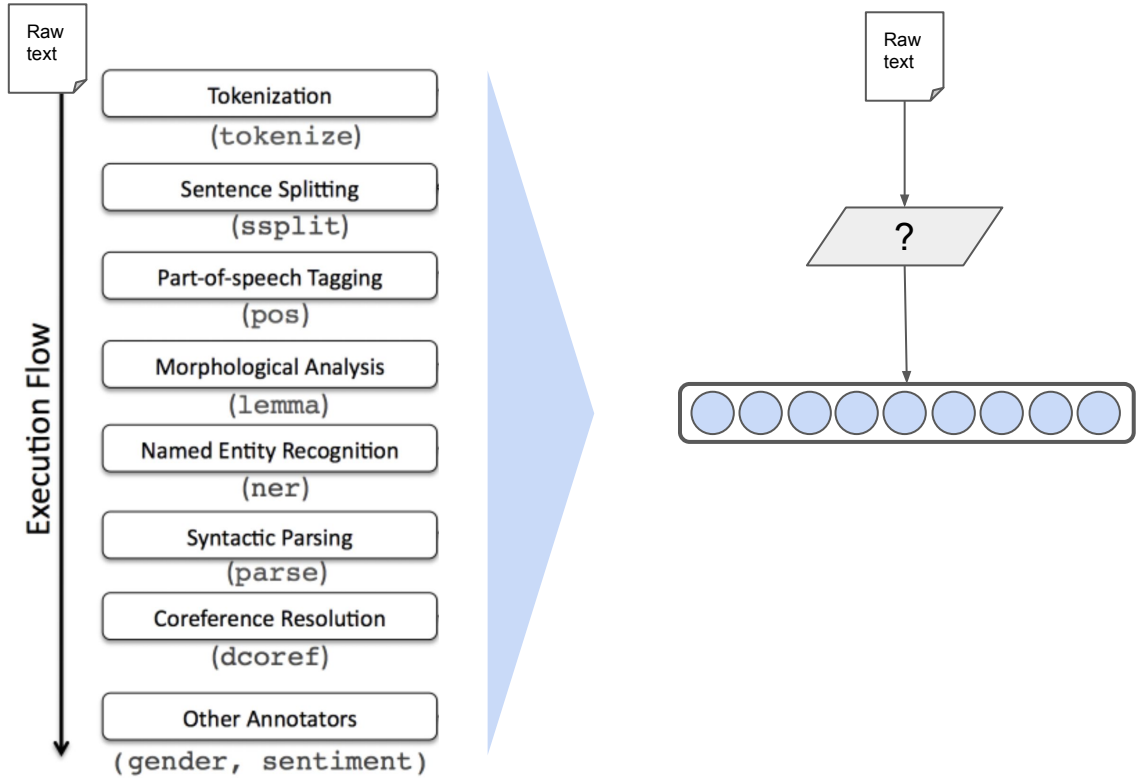


Conventional NLP pipeline

NLTK, spaCy, Stanford CoreNLP etc.

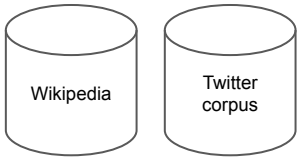


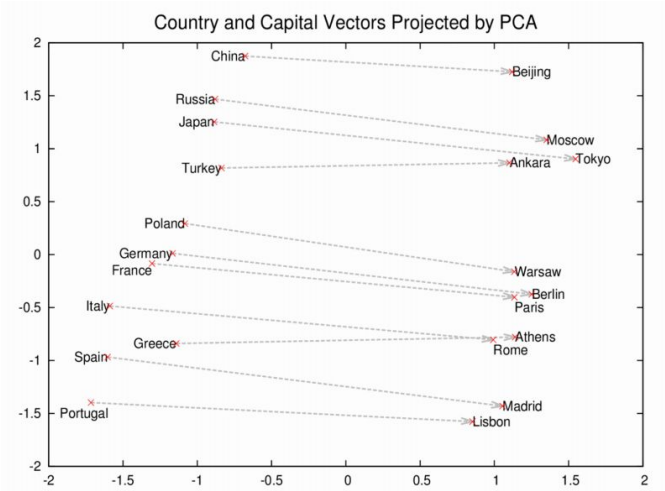
Question: Can we encode different types of syntactic and semantic information into vector representation?



Pre-training models (as of 2016): "Word" embeddings

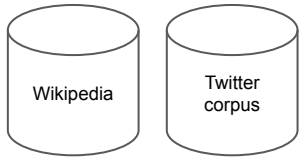
- word2vec, GloVe

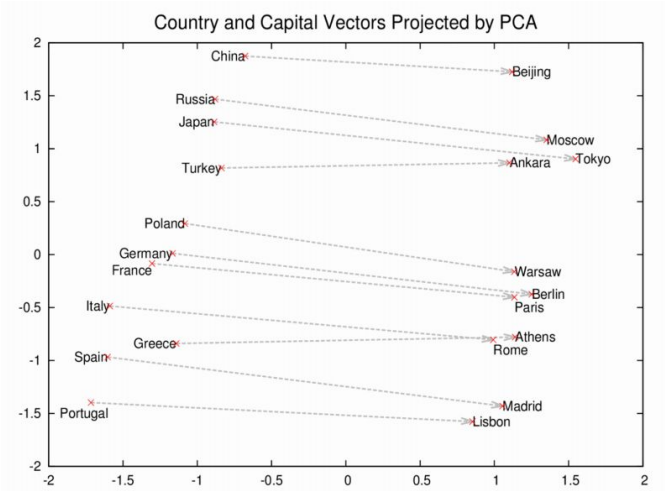


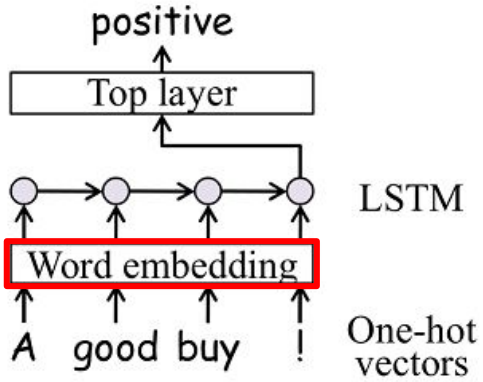


Pre-training models (as of 2016): "Word" embeddings

- word2vec, GloVe
  - Use a "pre-trained model" (e.g., word2vec model) as **initial parameters of the embedding layer** of a Deep Learning model





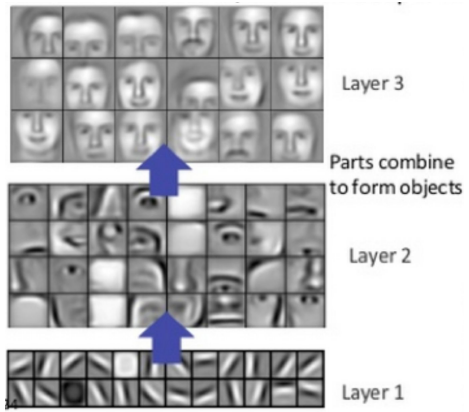


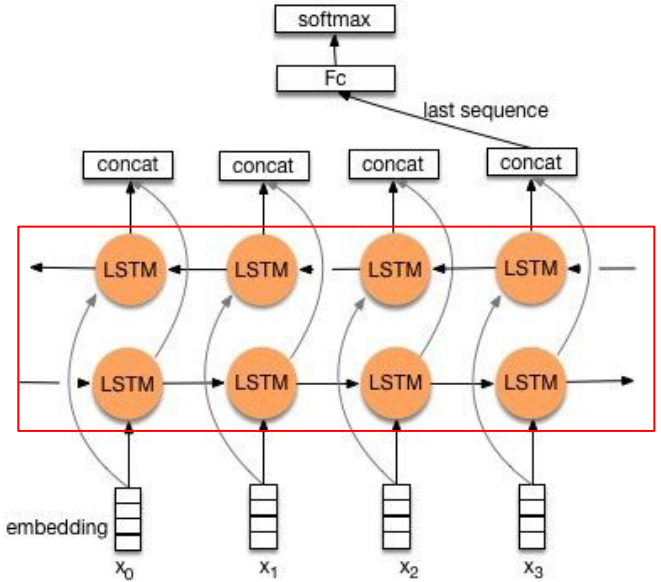
Limitations of word embeddings

- Issue 1) A same word is always converted into the same embedding vector :(
  - Cannot handle the polysemy issue (e.g., apple)
- Issue 2) Cannot take into account **contextual** information :(
  - Word embedding only considers the input word

Word embeddings to contextualized embeddings

- "Contextualized" embeddings
  - e.g., Language models as pre-trained models (ELMo)





Why don't we use deeper hidden states representations?

Power of "contextualized" embeddings

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

Recent Advances in Pre-training Models for NLP

2017-2018: ELMo and BERT etc.

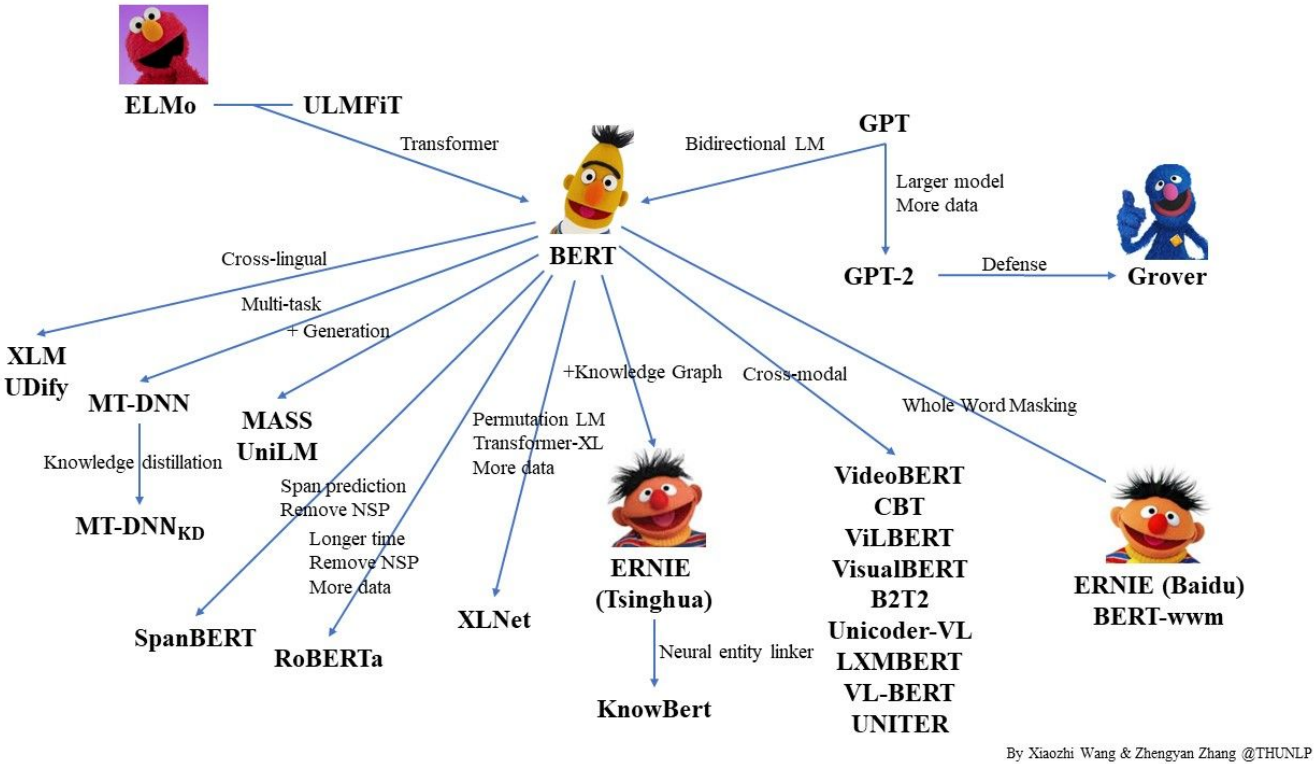
- **Apr 2017: TagLM by AI2**
  - The original version of ELMo
  - Presented at ACL 2017 (July 2017)
- **Aug 2017: CoVe by Salesforce**
  - Presented at NIPS 2017 (Dec 2017)
- **Oct 2017: ELMo by AI2**
  - Presented at NAACL-HLT 2018 (Jun 2018)
- **Jun 2018: Generative Pre-Training (GPT) by OpenAI**
  - https://blog.openai.com/language-unsupervised/
  - Preprint (Right after NAACL '18)
- **Oct 2018: BERT by Google AI**
  - Announced publishing pre-trained model during EMNLP '18
  - Preprint (2 weeks before EMNLP '18)

2017-2018: ELMo and BERT etc.

- **Apr 2017: TagLM by AI2**
  - The original version of ELMo
  - Presented at ACL 2017 (July 2017)
- **Aug 2017: CoVe by Salesforce**
  - Presented at NIPS 2017 (Dec 2017)
- **Oct 2017: ELMo by AI2**
  - Presented at NAACL-HLT 2018 (Jun 2018)
- **Jun 2018: Generative Pre-Training (GPT) by OpenAI**
  - https://blog.openai.com/language-unsupervised/
  - Preprint (Right after NAACL '18)
- **Oct 2018: BERT by Google AI**
  - Announced publishing pre-trained model during EMNLP '18
  - Preprint (2 weeks before EMNLP '18)

Name matters!

After BERT...



https://github.com/thunlp/PLMpapers

3377 citations / 15 months

- 7 new papers every day!!

arxiv.org › cs ▾

BERT: Pre-training of Deep Bidirectional Transformers for ...

by J Devlin - 2018 - Cited by 3377 - Related articles

Oct 11, 2018 - Computer Science > Computation and Language. Title:**BERT**: **Pre-training of Deep Bidirectional Transformers for Language Understanding** ... language representation models, BERT is designed to pre-train deep bidirectional ...

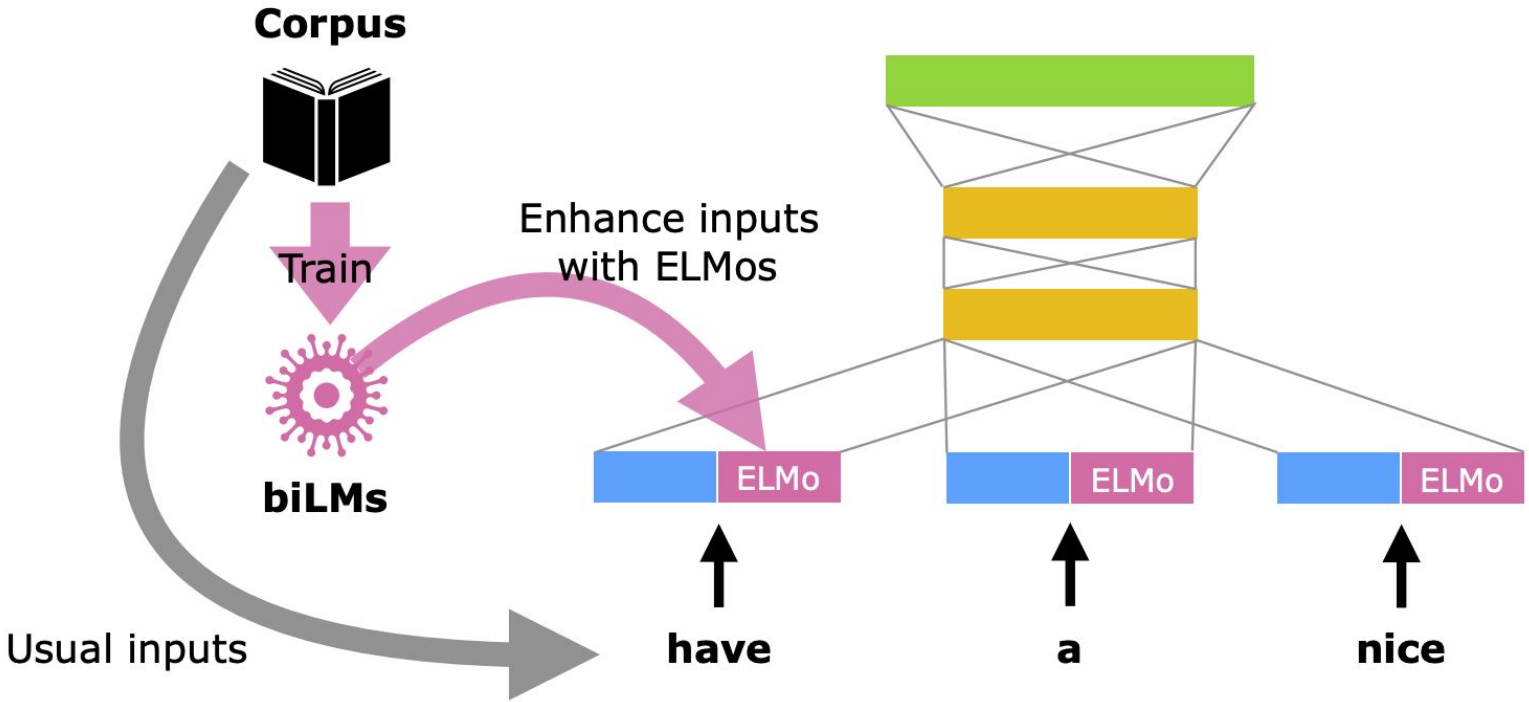
ELMo: Deep contextualized word representations

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer

NAACL-HLT 2018

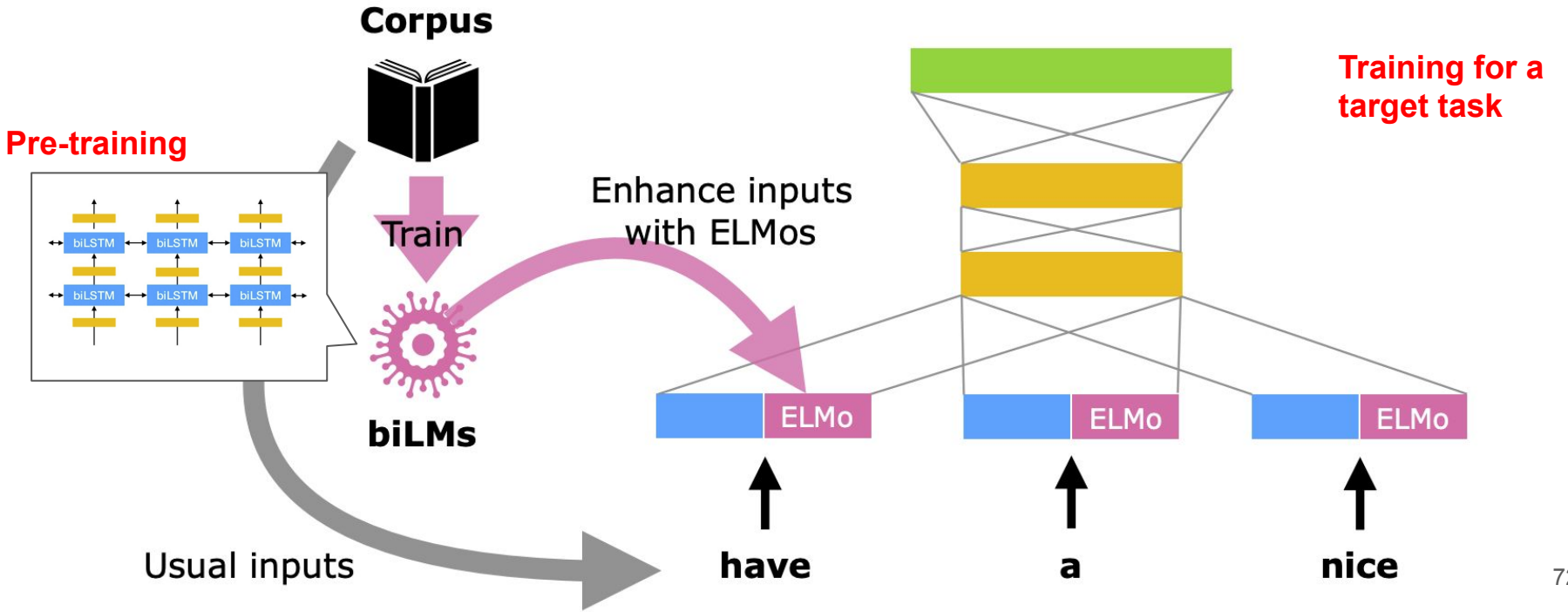
ELMo in a single slide

ELMo can be integrated to almost all neural NLP tasks with simple concatenation to the embedding layer



ELMo in a single slide

ELMo can be integrated to almost all neural NLP tasks with simple concatenation to the embedding layer



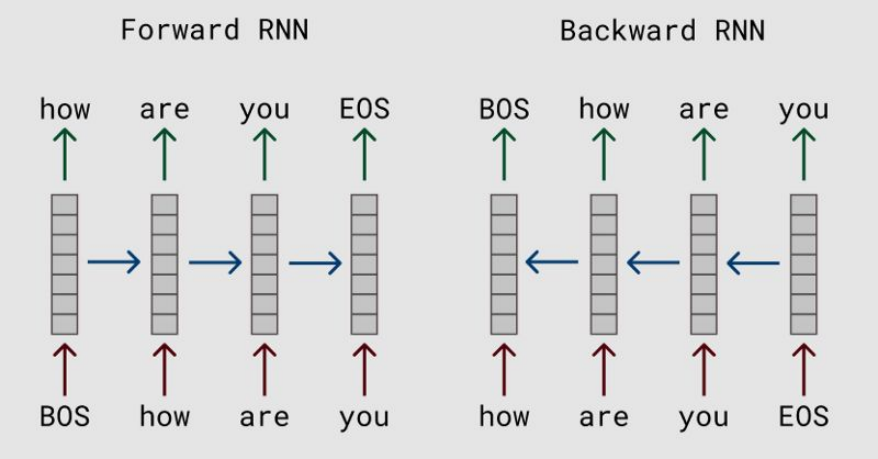
Steps

- Step 1) Train a **language model** using BiLSTM-RNN based on a large-scale corpus
- Step 2) **Integrate hidden states** of the pre-trained model into a model (for a target task) as "features" and **train the final model**

Step 1: Forward/Backward Language Model

- Forward language model
  - Predict the next word given a previous context
- Backward language model
  - Predict the previous word given a later context

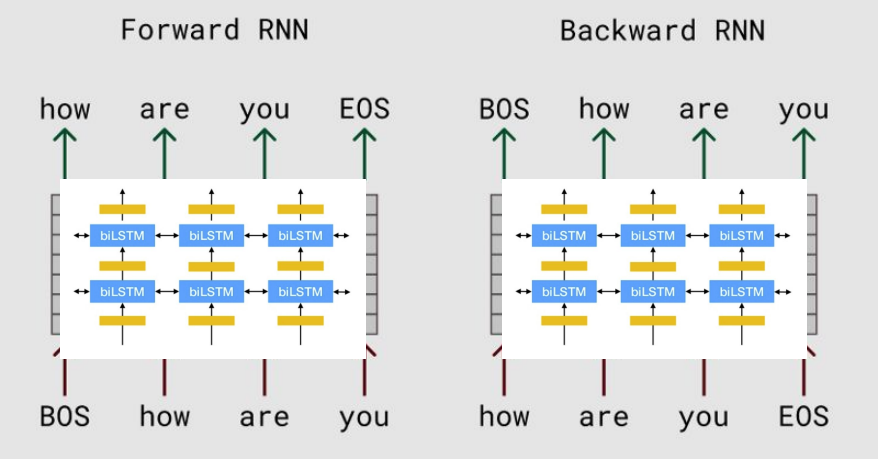
$$\sum_{k=1}^{N} (\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s))$$



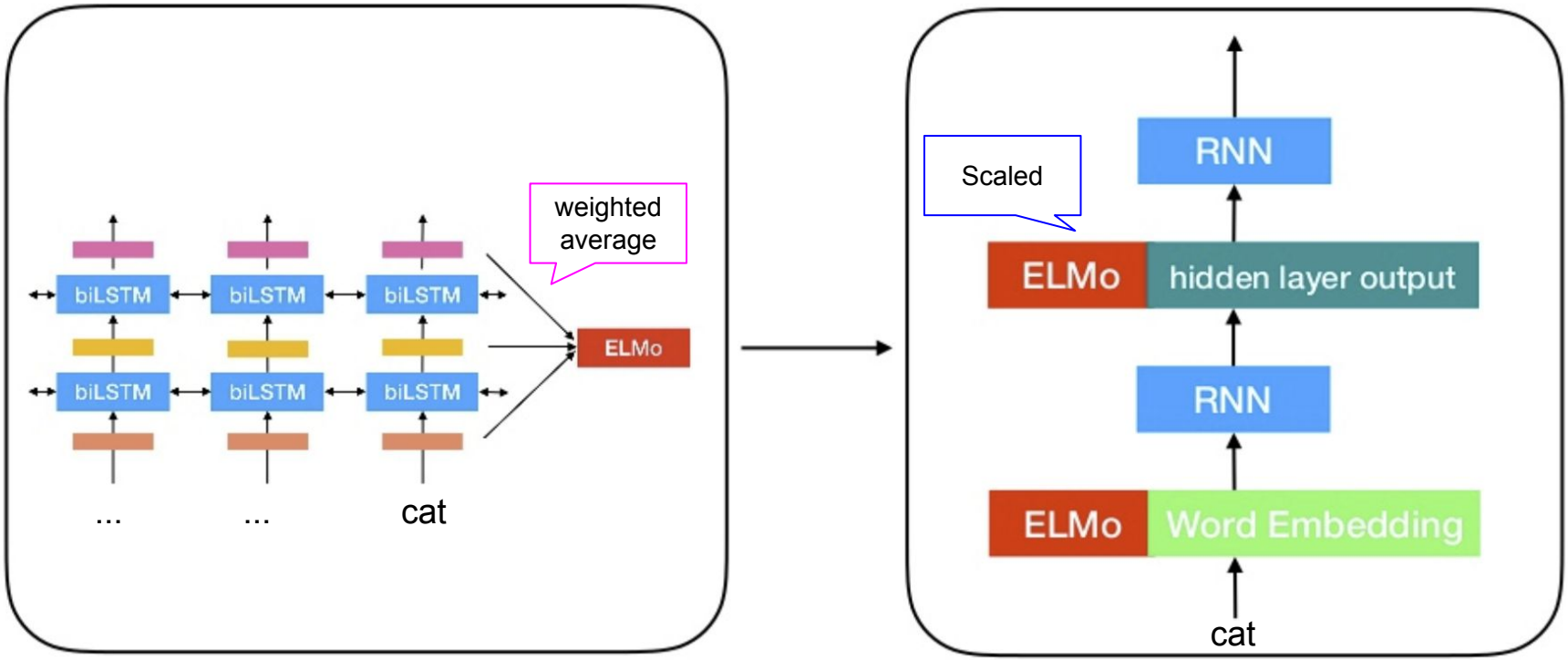
Step 1: Forward/Backward Language Model

- Forward language model
  - Predict the next word given a previous context
- Backward language model
  - Predict the previous word given a later context

$$\sum_{k=1}^{N} (\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s))$$



Step 2: Integrate "ELMo vector" into a target model



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

Many NLP tasks are improved with ELMo

	Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
Q&A	SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
Textual entailment	SNLI	Chen et al. (2017)	88.6	88.0	88.7 $\pm$ 0.17	0.7 / 5.8%
Semantic role labelling	SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coreference resolution	Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
Named entity recognition	NER	Peters et al. (2017)	91.93 $\pm$ 0.19	90.15	92.22 $\pm$ 0.10	2.06 / 21%
Sentiment analysis	SST-5	McCann et al. (2017)	53.7	51.4	54.7 $\pm$ 0.5	3.3 / 6.8%

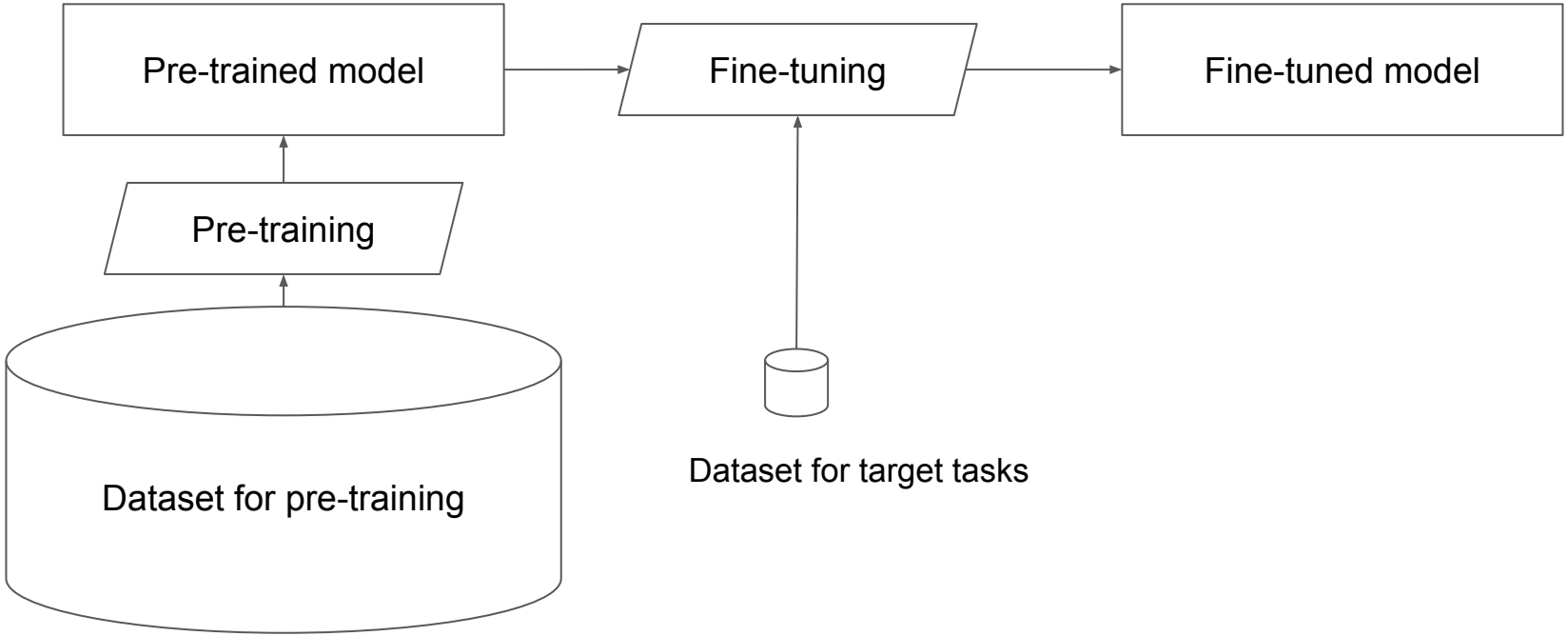
Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

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

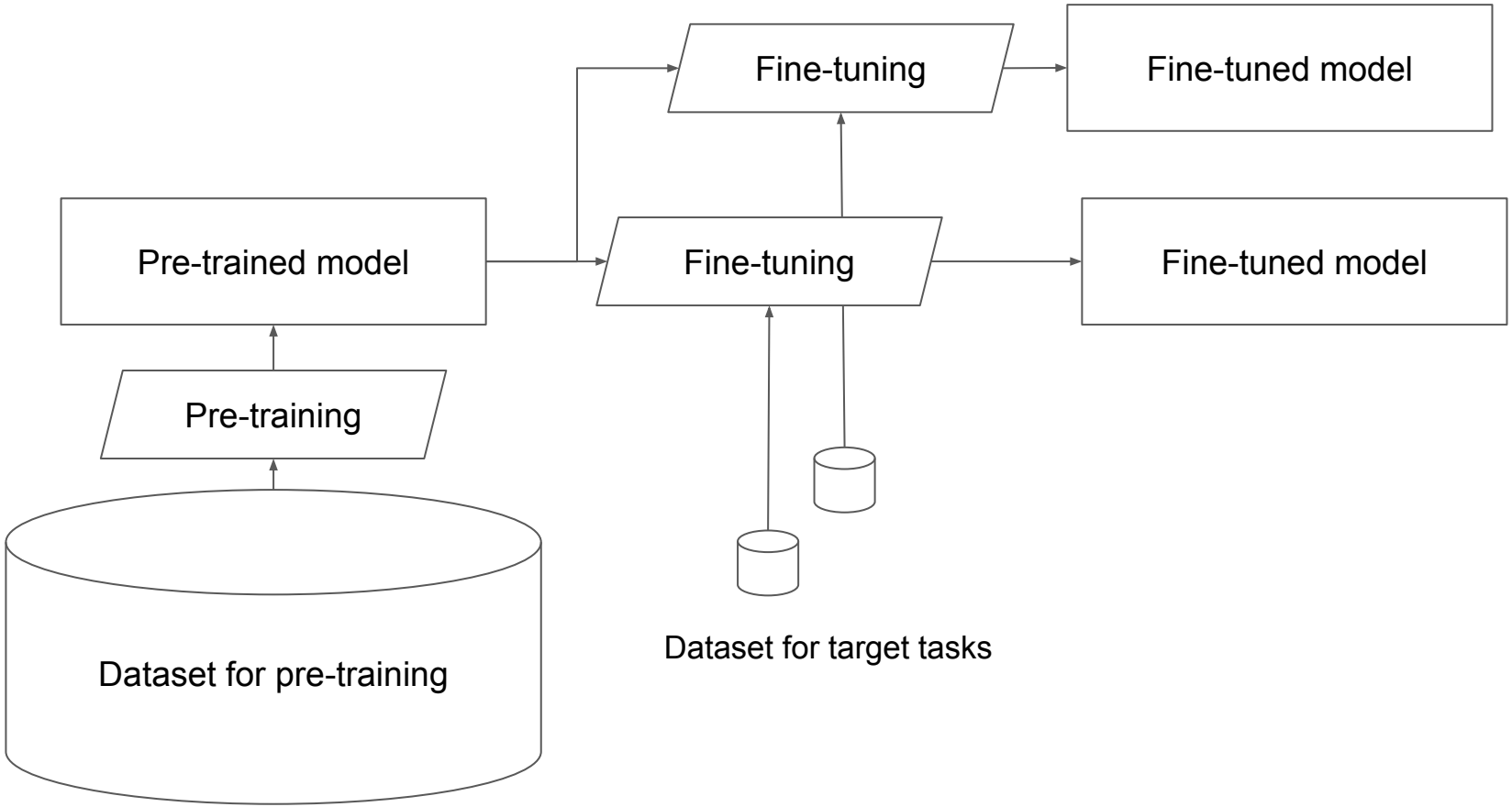
Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

ArXiv Oct 11, 2018

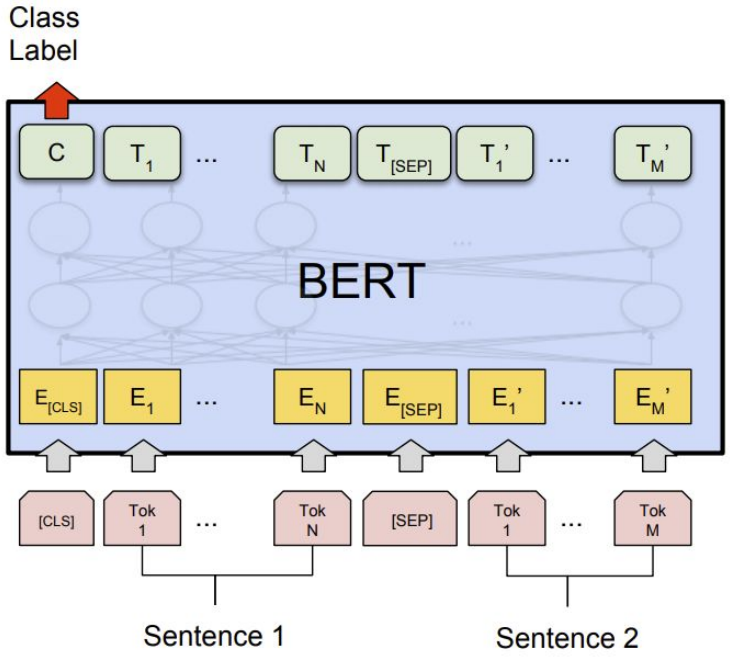
Key Concept: Pre-trained model + Fine-tuning



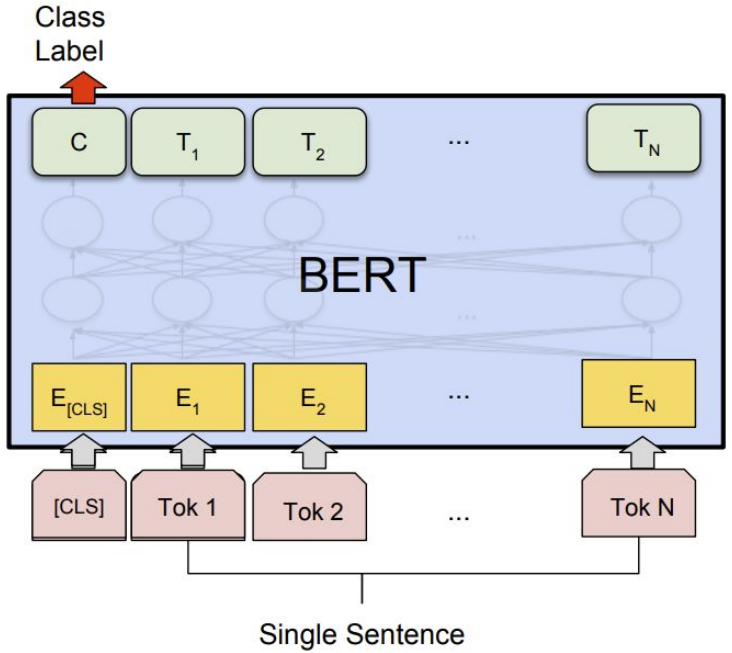
Key Concept: Pre-trained model + Fine-tuning



BERT for different NLP tasks (1/2)

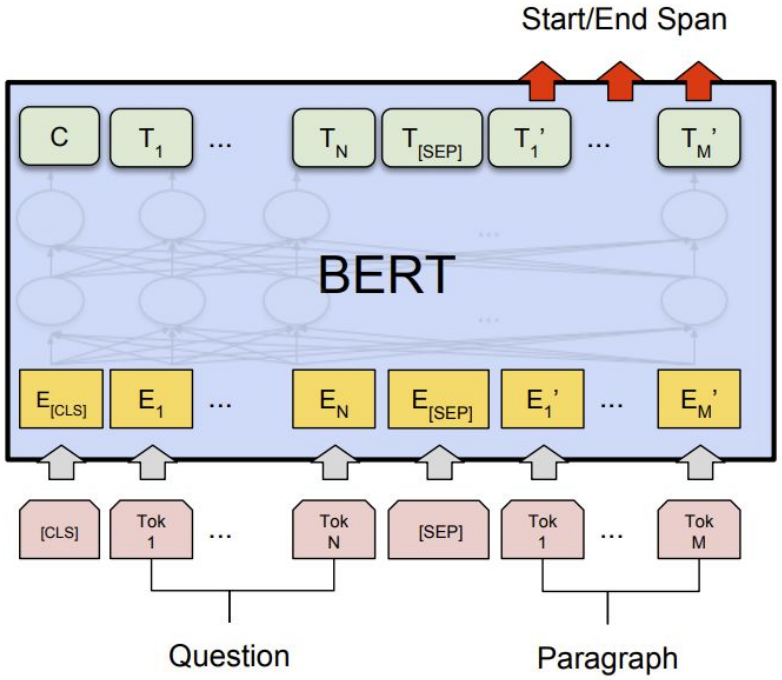


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

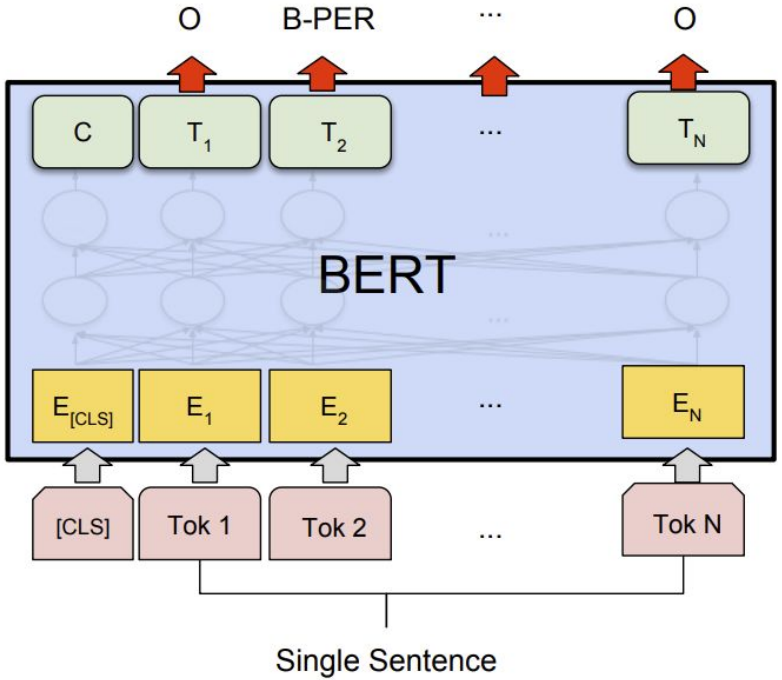


(b) Single Sentence Classification Tasks: SST-2, CoLA

BERT for different NLP tasks (2/2)



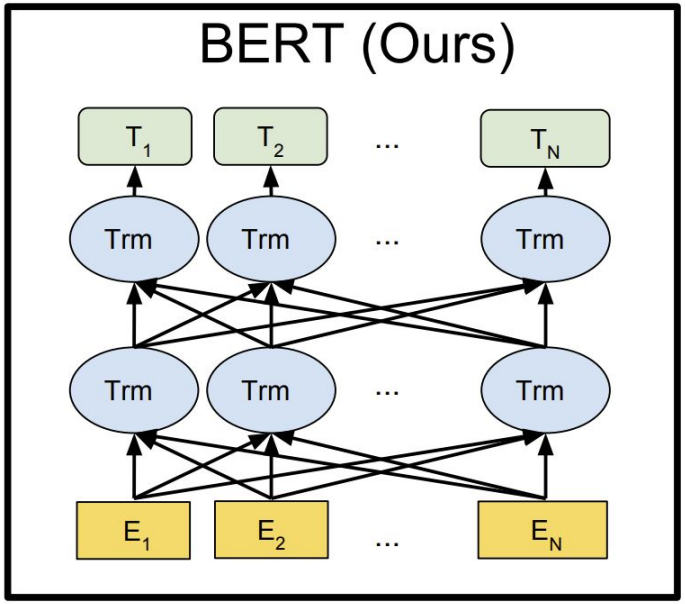
(c) Question Answering Tasks: SQuAD v1.1

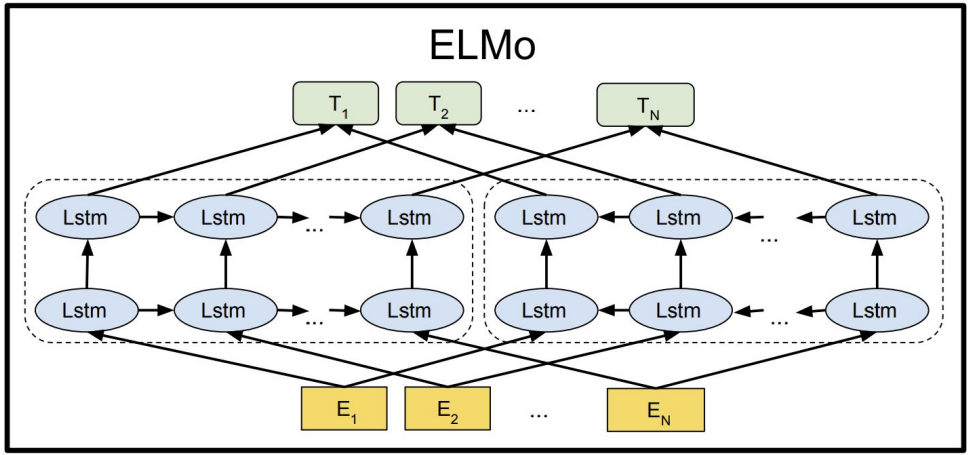


(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

BERT Architecture

- **B**idirectional **E**ncoder **R**epresentation from **T**ransformers
  - which consists of **Transformers** instead of LSTM blocks





The Transformer (NIPS '17)

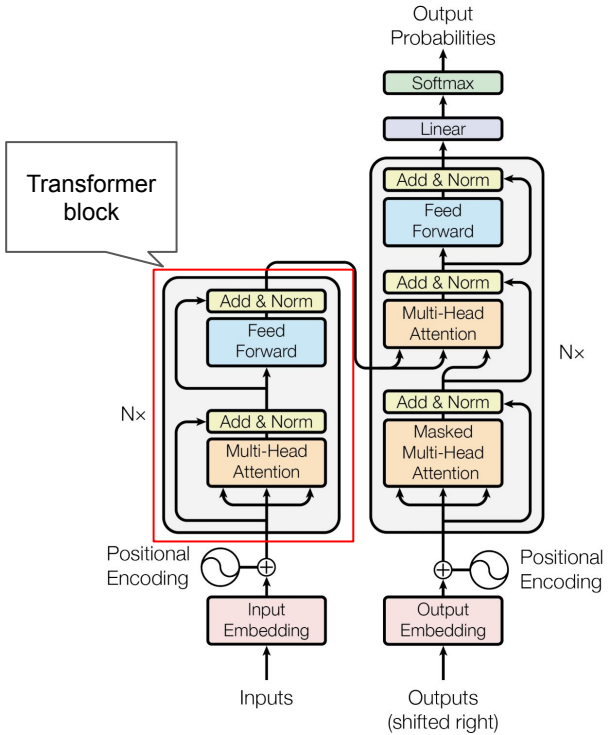


Figure 1: The Transformer - model architecture.

The Transformer (NIPS '17)

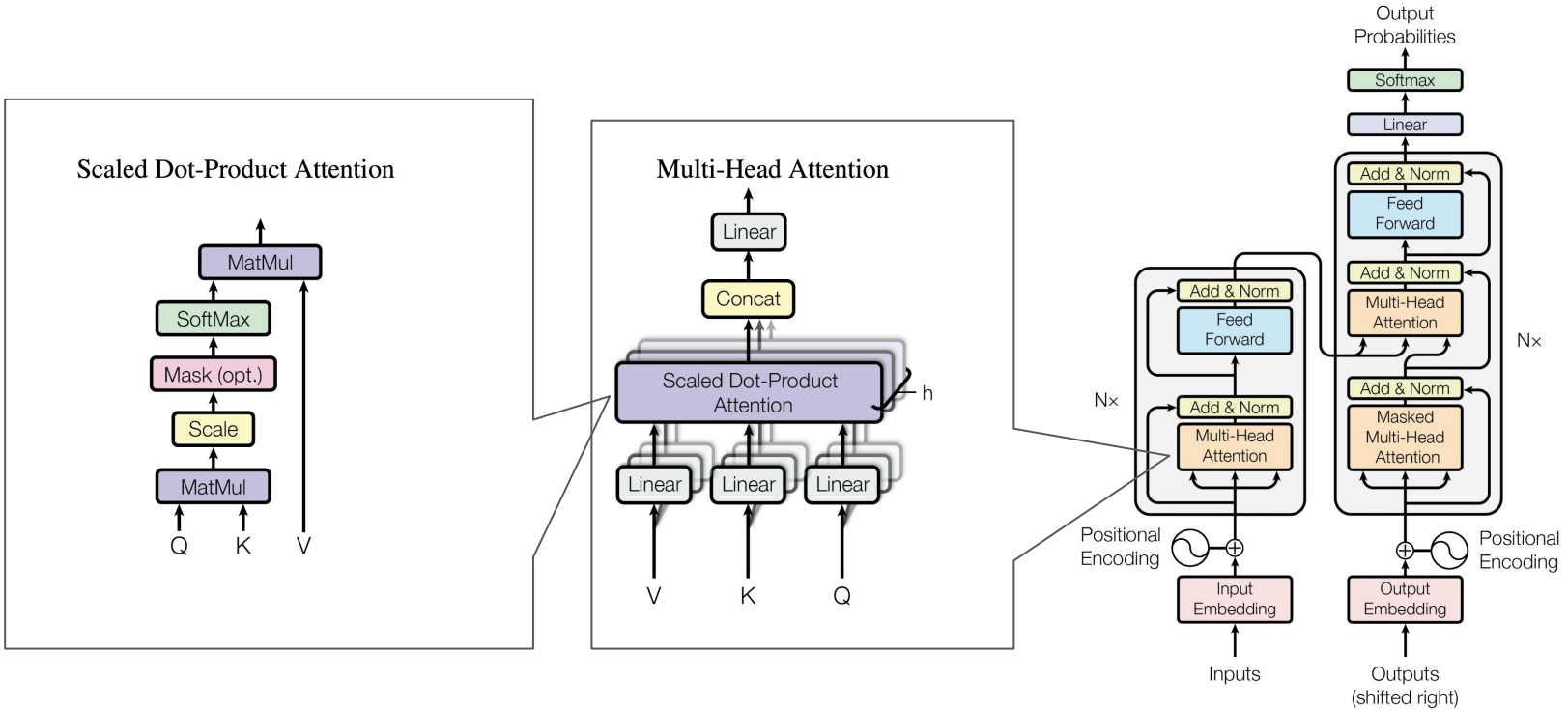


Figure 1: The Transformer - model architecture.

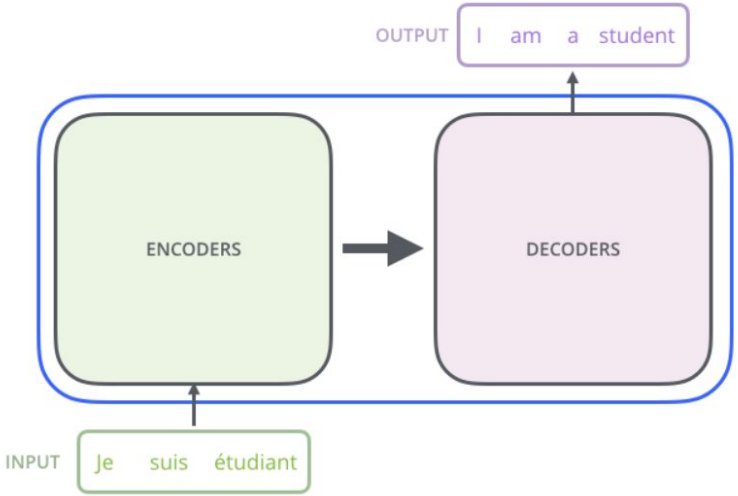
Transformer:
Quick Walkthrough

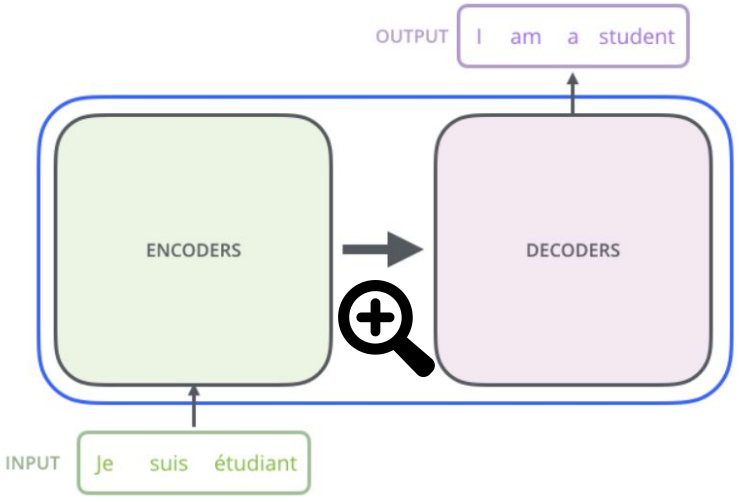
Transformer from the sky

Transformer

Transformer

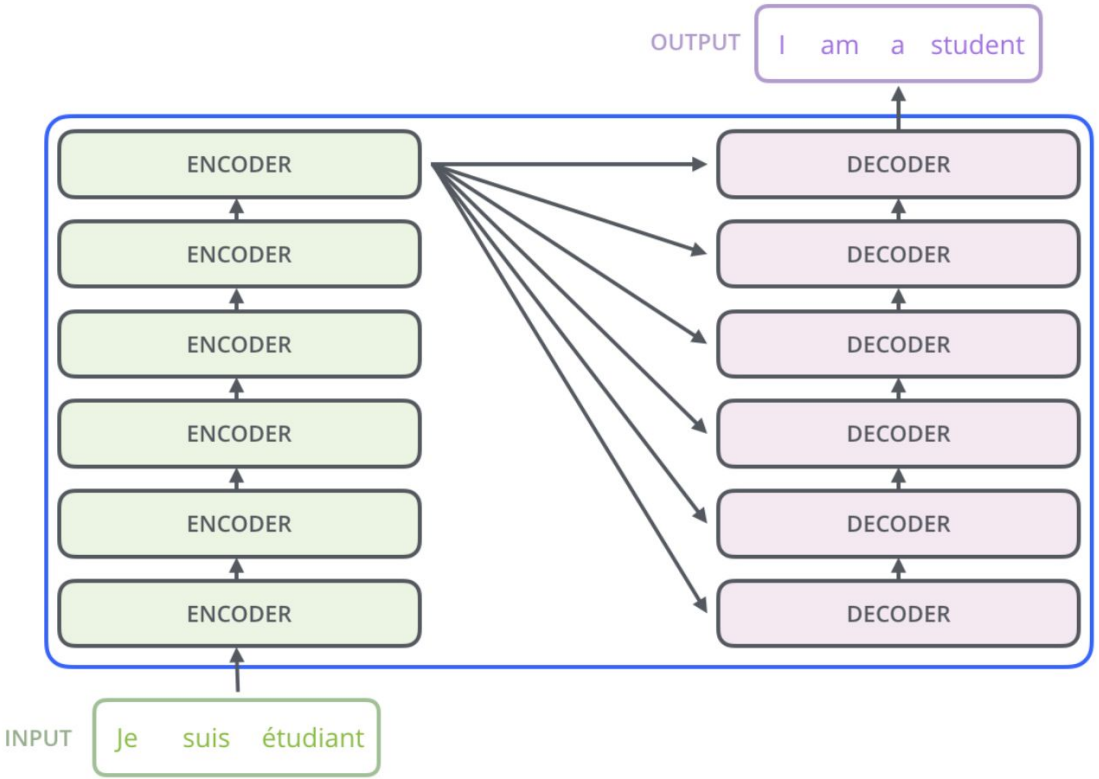
100%


OUTPUT
I am a student
ENCODERS
DECODERS
INPUT
Je suis étudiant


OUTPUT
I am a student
ENCODERS
DECODERS
INPUT
Je suis étudiant

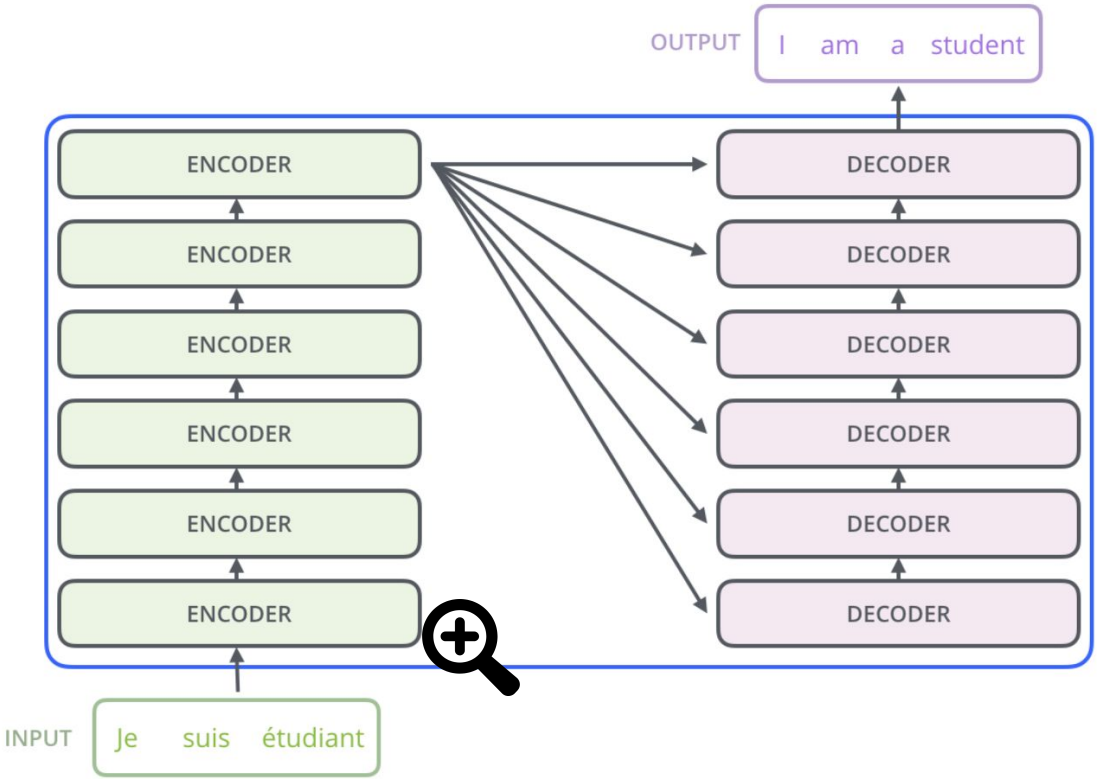
One step closer:

300%



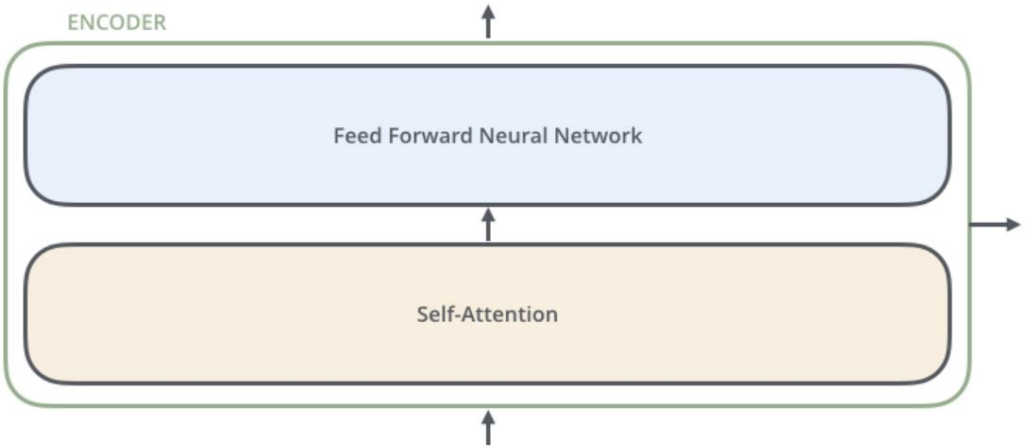
One step closer:

300%

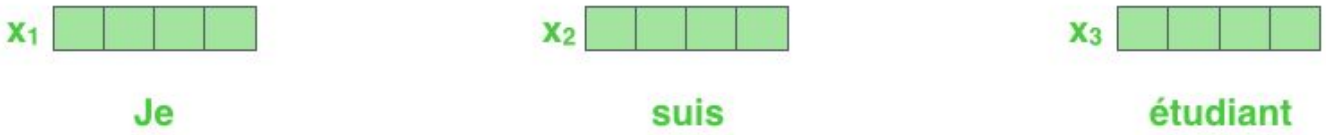


Encoder block

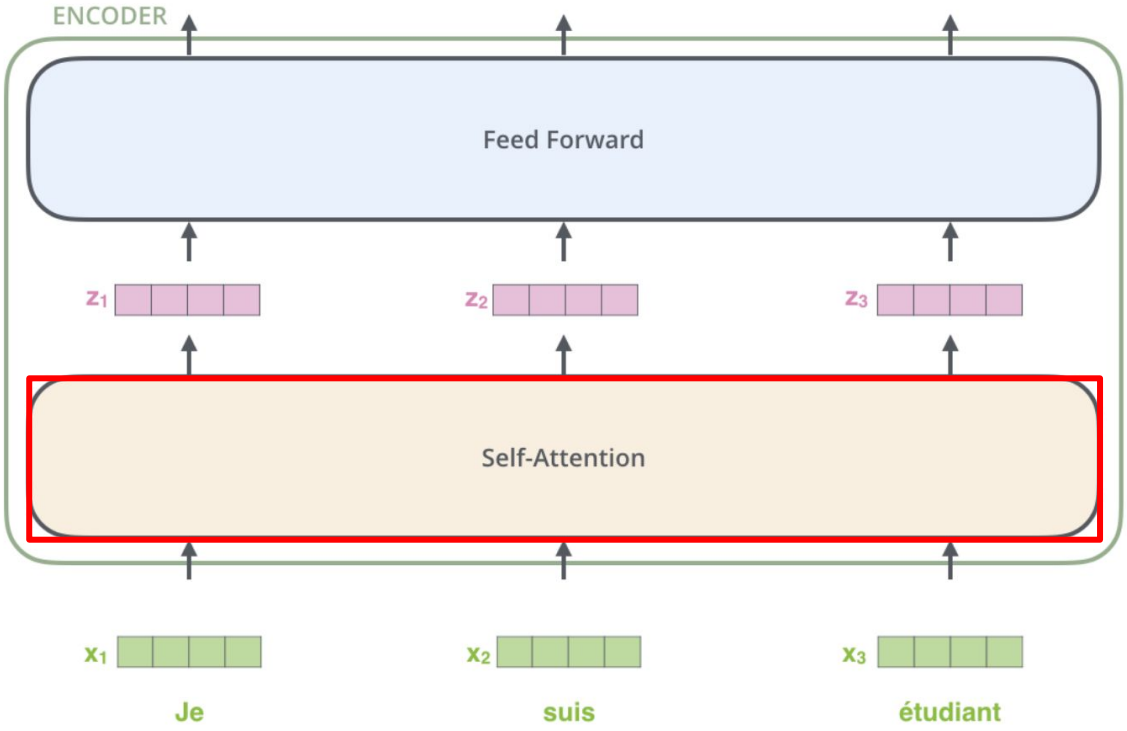
- Encoder block = Self-attention + Feed Forward NN



Step 1: Input tokens → word embeddings

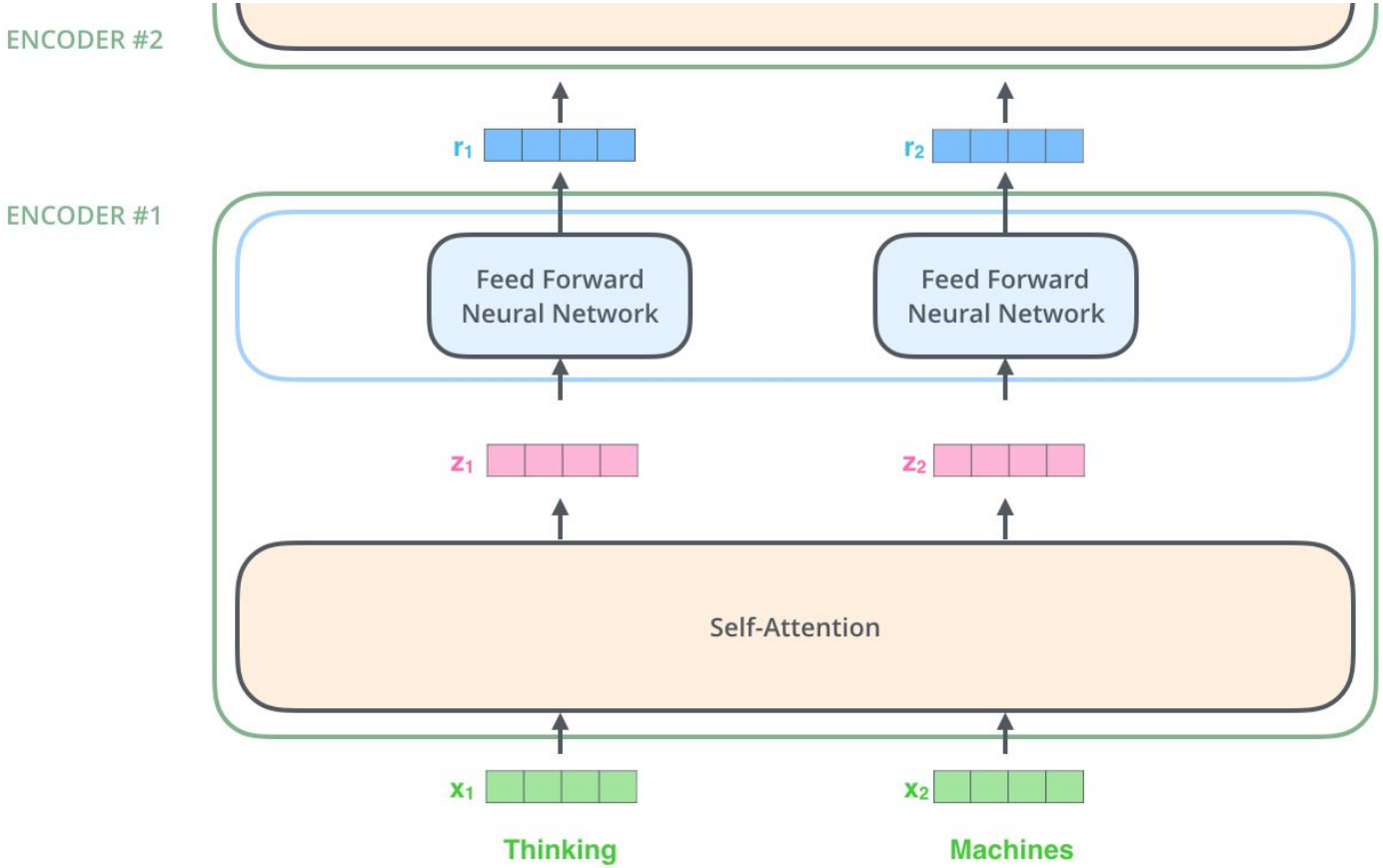


Step 2: Word embeddings → attention vectors

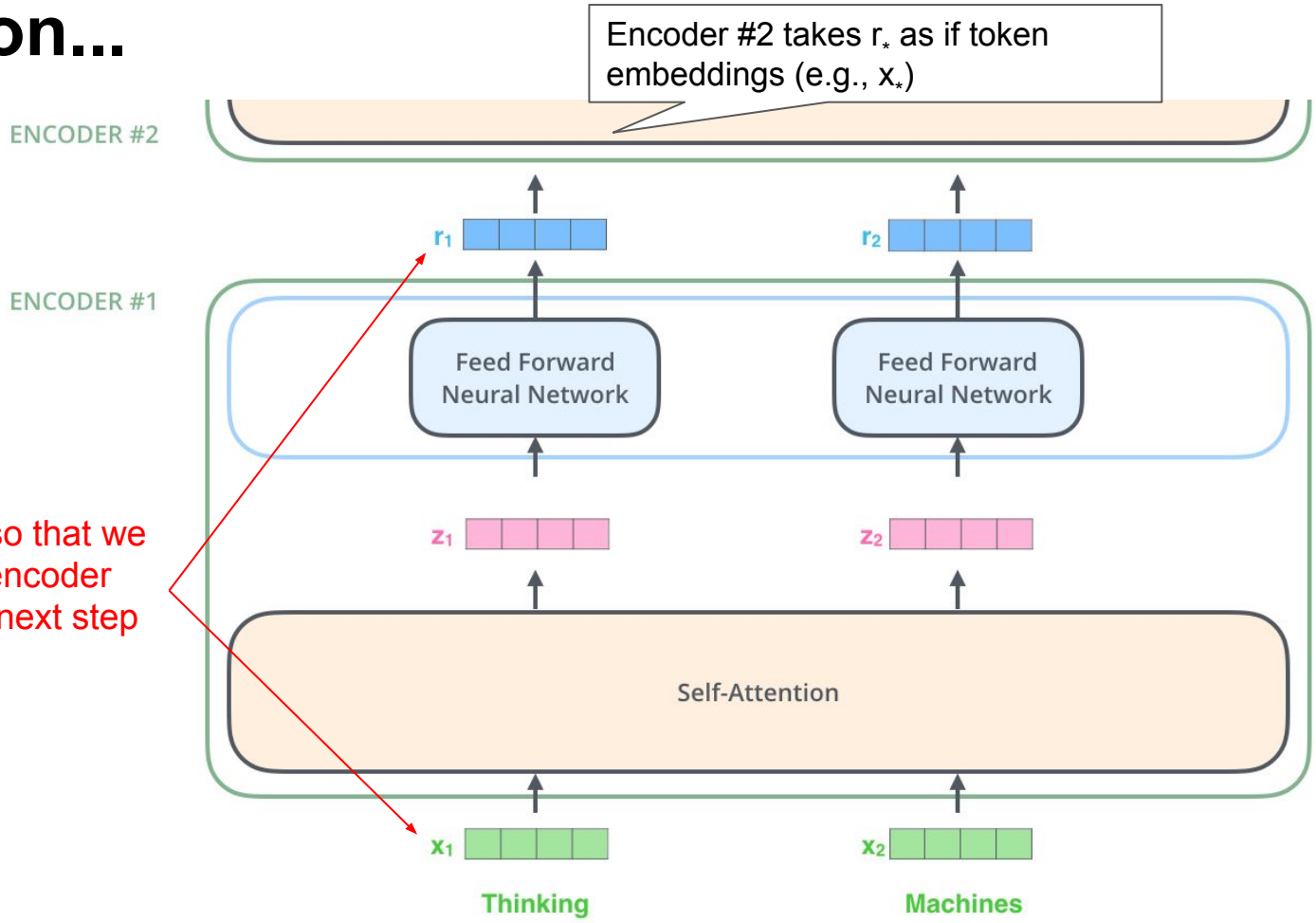


← attention vectors

Step 3: Attention vectors → output representations



And so on...

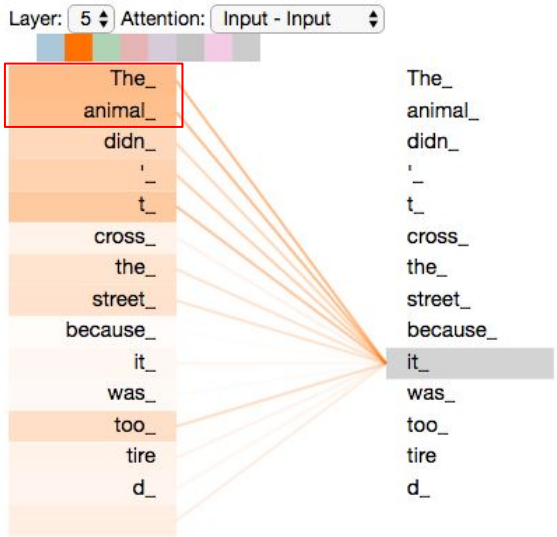


Same dimensions so that we can use the same encoder architecture as the next step

Let’s dive into Self-Attention!

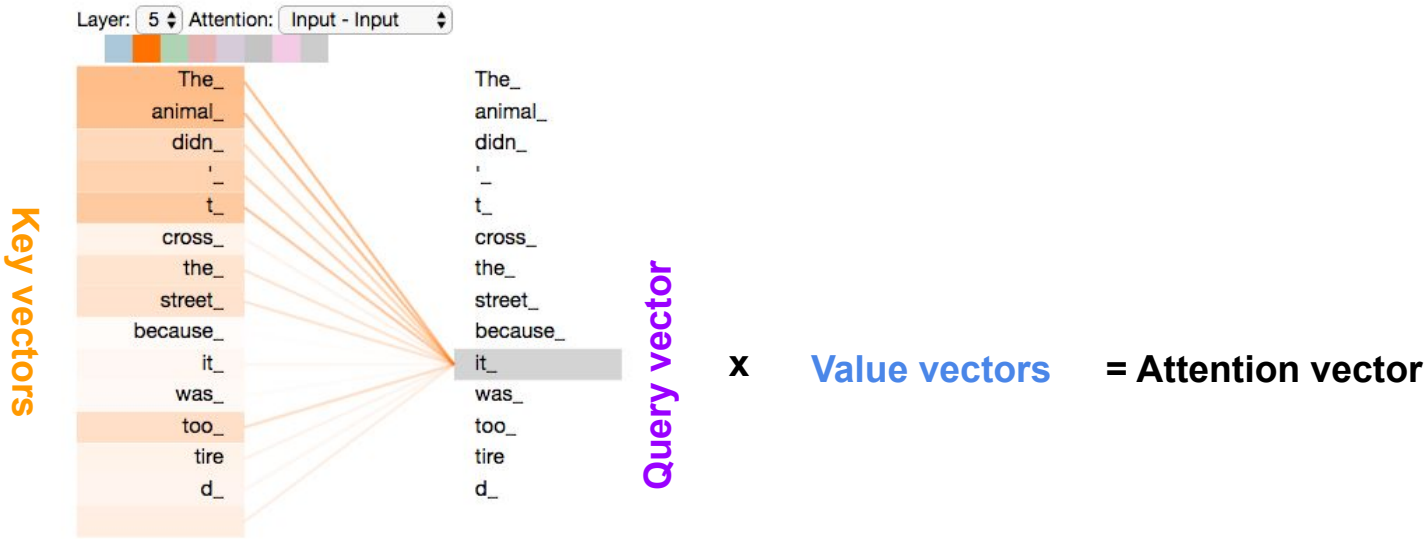
Self-Attention at a High Level

- Self-attention looks at **other positions** in the input sequence for **clues** that can help lead to a better encoding for this token
  - Example: “The animal didn’t cross the street because **it** was too tired.”



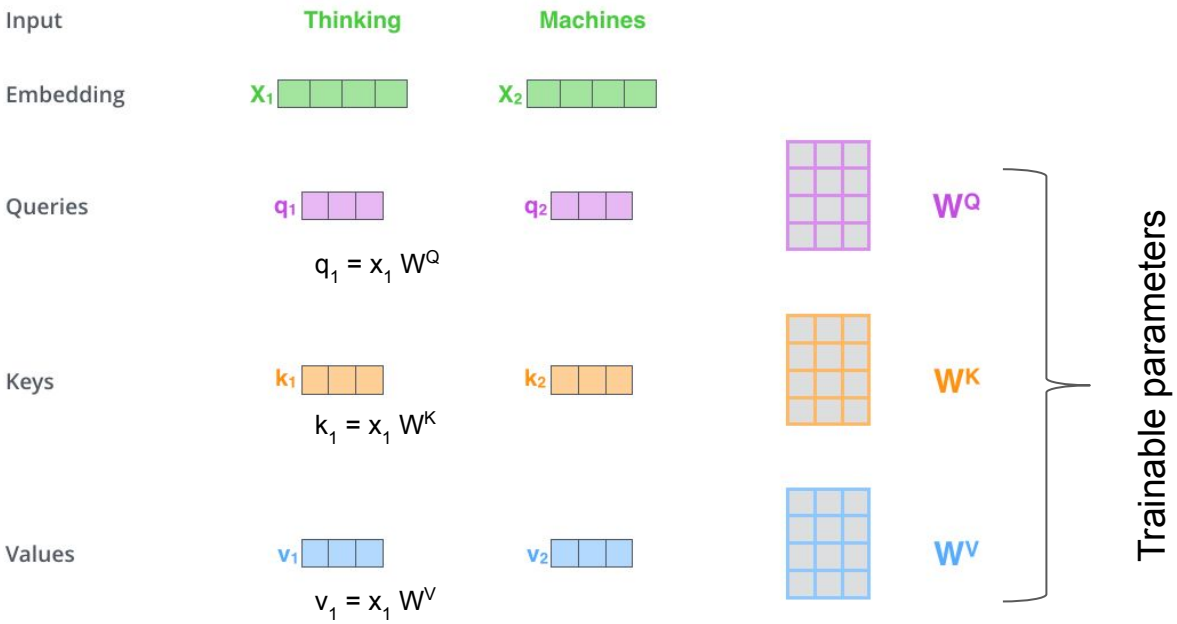
Self-Attention in Detail

- Self-Attention encodes a word embedding by averaging **Value vectors** of input tokens based on the similarity between **Key vectors** of input tokens and **Query vector** of the target token

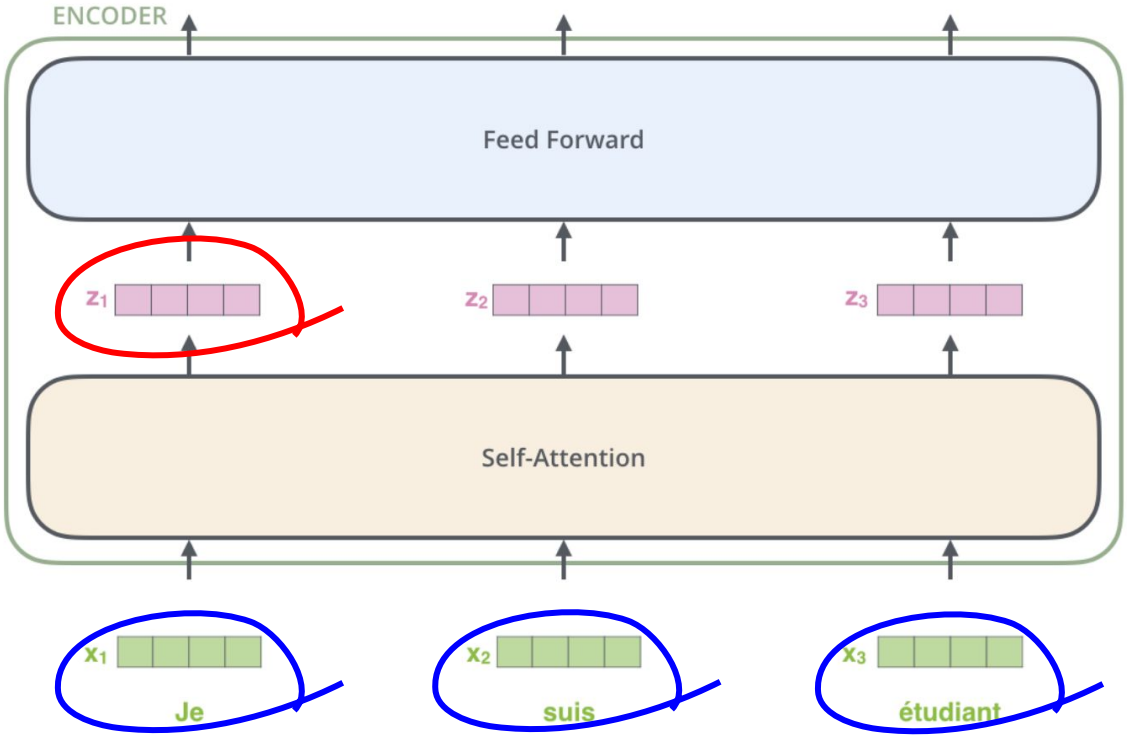


Word embeddings → Query, Key, Value vectors

- Self-attention layer has linear-conversion matrixes for query, key, and value vectors



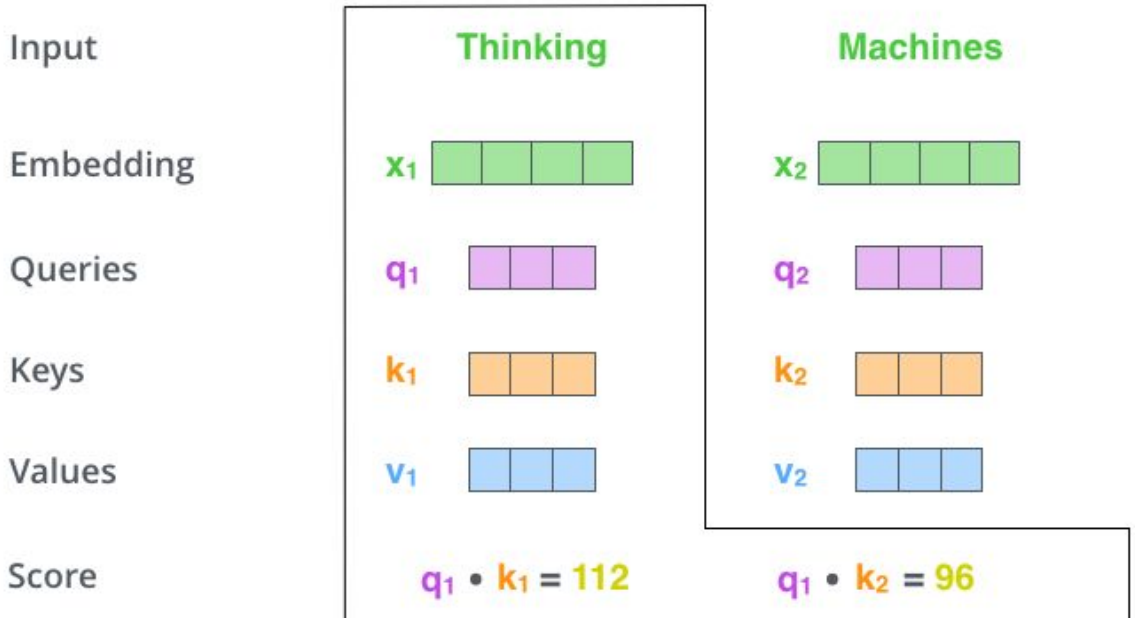
Calculating z_1 from input embeddings x_{1-3}



Calculating attention vector for “Thinking”

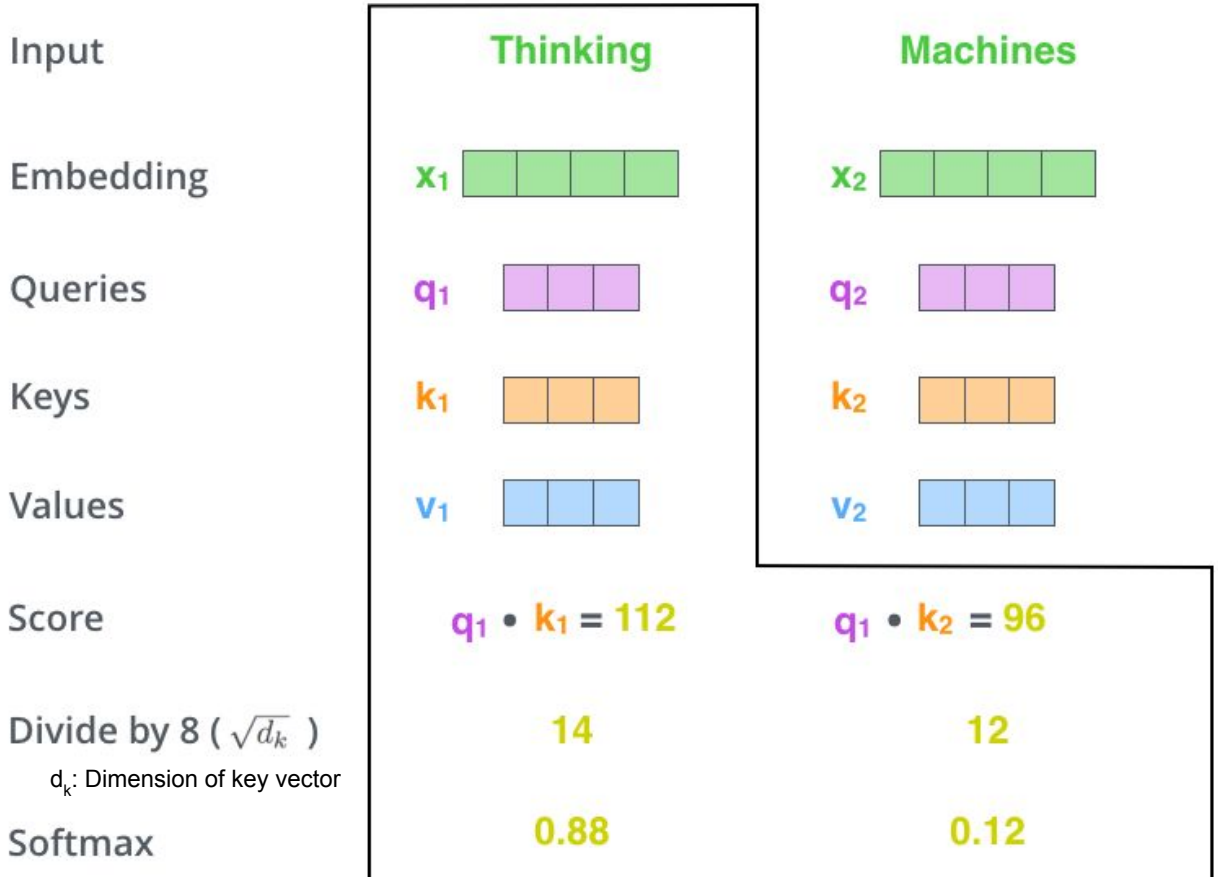
Step 1: Query-Key similarity calculation

- The inner product of q_1 and k_*



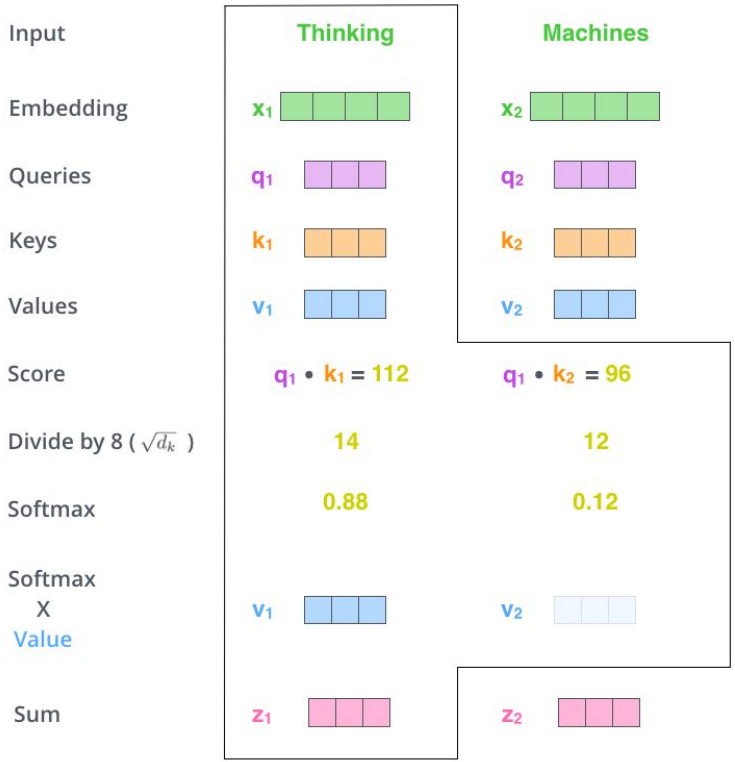
Calculating attention vector for “Thinking”

Step 2: Scaling and normalization



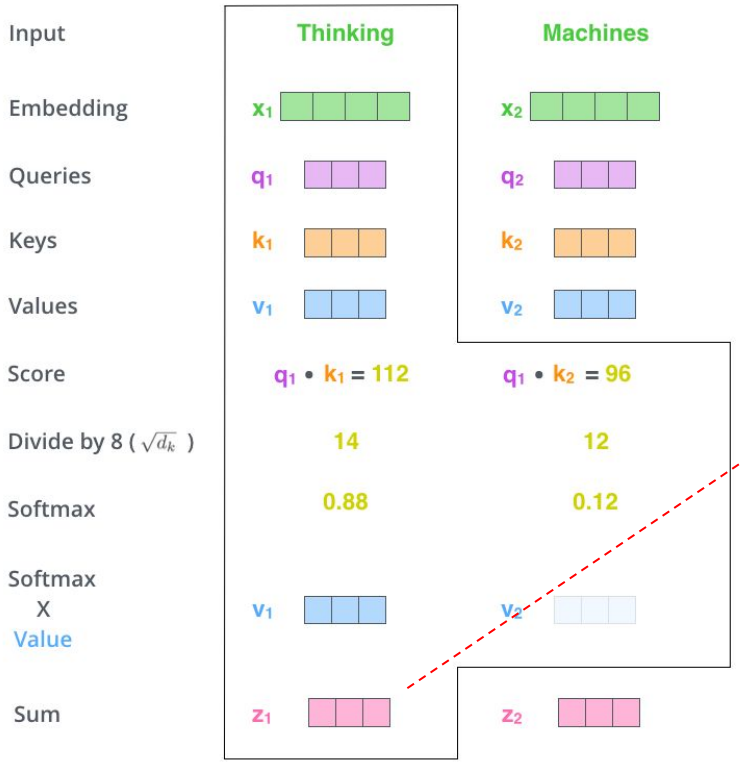
Calculating attention vector for “Thinking”

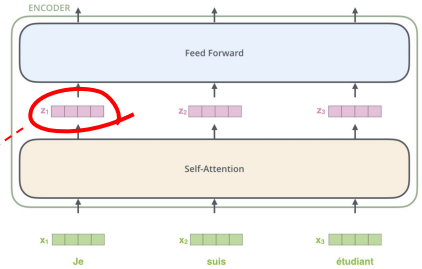
Step 3: Weighted average of Value vectors



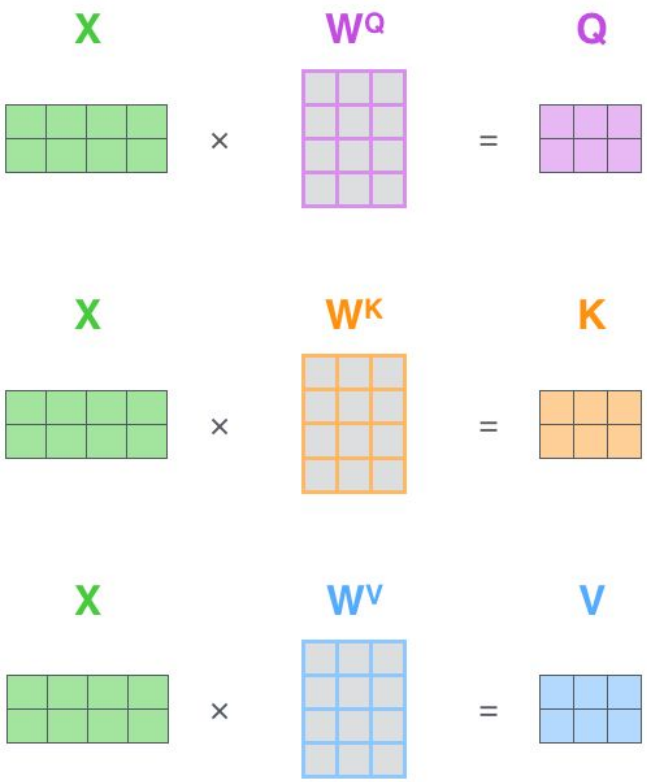
Calculating attention vector for “Thinking”

Step 3: Weighted average of Value vectors





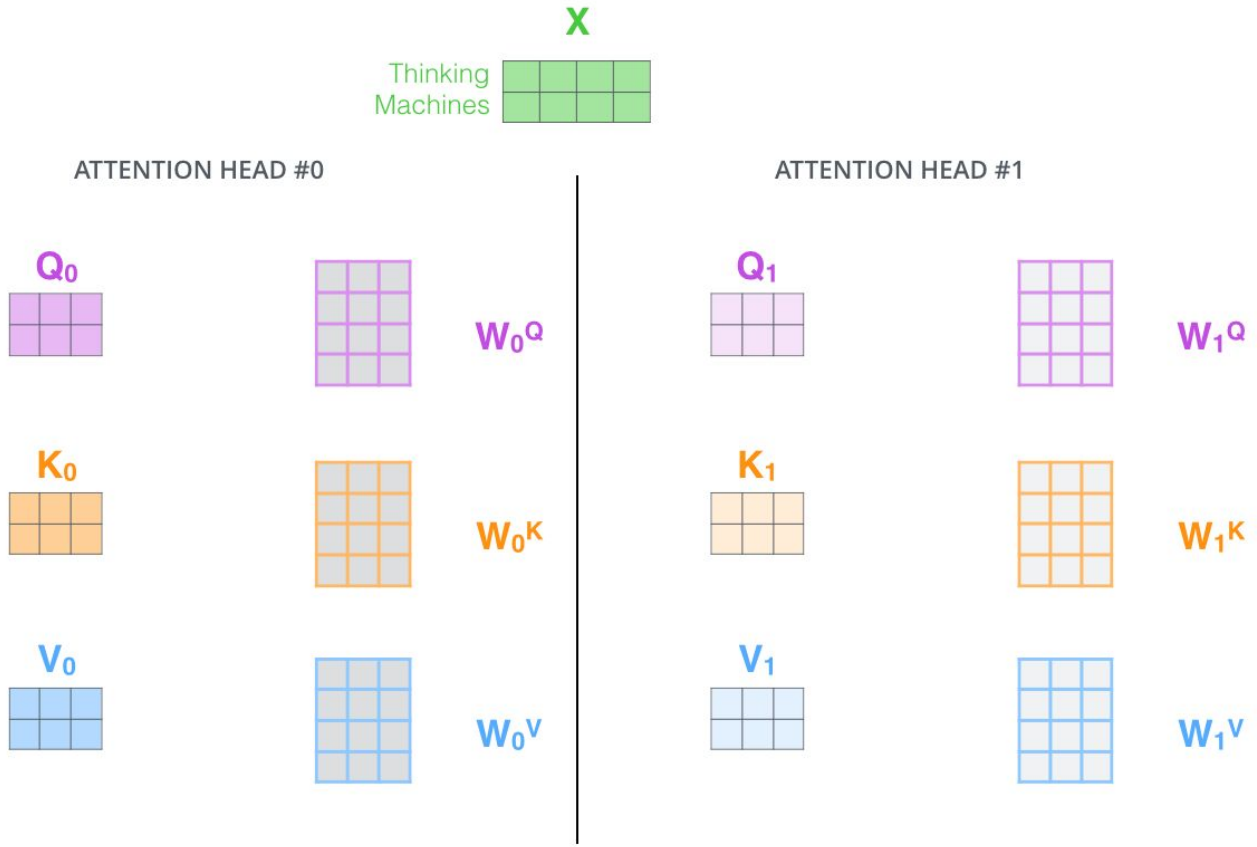
Understanding Self-Attention in Matrix Calculation



$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

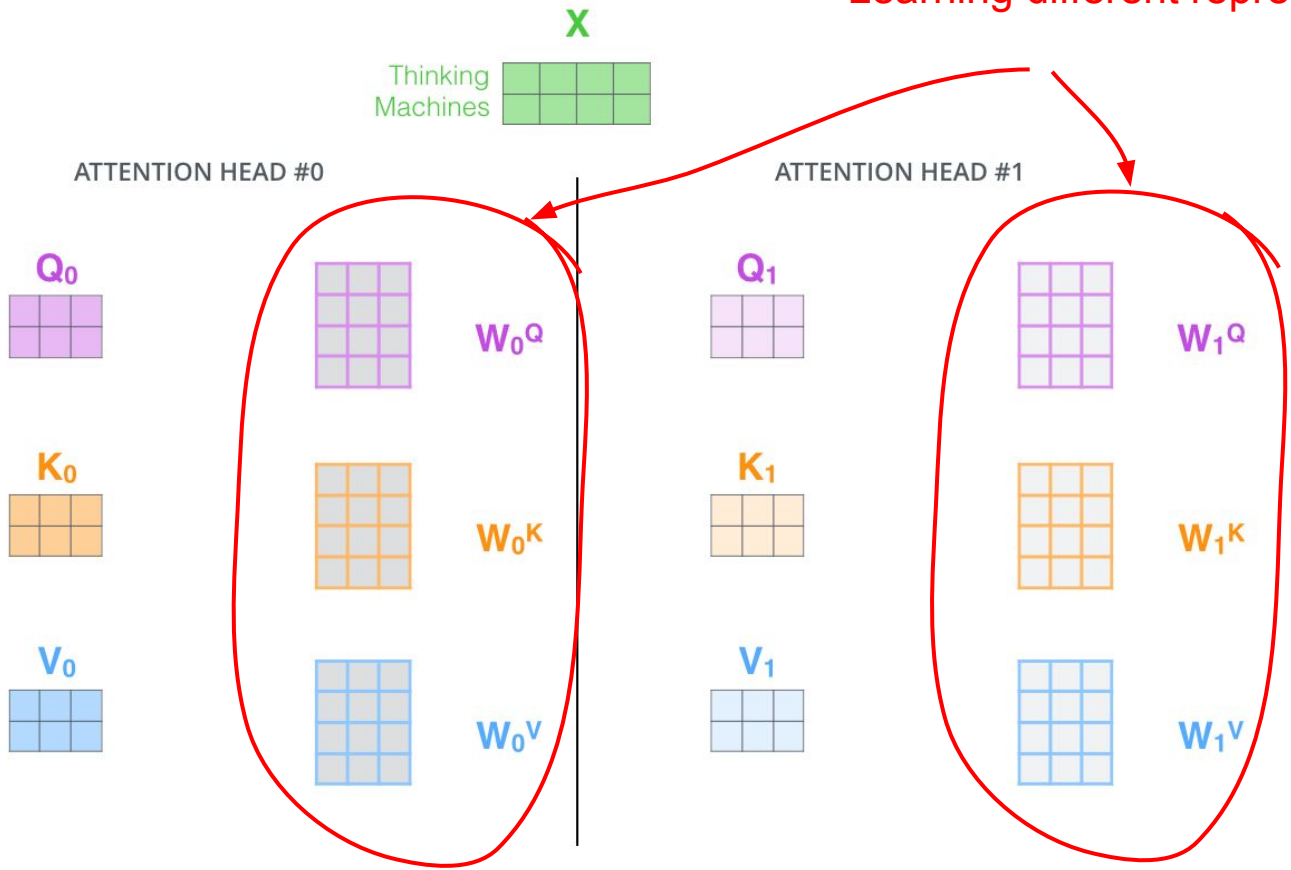
Multi-head Attention!

2-head Self-Attention

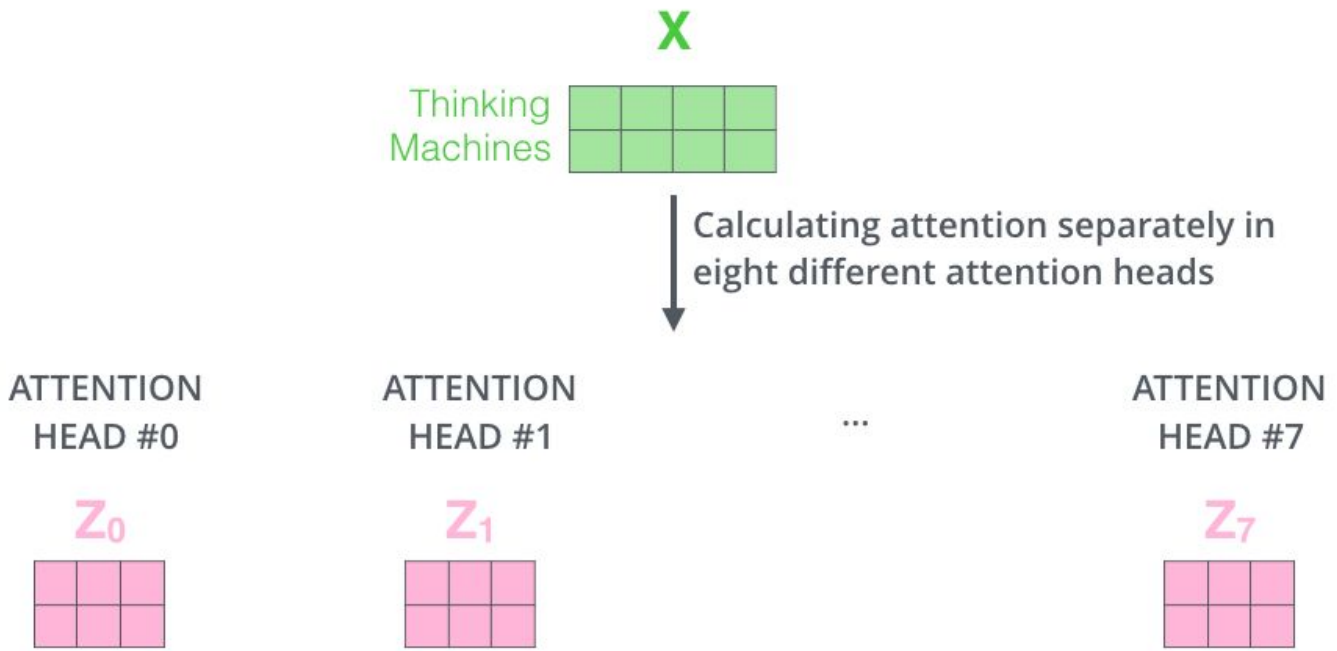


2-head Self-Attention

[Important!] Different parameters
= Learning different representations

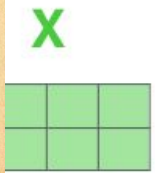


8-head Self-Attention = 8 encoded vectors



8 encoded vectors

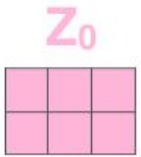
Yamata no orochi (やまたのおろち 八岐大蛇)
A legendary 8-head 8-tail Japanese dragon
https://en.wikipedia.org/wiki/Yamata_no_Orochi

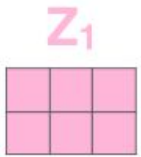


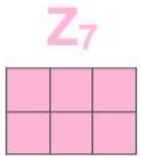
Calculating attention separately in eight different attention heads

...

ATTENTION HEAD #7

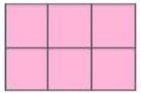


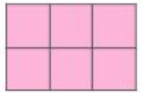


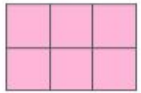


Yamata no orochi (やまたのおろち 八岐大蛇)
A legendary 8-head 8-tail Japanese dragon
https://en.wikipedia.org/wiki/Yamata_no_Orochi

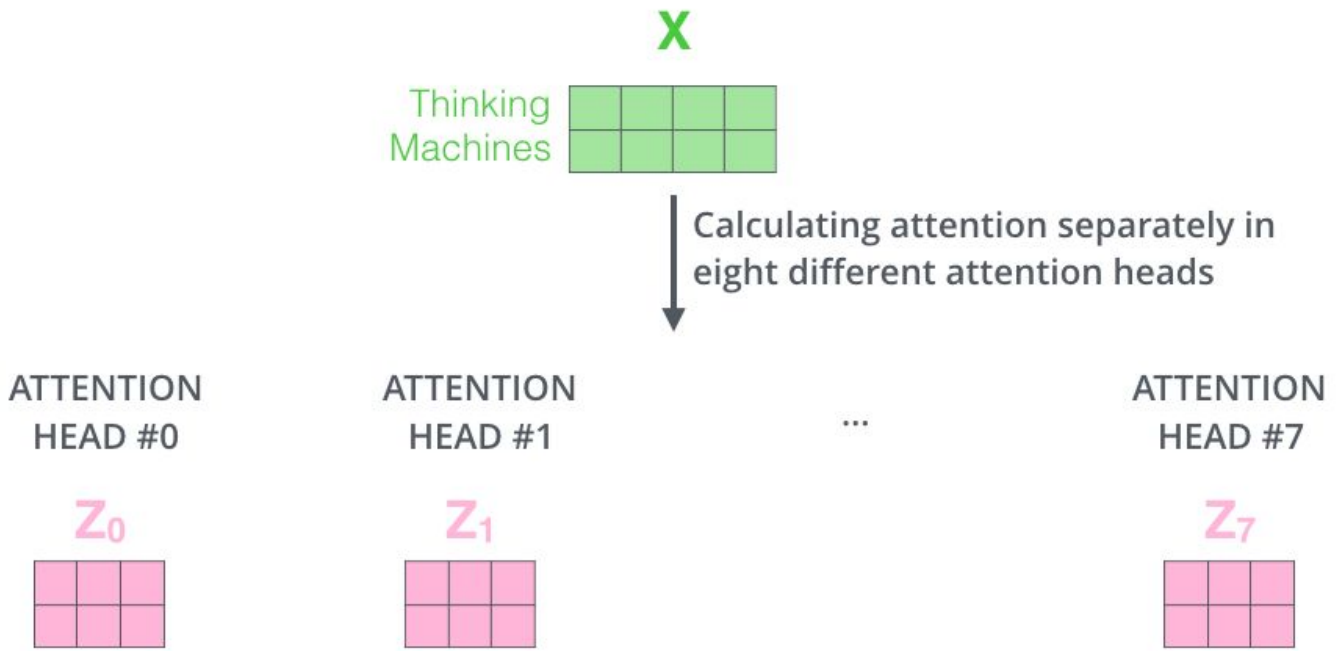
cf. Hydra (Greek myth)
Multi-head serpent
https://en.wikipedia.org/wiki/Lernaean_Hydra







8-head Self-Attention = 8 encoded vectors



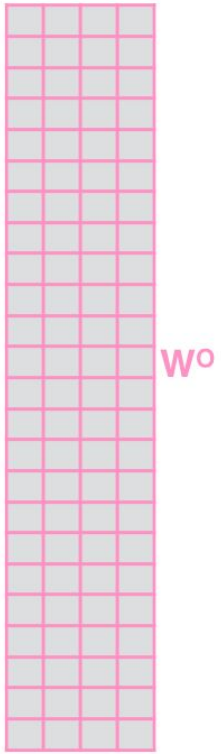
Q. How do we aggregate them to get a single z_1?

Concatenation & Linear-transformation

1) Concatenate all the attention heads

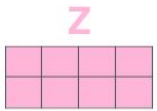
2) Multiply with a weight matrix W^O that was trained jointly with the model

X

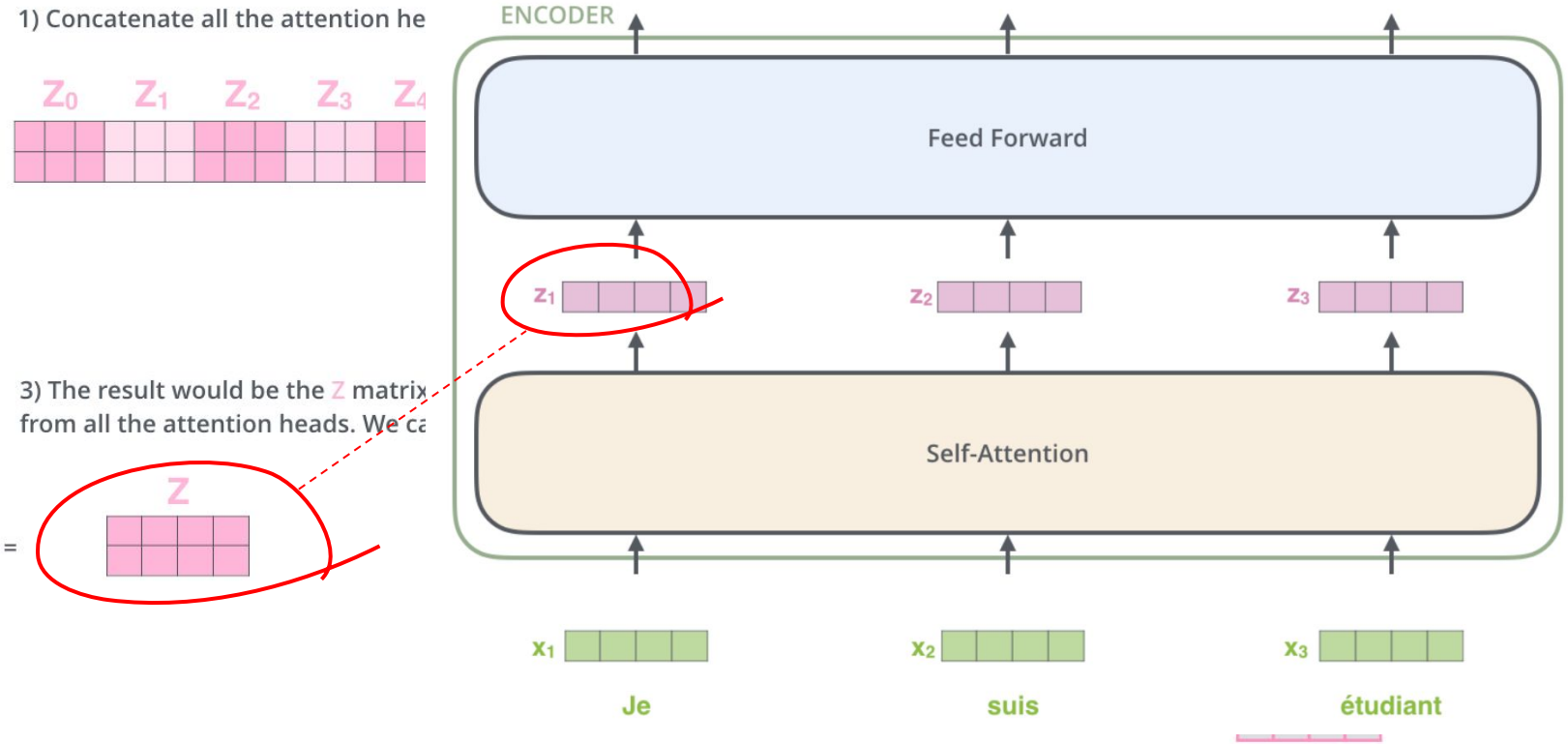


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

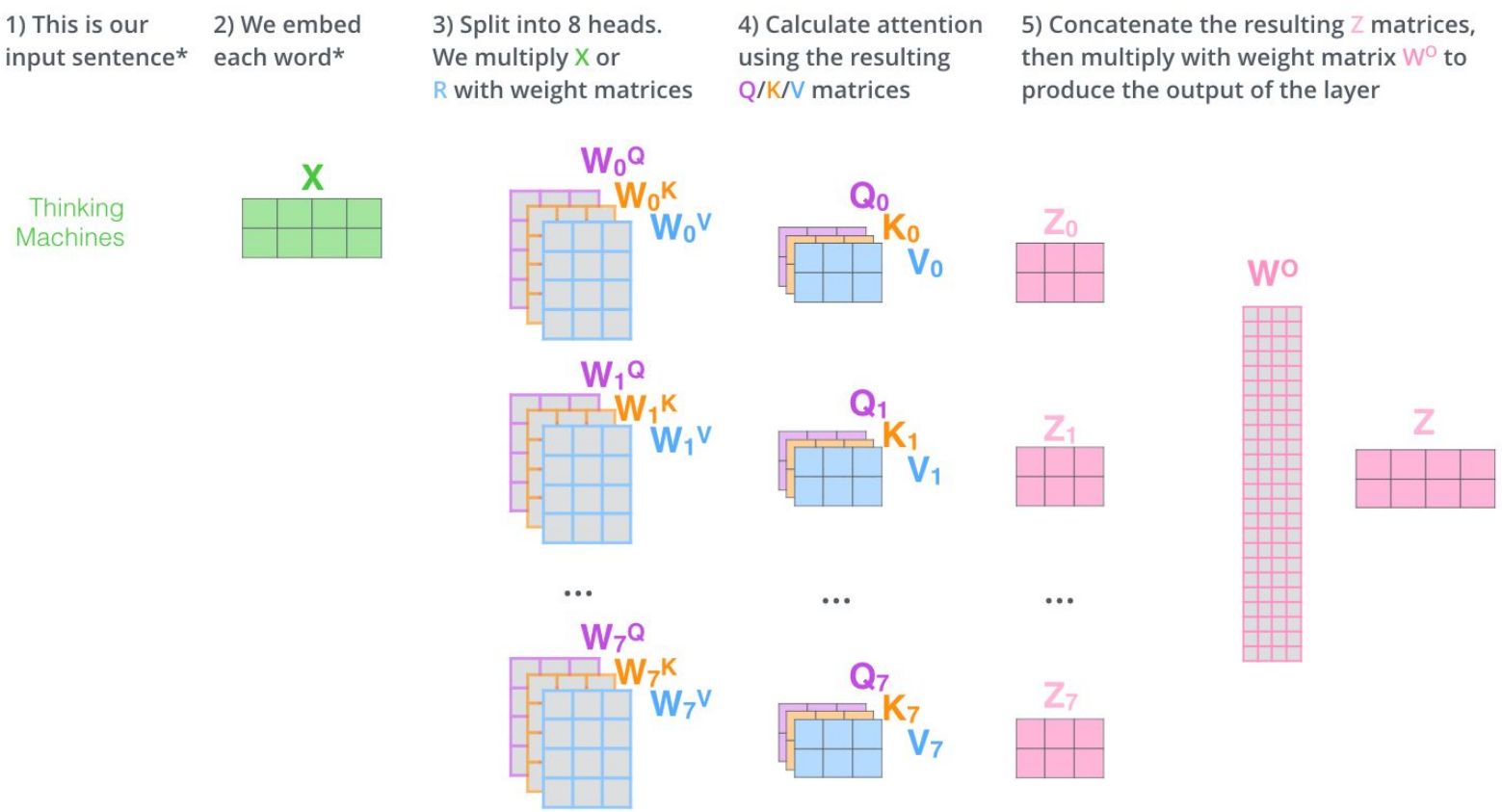
=



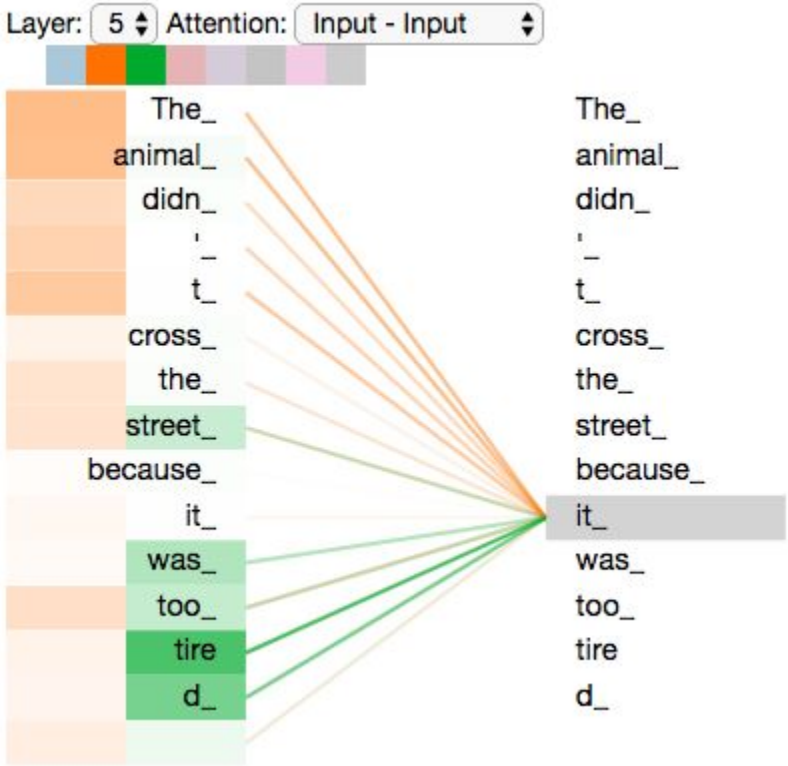
Concatenation & Linear-transformation



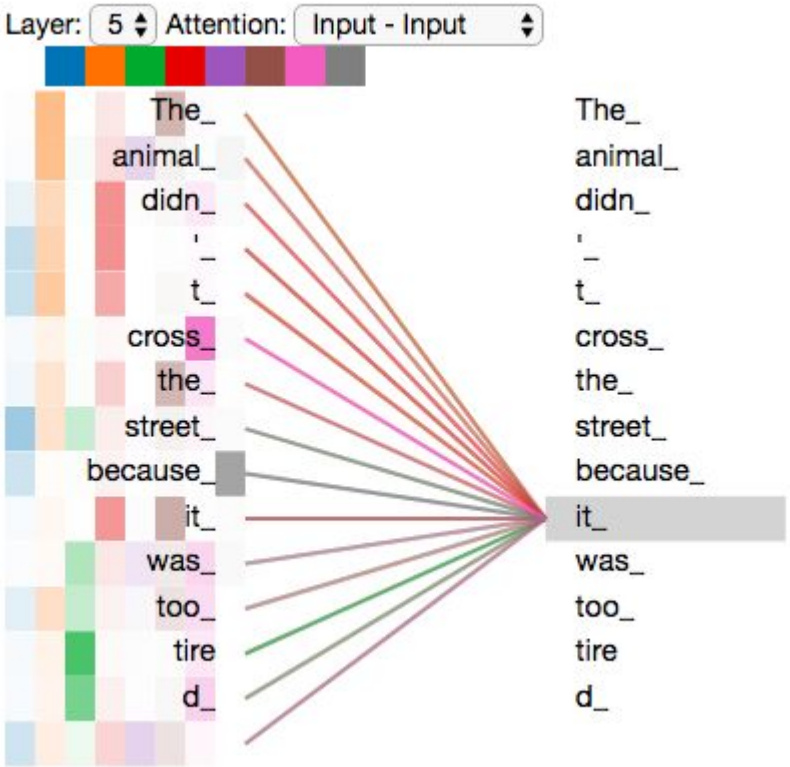
Recap: How Multi-head Self-Attention Works



Example: Different heads focus on different parts

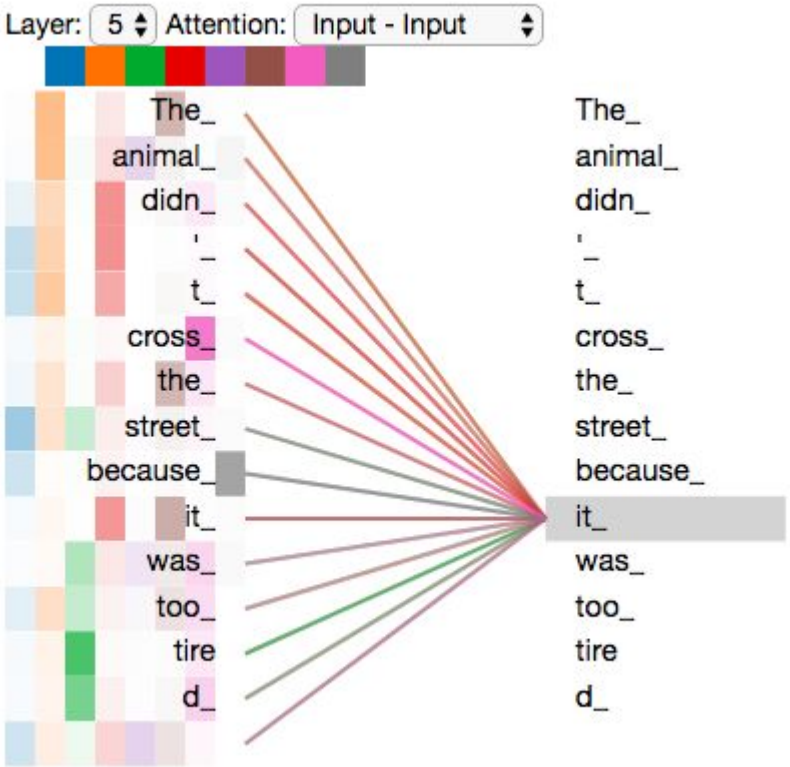


With 8 attention heads: Hard to interpret



???

With 8 attention heads: Hard to interpret



???

Q. Do we really need so many attention heads?

Q. Do we really need so many attention heads?

- Short answer is **No**

ACL ‘19

Analyzing Multi-Head Self-Attention:
Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned

Elena Voita[1,2] **David Talbot**[1] **Fedor Moiseev**[1,5] **Rico Sennrich**[3,4] **Ivan Titov**[3,2]

[1]Yandex, Russia [2]University of Amsterdam, Netherlands
[3]University of Edinburgh, Scotland [4]University of Zurich, Switzerland
[5]Moscow Institute of Physics and Technology, Russia
{lena-voita, talbot, femoiseev}@yandex-team.ru
rico.sennrich@ed.ac.uk ititov@inf.ed.ac.uk

ArXiv May ‘19 (under review?)

Are Sixteen Heads Really Better than One?

Paul Michel
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA
pmichel1@cs.cmu.edu

Omer Levy
Facebook Artificial Intelligence Research
Seattle, WA
omerlevy@fb.com

Graham Neubig
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA
gneubig@cs.cmu.edu

BlackBoxNLP ‘19@ACL ‘19

What Does BERT Look At?
An Analysis of BERT’s Attention

Kevin Clark[†] **Urvashi Khandelwal**[†] **Omer Levy**[‡] **Christopher D. Manning**[†]
[†]Computer Science Department, Stanford University
[‡]Facebook AI Research
{kevclark,urvashik,manning}@cs.stanford.edu
omerlevy@fb.com

the simple weighted average. In this paper we make the surprising observation that even if models have been trained using multiple heads, in practice, a large percentage of attention heads can be removed at test time without significantly impacting performance. In fact, some layers can even be reduced to a single head. We further examine greedy algorithms for pruning down models, and

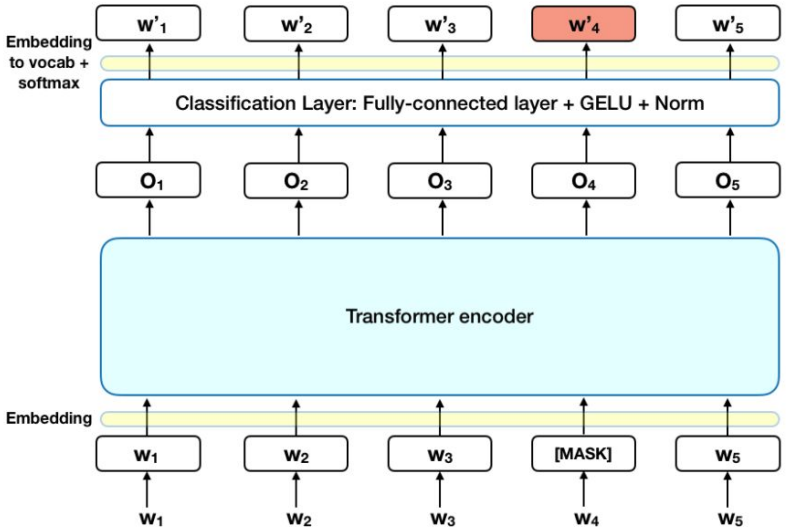
Let's go back to BERT

Pre-training a BERT model

- Task #1: Masked Language Model (LM)
- Task #2: Next Sentence Prediction

Pre-training Task #1: Masked Language Model

- Training a model so it can predict a randomly chosen **masked** word
  - 80%: Replace the word with [MASK]
  - 10%: Replace the word with a random word
  - 10%: Keep the word unchanged



Pre-training Task #2: Next Sentence Prediction

- Binary classification task

$$\text{Input} = \texttt{[CLS] the man went to [MASK] store [SEP]}$$
$$\texttt{he bought a gallon [MASK] milk [SEP]}$$
$$\text{Label} = \texttt{IsNext}$$

$$\text{Input} = \texttt{[CLS] the man [MASK] to the store [SEP]}$$
$$\texttt{penguin [MASK] are flight ##less birds [SEP]}$$
$$\text{Label} = \texttt{NotNext}$$

Pre-training Task #2: Next Sentence Prediction

- Binary classification task

$$\text{Input} = \texttt{[CLS] the man went to [MASK] store [SEP]}$$

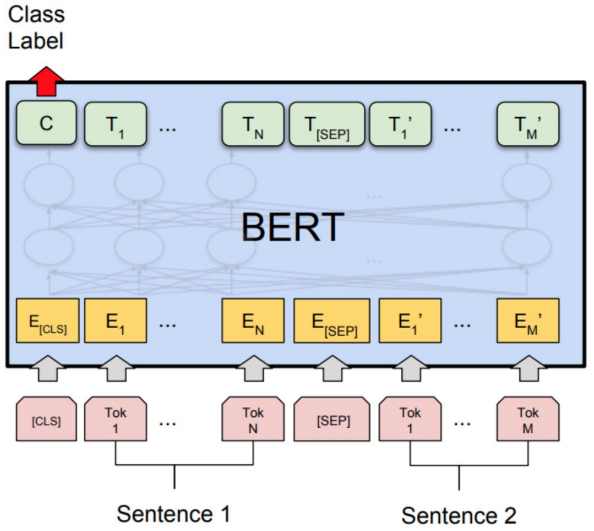
$$\texttt{he bought a gallon [MASK] milk [SEP]}$$

$$\text{Label} = \texttt{IsNext}$$

$$\text{Input} = \texttt{[CLS] the man [MASK] to the store [SEP]}$$

$$\texttt{penguin [MASK] are flight ##less birds [SEP]}$$

$$\text{Label} = \texttt{NotNext}$$



Massive Amount of Data!

- 3.3B words!!
  - 1.65M New York Times articles (2000 words/article)

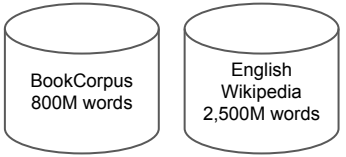
Pre-training data The pre-training procedure largely follows the existing literature on language model pre-training. For the pre-training corpus we use the BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). For Wikipedia we extract only the text passages and ignore lists, tables, and headers. It is critical to use a document-level corpus rather than a shuffled sentence-level corpus such as the Billion Word Benchmark (Chelba et al., 2013) in order to extract long contiguous sequences.

Two pre-trained BERT models

- **$\text{BERT}_{\text{BASE}}$**: L=12, H=768, A=12, Total Parameters=110M
- **$\text{BERT}_{\text{LARGE}}$**: L=24, H=1024, A=16, Total Parameters=340M

- L: # of layers (i.e., Transformer blocks)
- H: the hidden size
- A: # of self-attention heads

Two pre-trained BERT models



- **BERT$_{\text{BASE}}$**: L=12, H=768, A=12, Total Parameters=110M

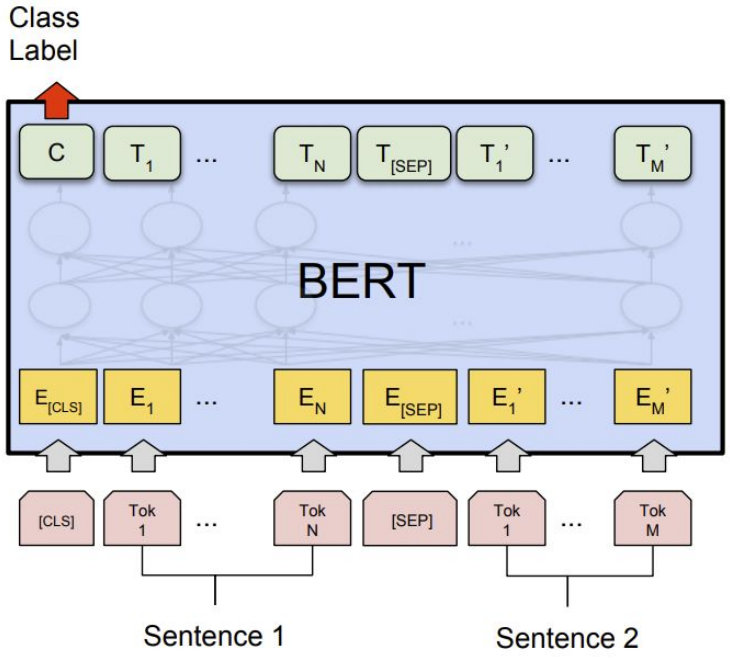
4 days with 16 TPU chips

- **BERT$_{\text{LARGE}}$**: L=24, H=1024, A=16, Total Parameters=340M

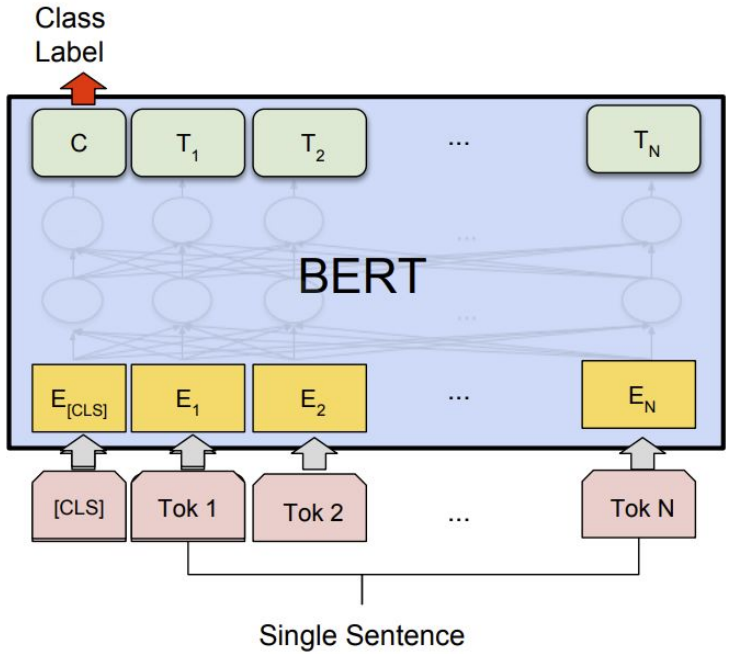
4 days with 64 TPU chips

- L: # of layers (i.e., Transformer blocks)
- H: the hidden size
- A: # of self-attention heads

BERT for different NLP tasks (1/2)

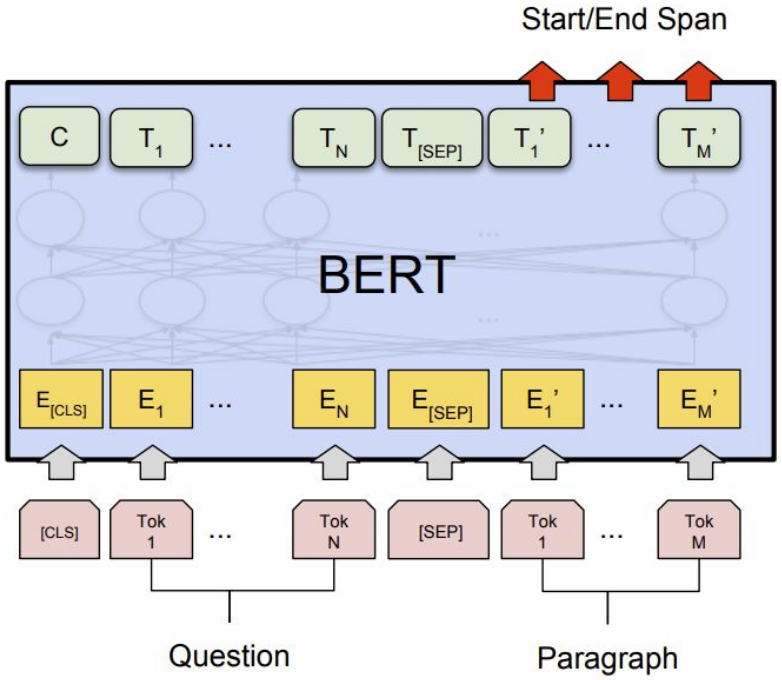


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

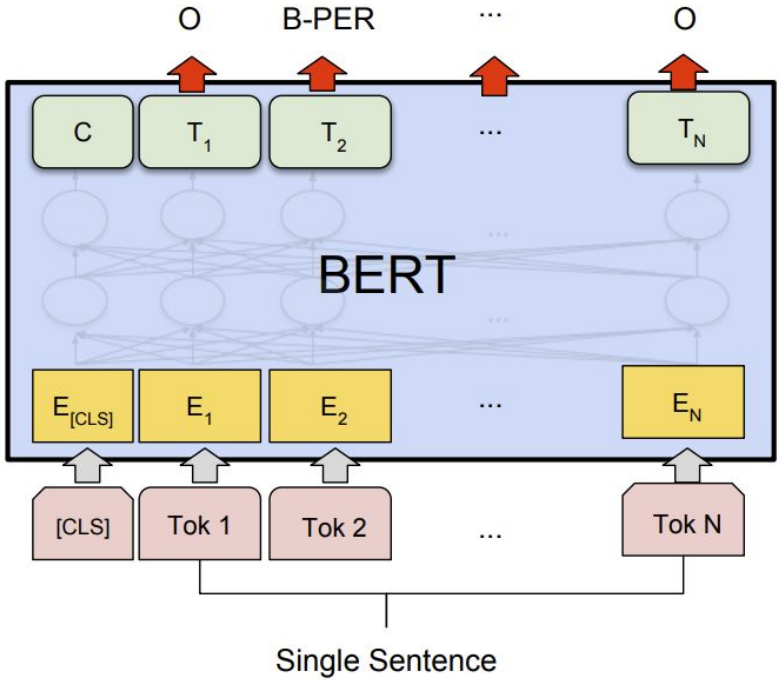


(b) Single Sentence Classification Tasks: SST-2, CoLA

BERT for different NLP tasks (2/2)



(c) Question Answering Tasks: SQuAD v1.1



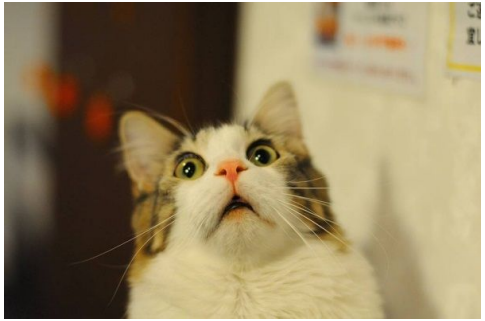
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

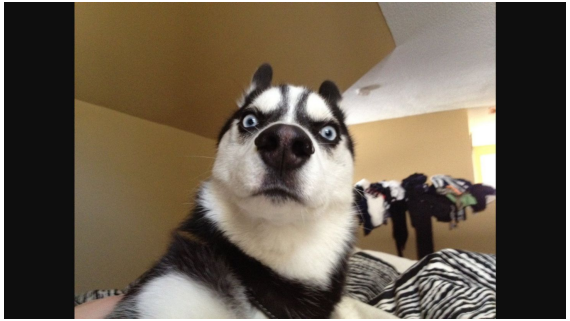
BERT performance

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**

BERT performance

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**

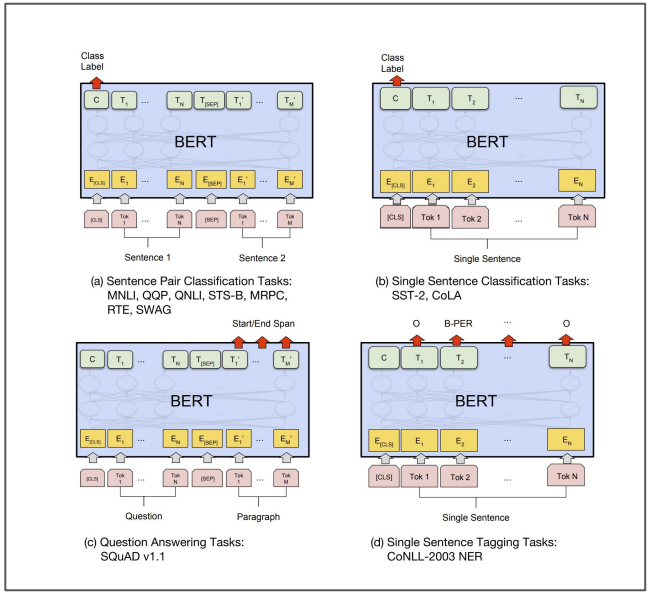




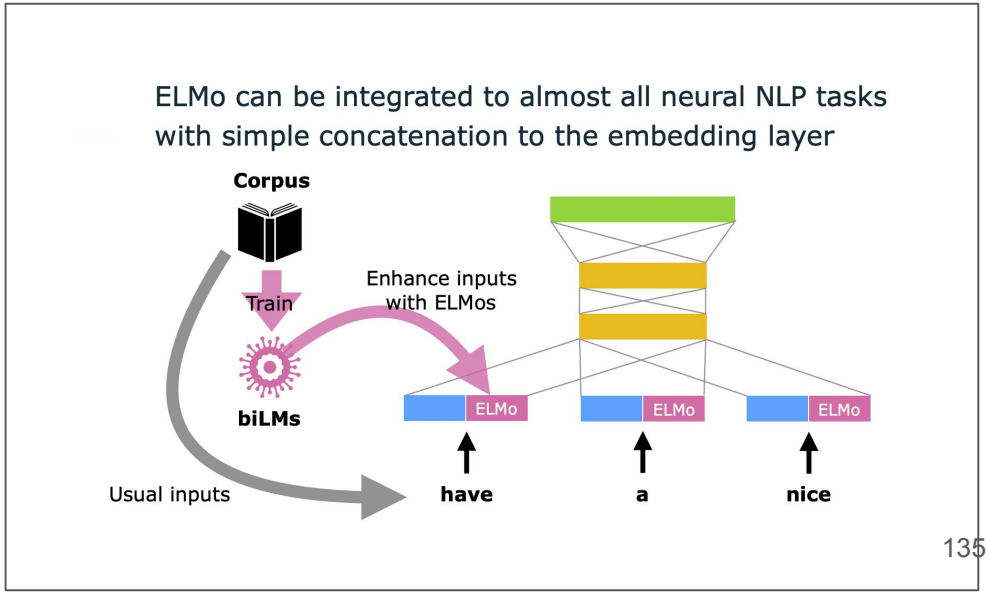
Fine-tuning approach

- We do not change the network architecture of BERT, but simply re-train (i.e., **fine-tuning**) the model based on the target task

Fine-tuning approach (e.g., BERT)



Feature-based approach (e.g., ELMo)



BERT performs well in the feature-based approach, tho

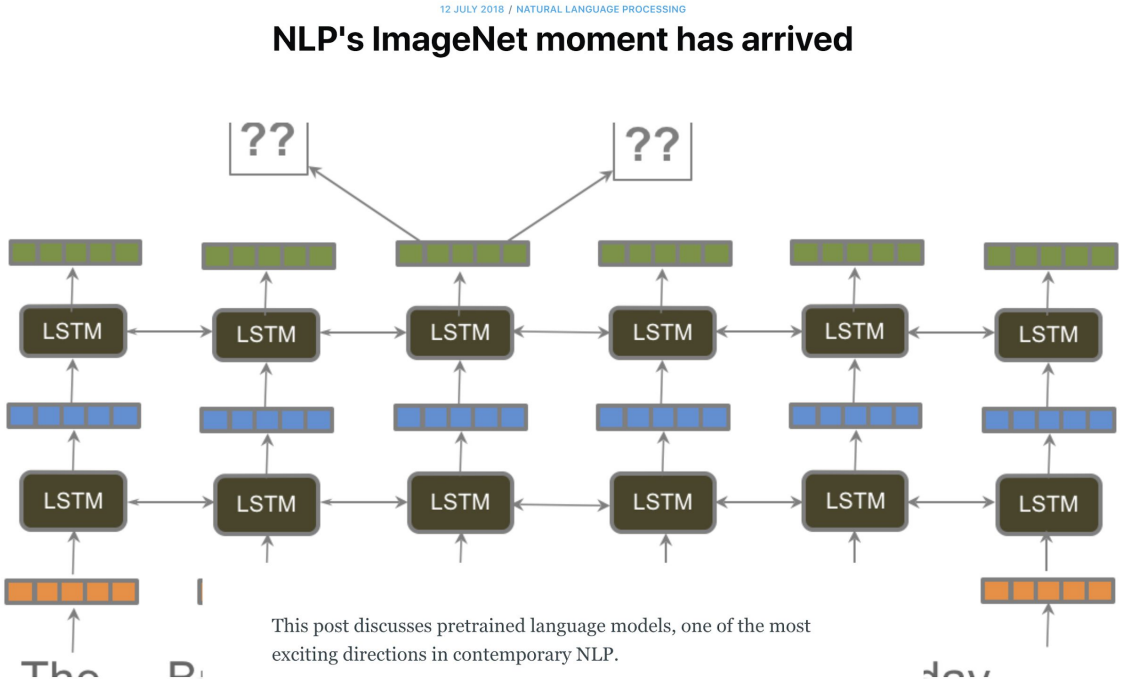
Layers	Dev F1
Finetune All	96.4
First Layer (Embeddings)	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Sum All 12 Layers	95.5

Table 7: Ablation using BERT with a feature-based approach on CoNLL-2003 NER. The activations from the specified layers are combined and fed into a two-layer BiLSTM, without backpropagation to BERT.

BiLSTM-CRF + ELMo 92.22 (test set)

NLP's ImageNet moment has arrived

- http://ruder.io/nlp-imagenet/



Today's Highlights

One-hot encoding

apple

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

orange

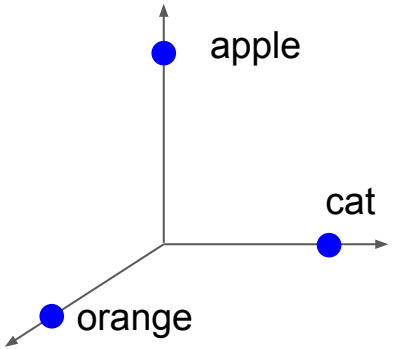
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

apple (orange) and orange (cat) are Equally Different

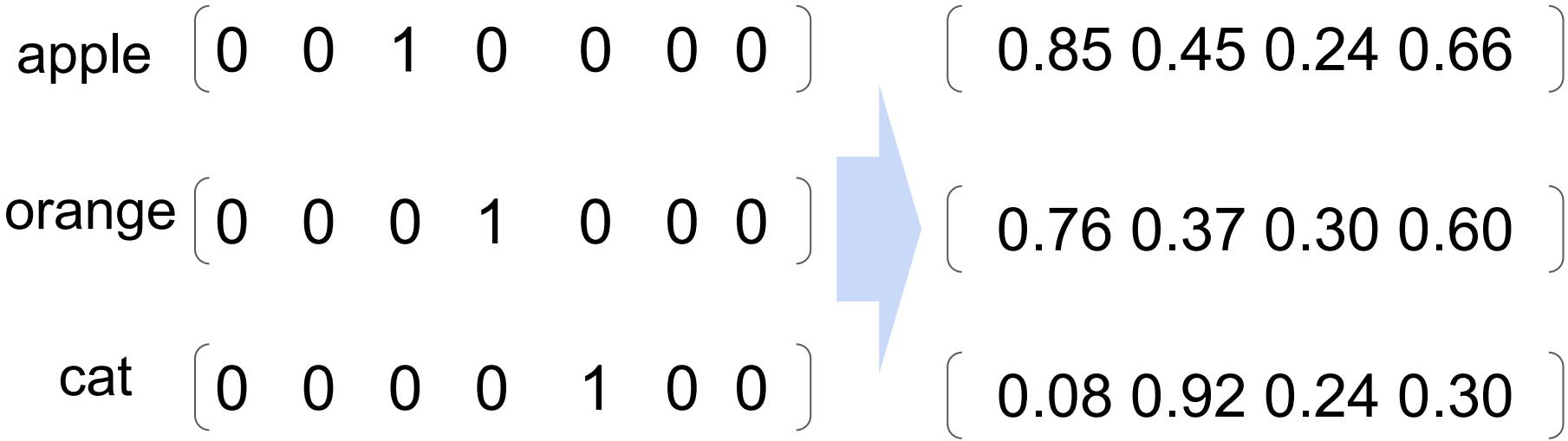
apple $\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$

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

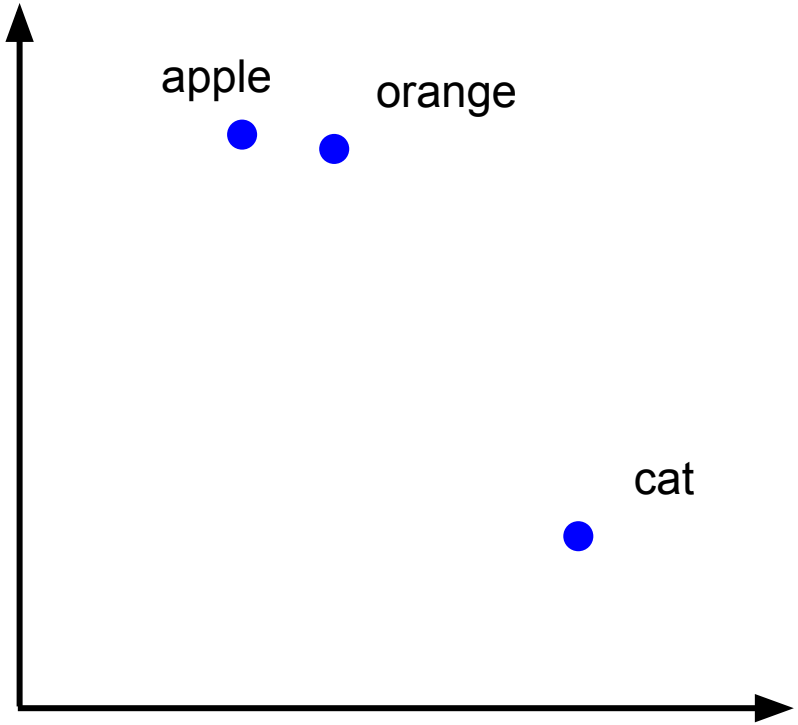
cat $\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$



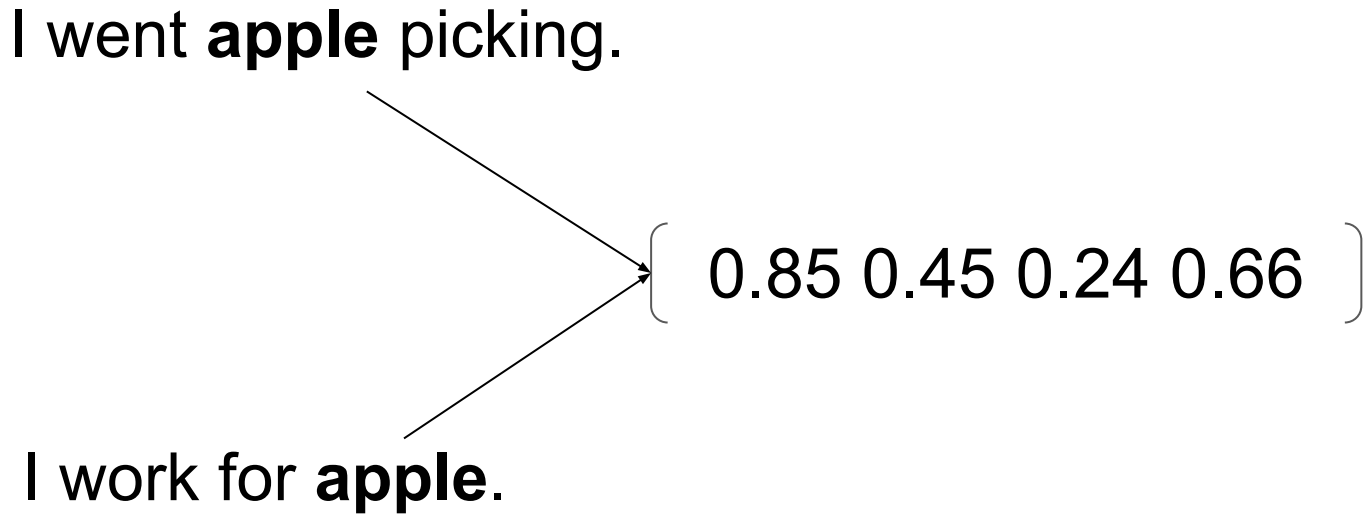
Make Dense Vector Representations



Dense Vectors = Word Embeddings



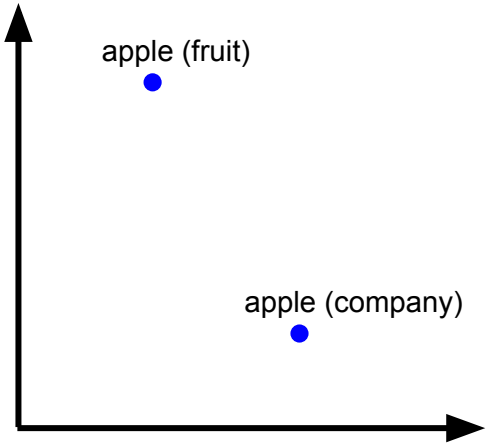
Word Embeddings are NOT enough



"Contextualized" Word Embeddings

I went **apple** picking.

I work for **apple**.

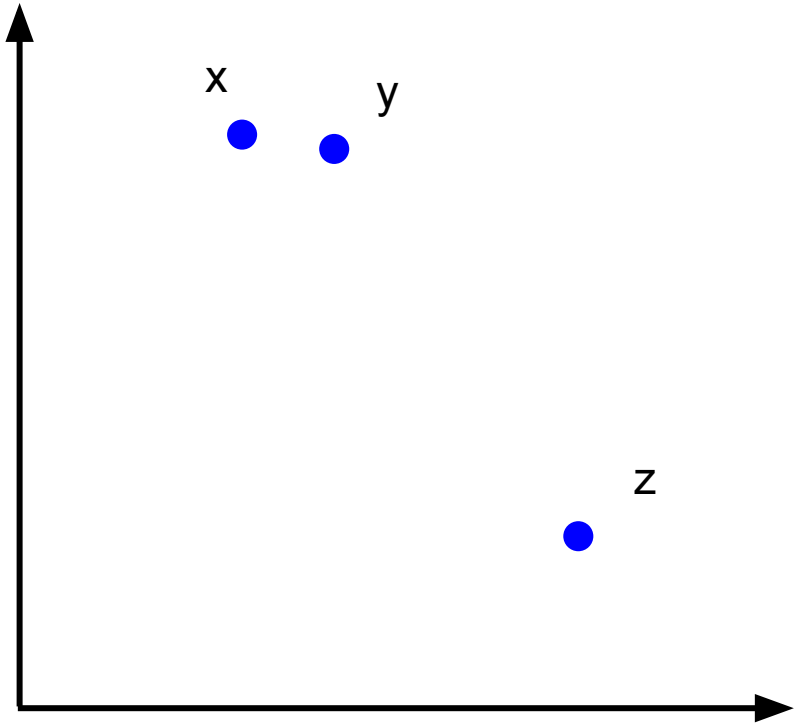


FYI: Ugly duckling theorem

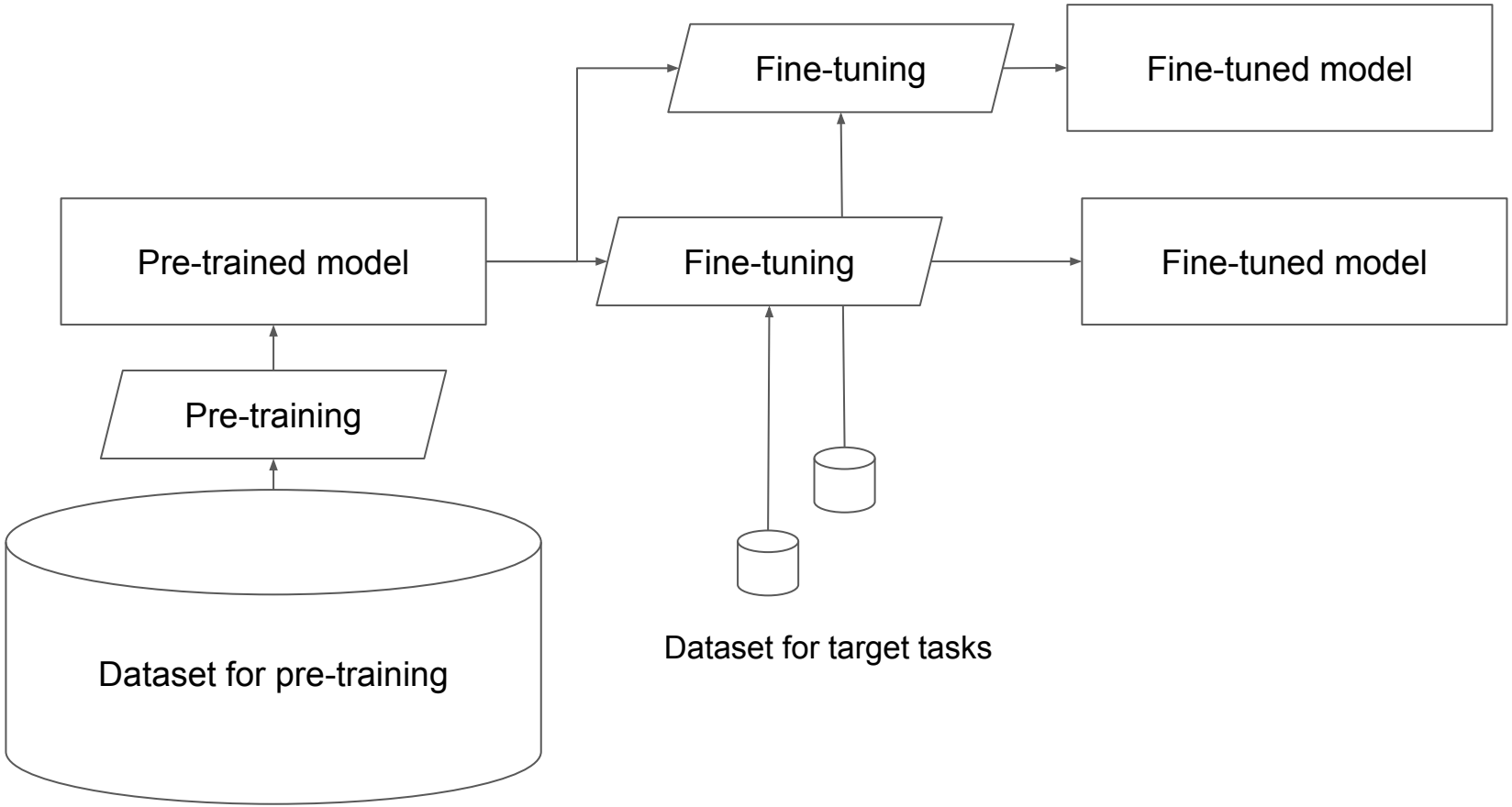
- Classification is not really possible without some sort of bias

	A	B	C						
1	1	0	0	F	$=$	$F\wedge$	W	first
2	1	1	0		W $=$	$F\vee$	W	white
3	0	0	0	0	$=$	$F\wedge\neg W$		first and non-white
4	0	0	1		$\neg W$ $=$	$\neg F\wedge\neg W$		non-white
5	0	1	0	$\neg F\wedge$	W $=$	$F\oplus$	W	non-first and white
6	0	1	1	$\neg F$	$=$	$\neg F\vee\neg W$		non-first
7	1	0	1	$F\vee\neg W$	$=$	$\neg F\oplus$	W	first or non-white
8	1	1	1	1	$=$	$\neg F\vee$	W	non-first or white

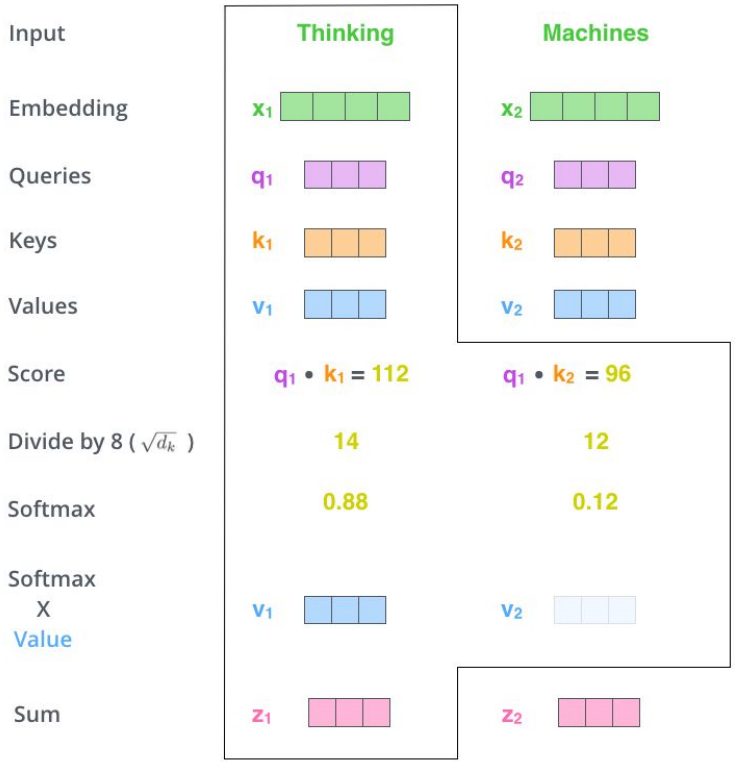
Google "<Something>2vec"



Key Concept: Pre-trained model + Fine-tuning



Key Concept: Self-Attention



Explain Transformer if > 1h

Break

Which Tool Should I Use?

- Depends!
- Knowing the differences would help you make a right decision :)

If all you have is a hammer, everything looks like a nail. (Maslow's hammer)

Who cares! Let's hit the nail! :)