

# **Applied Machine Learning for Business Analytics**

Lecture 8: Frontiers in NLP

# Agenda

1. Representation Learning in NLP
2. Word Embeddings
3. Neural Networks for NLP
4. Attention is all you Need
5. Introduction to BERT
6. From GPT to ChatGPT
7. Fine-tuning vs Prompt Engineering

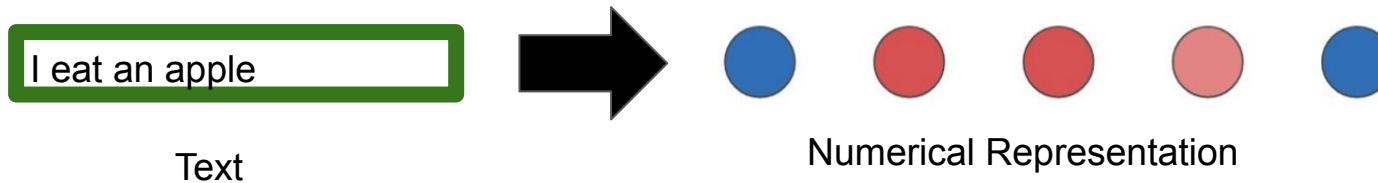
# 1. Representation Learning

# Representation learning

- We need to develop systems that read and understand text the way a person does, by forming a representation of the text, and other context information that humans create to understand a piece of text.

# Representation learning

- We need to develop systems that read and understand text the way a person does, by forming a representation of the text, and other context information that humans create to understand a piece of text.



The learned representation should capture high-level semantic and syntactic information.

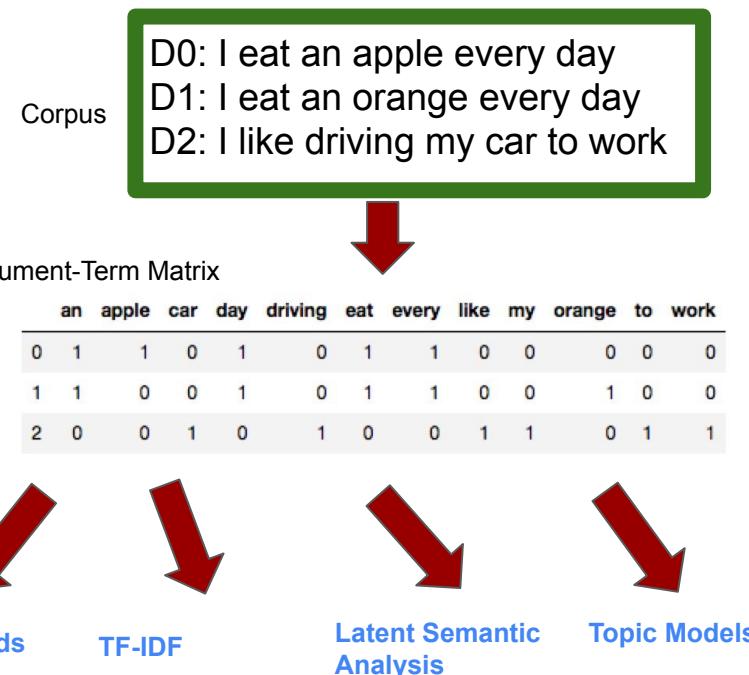
# History of NLP

- Now, neural nlp models are able to achieve state-of-arts results in all tasks.
- Before neural nlp:
  - Symbolic NLP: rule-based system (derived from linguistic)
  - Statistical NLP: data-driven and use statistical methods



# Statistical NLP

- Starting from Document-Term Matrix
  - It contains the co-occurrence information
  - Bag-of-Words: n-gram as features
  - TF-IDF: frequency of words to measure importance
  - Matrix Decomposition:
    - SVD->Latent Semantic Analysis
    - Probabilistic model-> Topic Model



# Limitations of document-Term matrix

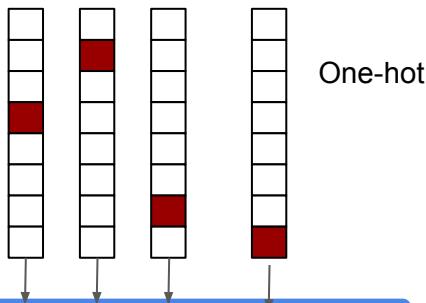
- Too strong assumption: all words are independent of each other
  - $|orange - peach| < |orange - car|$
- Can not capture the order information in the sequence
- High dimensionality due to large size of vocabulary

an	apple	car	day	driving	eat	every	like	my	orange	to	work
0	1	1	0	1	0	1	1	0	0	0	0
1	1	0	0	1	0	1	1	0	0	1	0
2	0	0	1	0	1	0	0	1	1	0	1

# A new perspective on BoW

- Each word in vocab is represented in one-hot embedding
- Sum one-hot vectors of the words in a sentence
- The final vector is the representation for the given sentence and then fed into a classifier.

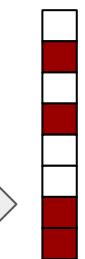
I eat an apple



One-hot

Composition Function:  
Sum

BoW  
Features

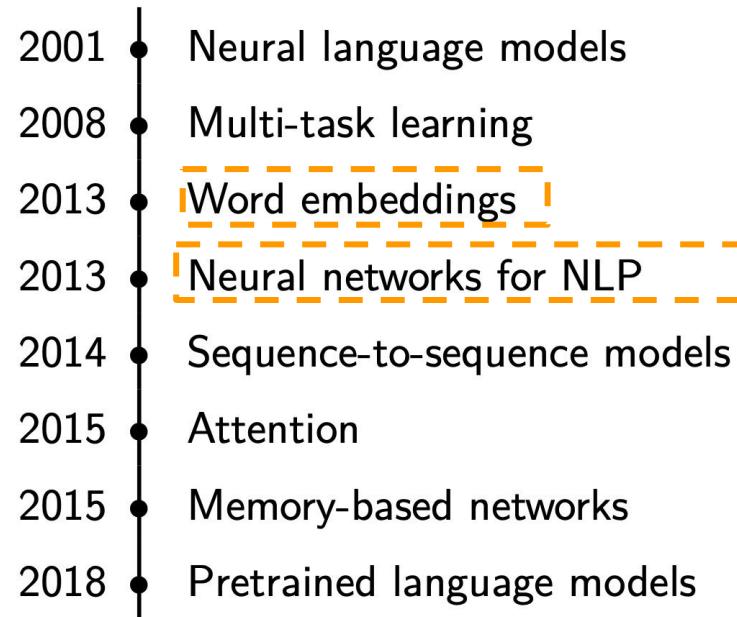


How to improve?

Classification Models

Labels

# Neural NLP



[https://www.kamperh.com/slides/ruder+kamper\\_indaba2018\\_talk.pdf](https://www.kamperh.com/slides/ruder+kamper_indaba2018_talk.pdf)

## 2. Word Embeddings

# Word representation

- How to represent words in a vector space

apple [0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0]

orange [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 ... 0 0 0 0 0]

car [0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0]

# Distributed representation

- Words should be encoded into a low-dimensional and dense vector

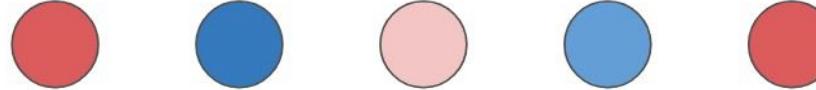
apple



orange



car



# Word vectors

Project word vectors in a two-dimensional space. And visualize them!

apple  
orange  
banana

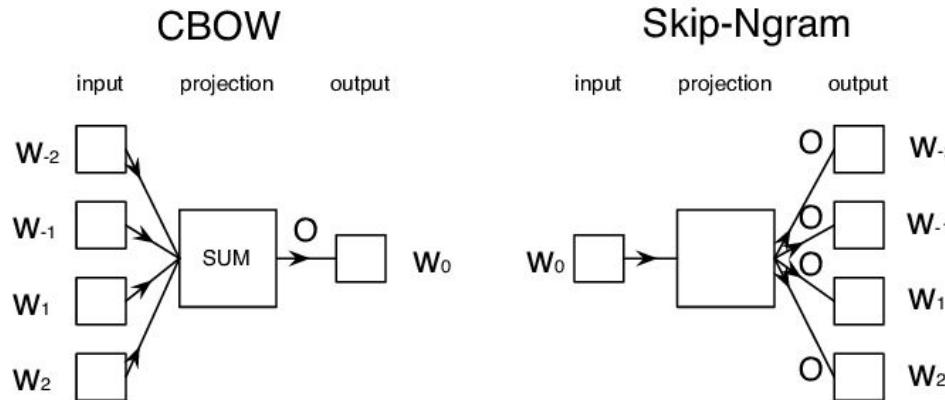
rice  
juice  
milk

bus  
train  
car

Similar words are close to each other.

# Word2Vec

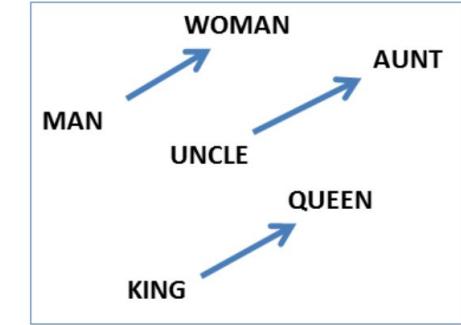
- A method of computing vector representation of words developed by Google.
- Open-source version of Word2Vec hosted by Google (in C)
- Train a simple neural network with a single hidden layer to perform word prediction tasks.
- Two structures proposed Continuous Bag of Words (CBoW) vs Skip-Gram



# Word2Vec as BlackBox



input, output



Corpus	Word2Vec Tool	Word Embeddings
A small image of an old document page with vertical columns of Chinese text, representing the input corpus.	A solid black cube, representing the Word2Vec tool as a black box.	A diagram illustrating vector relationships between words. It shows arrows originating from the word "MAN" pointing to "WOMAN", "UNCLE", and "QUEEN". Another arrow originates from "KING" and points to "AUNT". This represents the learned vector space where related words are close together.

Corpus

Word2Vec Tool

Word Embeddings

# A Good Visualization for Word2Vec

<https://ronxin.github.io/wevi/>

# Target

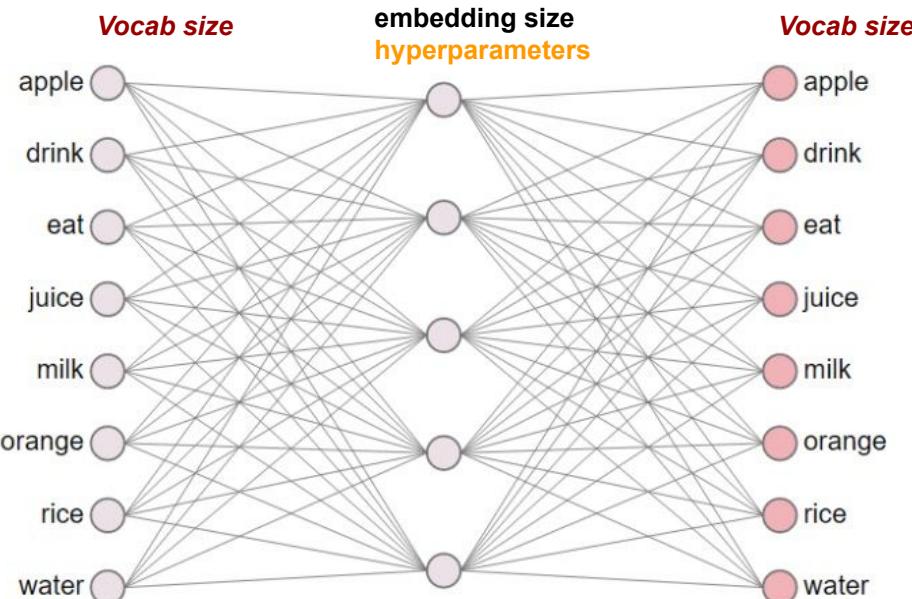
- Given a training corpus, we prepare a list of N (input\_word, output\_word).
- Objective Function: Maximize probability of all the output words given the corresponding input words.

$$\mathbf{J}(\theta) = \prod_{i=1}^N p(w_{output}^i | w_{input}^i, \theta)$$



**Neural network  
parameters that will  
be optimized**

# Model architecture



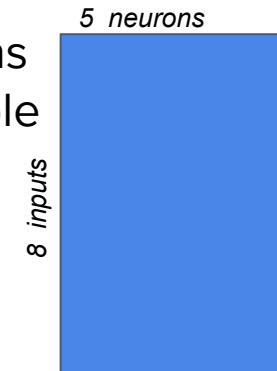
## Structure Highlights:

- **input layer**
  - one-hot vector
- **hidden layer**
  - linear (identity)
- **output layer**
  - softmax

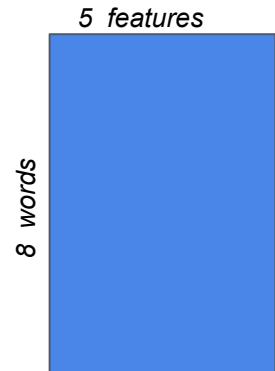
# Hidden layer

- **Linear-activation** function here
- **5** neurons are the word vec. dimensions
- This layer is operating as a ‘lookup’ table
- Input word matrix denoted as **IVec**

Hidden Layer Weights Matrix



Word Vector Look Up Table



One-hot vector

[0,0,**1**,0,0,0,0, 0]

Index of eat

1.06 2.91 0.29 1.39 0.33  
1.60 1.12 0.29 0.74 0.21  
0.96 1.50 1.37 0.34 1.04  
0.53 2.11 0.76 2.51 0.20  
0.31 0.64 2.08 0.24 1.23  
1.40 1.36 0.01 1.69 1.95  
2.97 2.13 0.86 0.90 2.21  
1.05 0.80 2.18 2.43 1.57



Word vector for “eat”

**0.96, 1.5, 1.37, 0.34, 1.04**

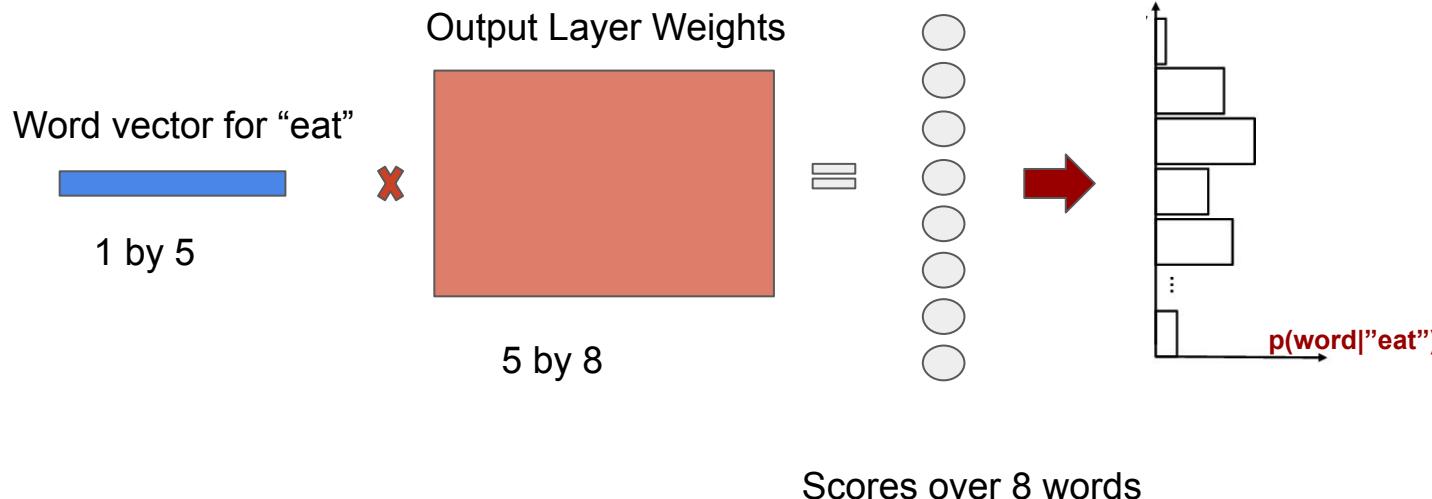
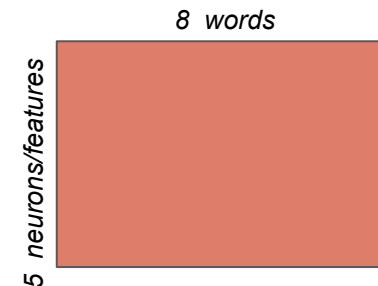


This is a **projection/look up** process: given the index of the word, we take the  $i$ th row in the word vector matrix out

# Output layer

- Softmax Classifier
- Output word matrix denoted as **OVec**

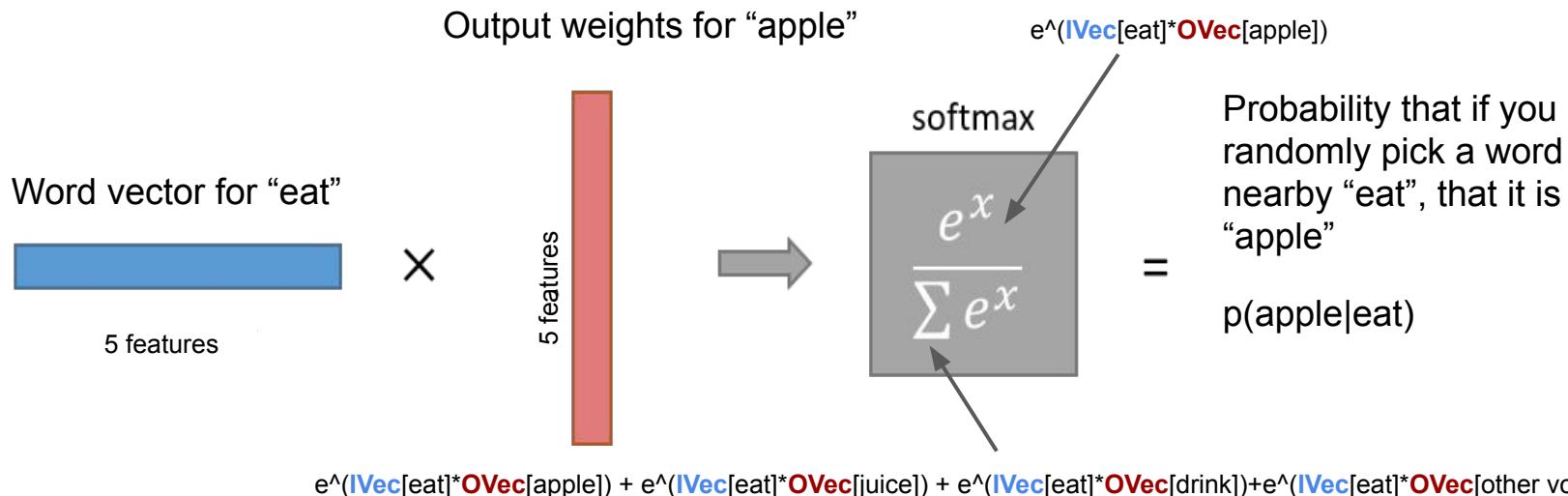
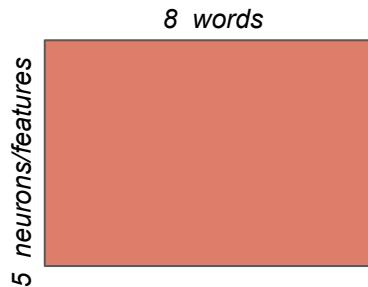
Output Layer Weights Matrix  
A.K.A Output word vectors



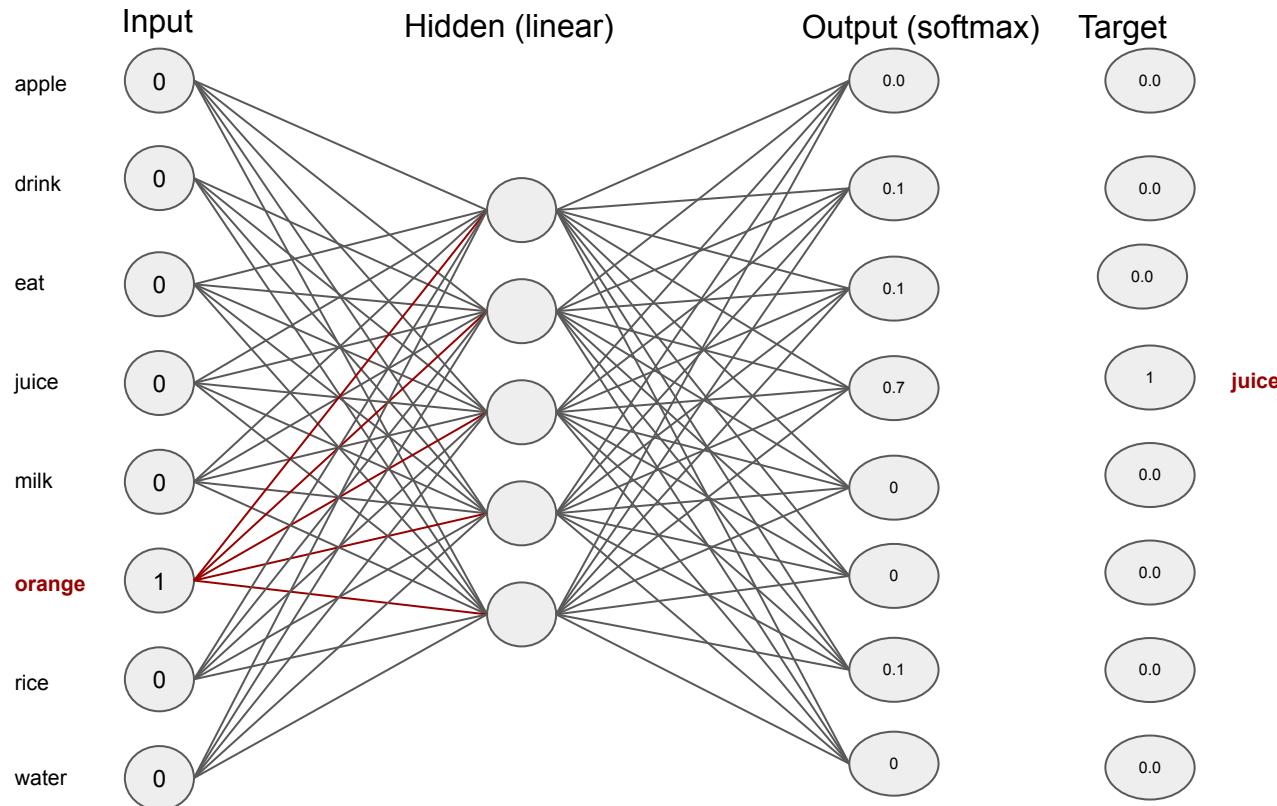
# Output layer

- Softmax Classifier
- Output word matrix denoted as **OVec**

Output Layer Weights Matrix  
A.K.A Output word vectors

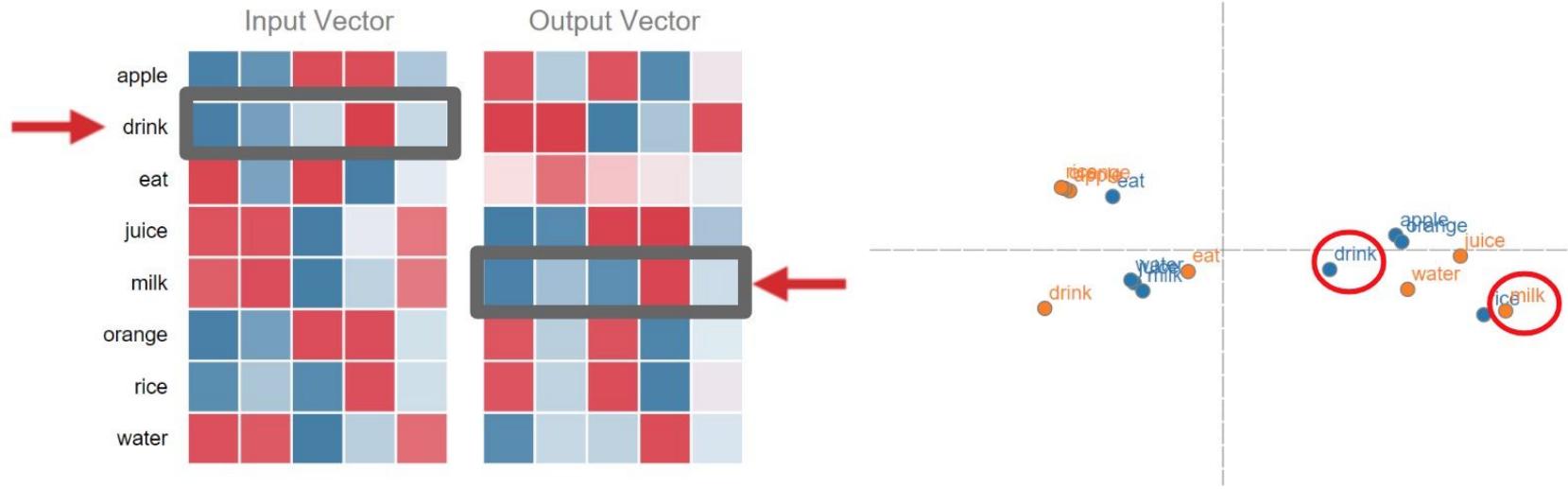


# Word2Vec

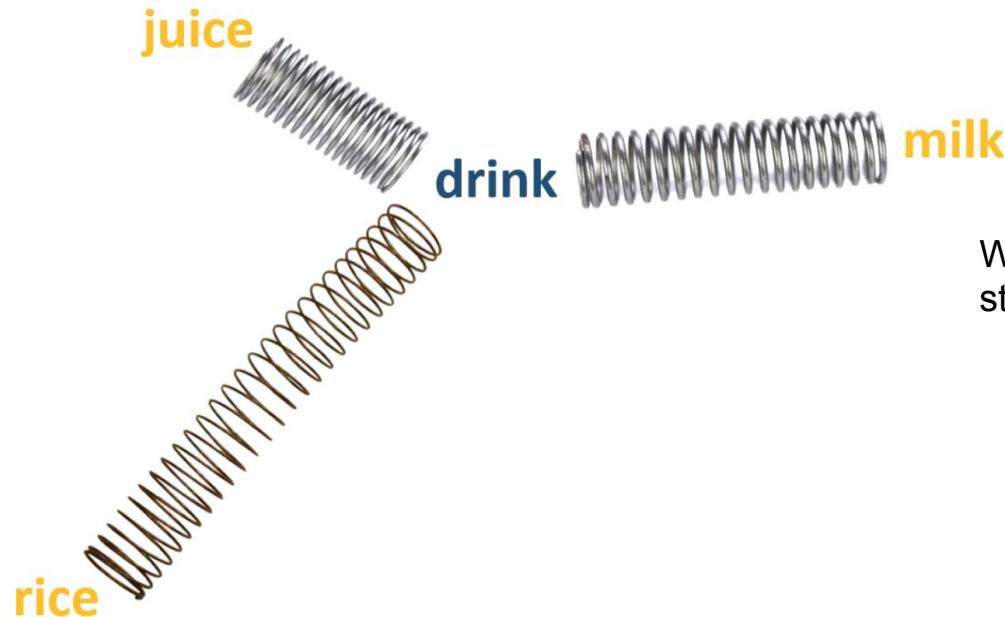


Then, we can compute the **loss** and call gradient descent to update model parameters.

# Updating word vectors



# A force-directed graph



What decides the strength of the string?

# Idea behind Word2Vec

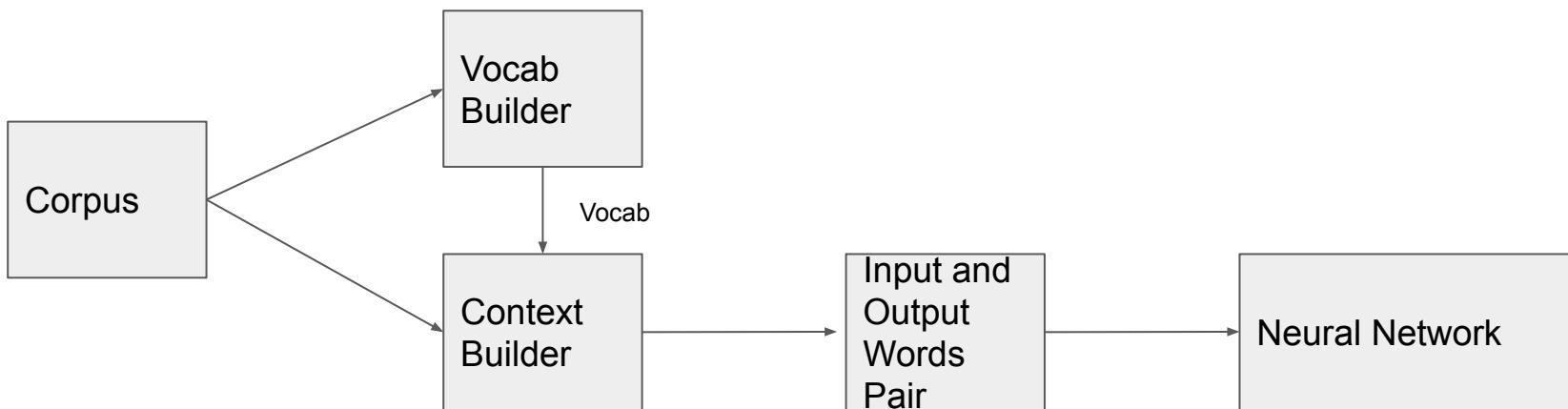
- Feature vector assigned to a word will be adjusted if it can not be used for accurate prediction of that word's context.
- Each word's context in the corpus is the teacher sending error signals back to modify the feature vector.
- It means that words with **similar context** will be assigned **similar vectors!**



**“You shall know a word by the company it keeps”** - by Firth (1957)

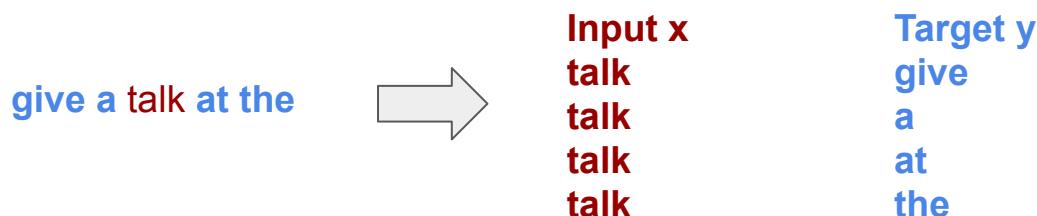
# Input and output words

- How to select them from corpus
- Skip-gram and CBoW differ here



# Skip-Gram

- Task Definition: given a specific word, predict its nearby word (probability output)
- Model input: source word, Model output: nearby word
- Input is one word, output is one word
- The output can be interpreted as prob. scores, which are regarded as how likely it is that each vocabulary word can be nearby your input word.



# CBoW

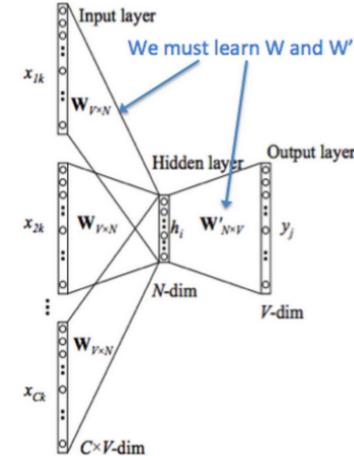
- Task Definition: given context, predict its target word
- Model input: context (several words), Model output: center word
- Input is several words, output is one word
- Core Trick: **average** these context vectors for prob. score computing

give a talk at the



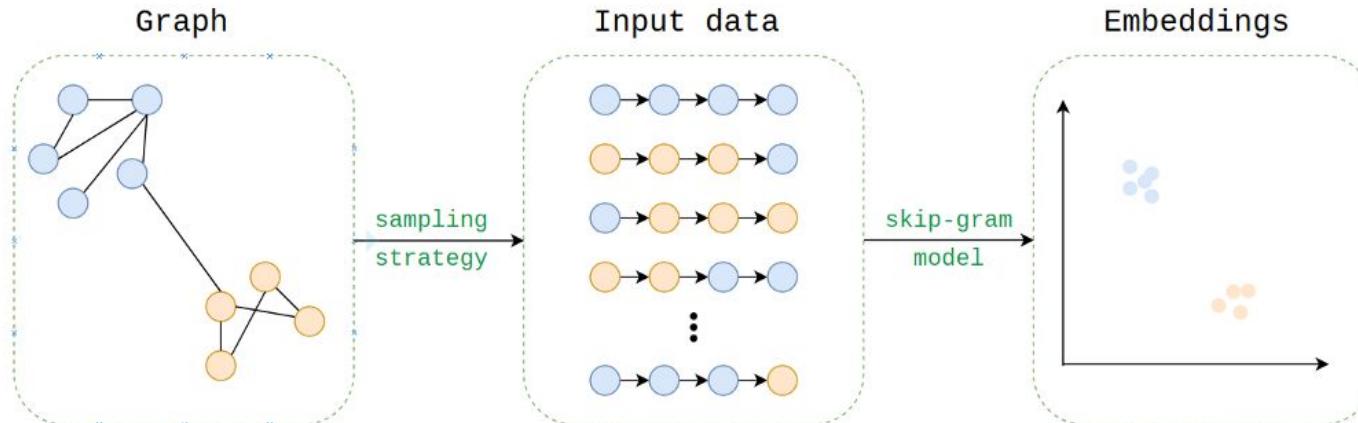
Input x  
(give,a,at,the)

Target y  
talk



# Embedding for graph data

- Embeddings can be extended beyond NLP domain
- Embeddings can be learned for any nodes in a graph
- Nodes can be items, web pages and so on in user clicked stream data
- Embeddings can be learned for any group of discrete and co-occurring states.

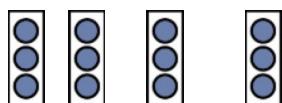


### **3. Neural Networks for NLP**

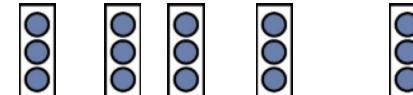
# Sequence of words

- Each sentence or document can be regarded as a sequence of vectors.
- The shape of matrix depends on the length of sequence. However, the majority of ML systems need fixed-length feature vectors.
- One simple solution: average the sequence of vectors, just like bag-of-words (abandon order information).

I hate this movie



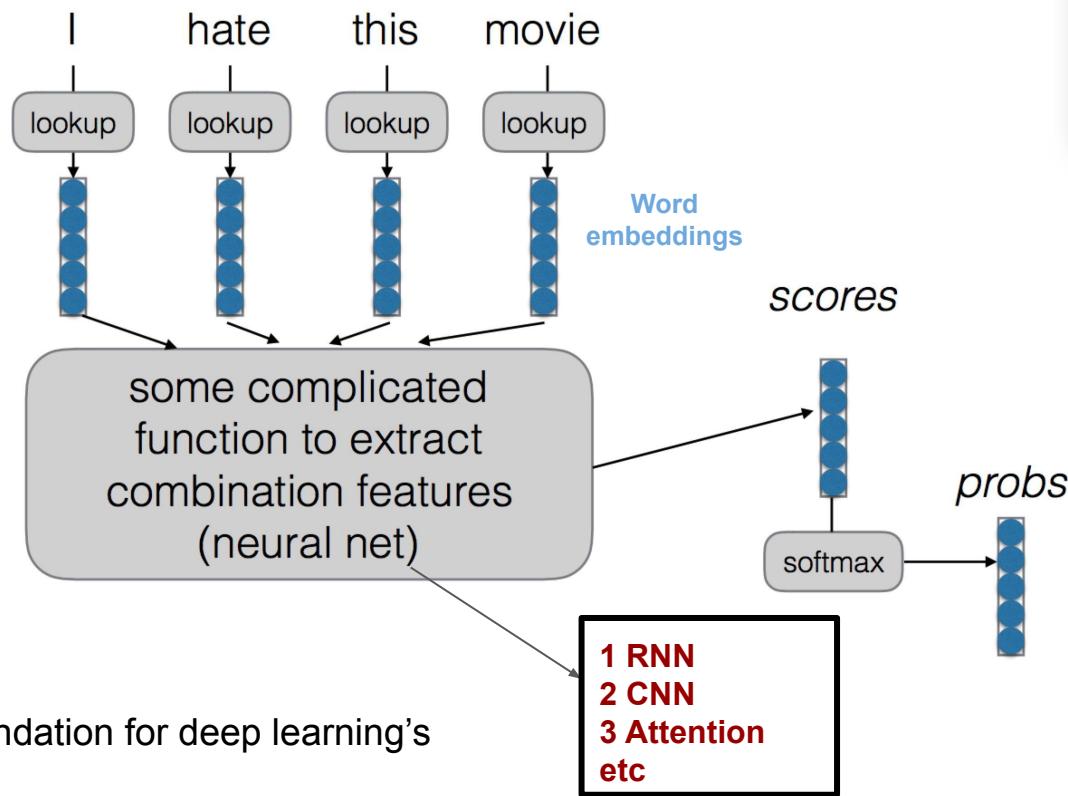
This is my favorite movie.



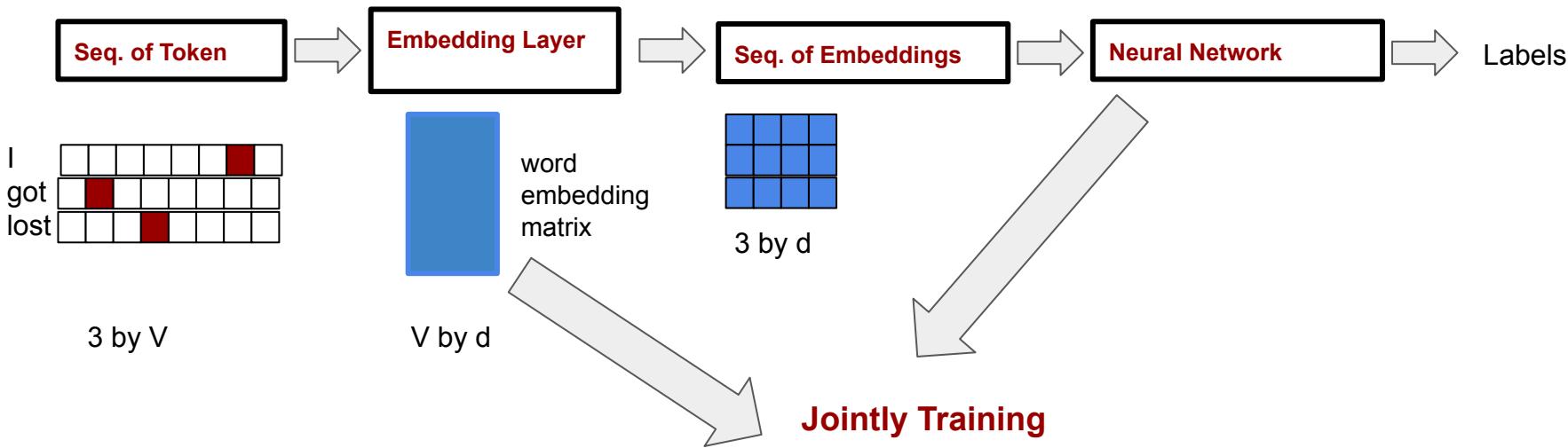
4 by d

5 by d

# Complex semantic



# Neural networks for NLP



- Learn from Scratch: Random initialize the word embedding matrix and update the matrix and neural network parameters in the specific task
- Pre-train: Got pre-trained word embeddings as the embedding layer and only update neural network parameters in the specific task
- **Pre-train then fine tune**

# Convolution operation

Word Vectors

I  
like  
this  
movie  
very  
much  
!

0.8	0.5	0.2	-0.1	0.4
0.8	0.9	0.1	0.5	0.1
0.4	0.6	0.1	-0.1	0.7
---	---	---	---	---
---	---	---	---	---
---	---	---	---	---
---	---	---	---	---

0.2	0.1	0.2	0.1	0.1
0.1	0.1	0.4	0.1	0.1

Filters updated  
during training

0.51

I  
like  
this  
movie  
very  
much  
!

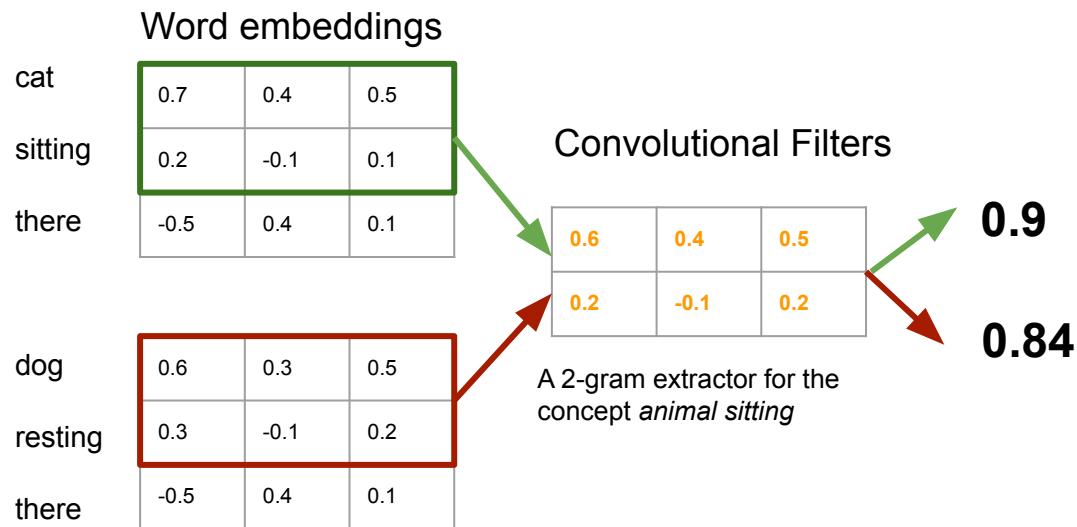
0.8	0.5	0.2	-0.1	0.4
0.8	0.9	0.1	0.5	0.1
0.4	0.6	0.1	-0.1	0.7
---	---	---	---	---
---	---	---	---	---
---	---	---	---	---
---	---	---	---	---

0.2	0.1	0.2	0.1	0.1
0.1	0.1	0.4	0.1	0.1

Feature Maps

0.51

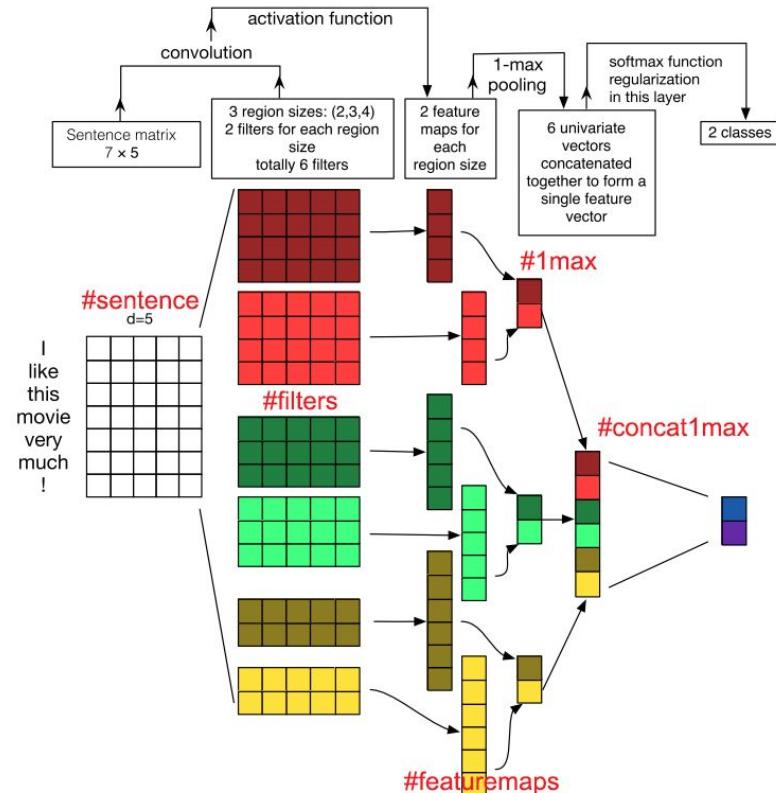
# Toy example



- This convolution provides high activations for 2-grams with certain meaning
- Can be extended to 3-grams, 4-grams, etc.
- Can have various filters, need to track many n-grams.
- They are called 1D since we only slice the windows only in one direction

Why is it better than BoW?

# CNN framework

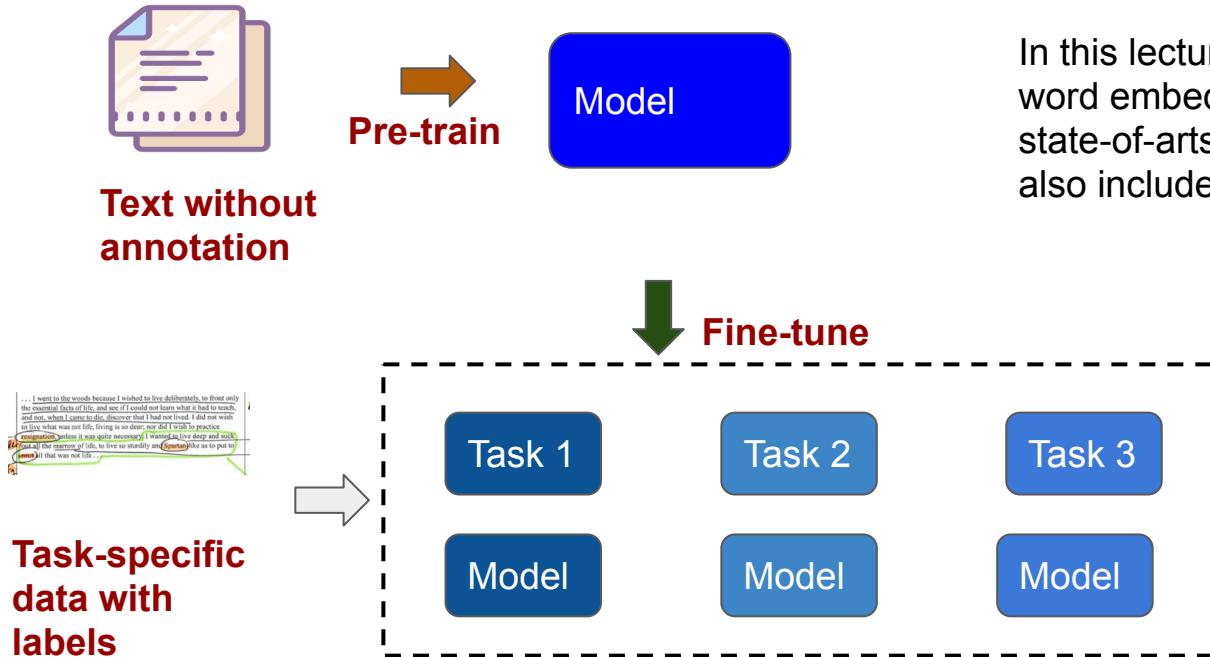


From Zhang 2015

# How to build word embedding layer

- Learn from Scratch: Random initialize the word embedding matrix and update the matrix and neural network parameters in the specific task
- Pre-train: Got pre-trained word embeddings as the embedding layer and only update neural network parameters in the specific task
- Pre-train then fine tune

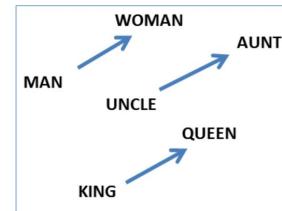
# Pre-train then Fine-tune



In this lecture, the model is only referred to word embedding matrix. However, in state-of-arts NLP techniques, the model will also include the following neural network.

# Is Word2Vec good enough?

- Can not capture different senses of words (context independent)
  - Solution: Take the word order into account
- Can not address Out-of-Vocabulary words
  - Solution: Use characters or subwords



Word Embeddings used for downstream tasks

Corpus

Word2Vec Tool

## **4. Attention is all you Need**

---

# Attention Is All You Need

---

**Ashish Vaswani\***

Google Brain

avaswani@google.com

**Noam Shazeer\***

Google Brain

noam@google.com

**Niki Parmar\***

Google Research

nikip@google.com

**Jakob Uszkoreit\***

Google Research

usz@google.com

**Llion Jones\***

Google Research

llion@google.com

**Aidan N. Gomez\*** <sup>†</sup>

University of Toronto

aidan@cs.toronto.edu

**Lukasz Kaiser\***

Google Brain

lukaszkaiser@google.com

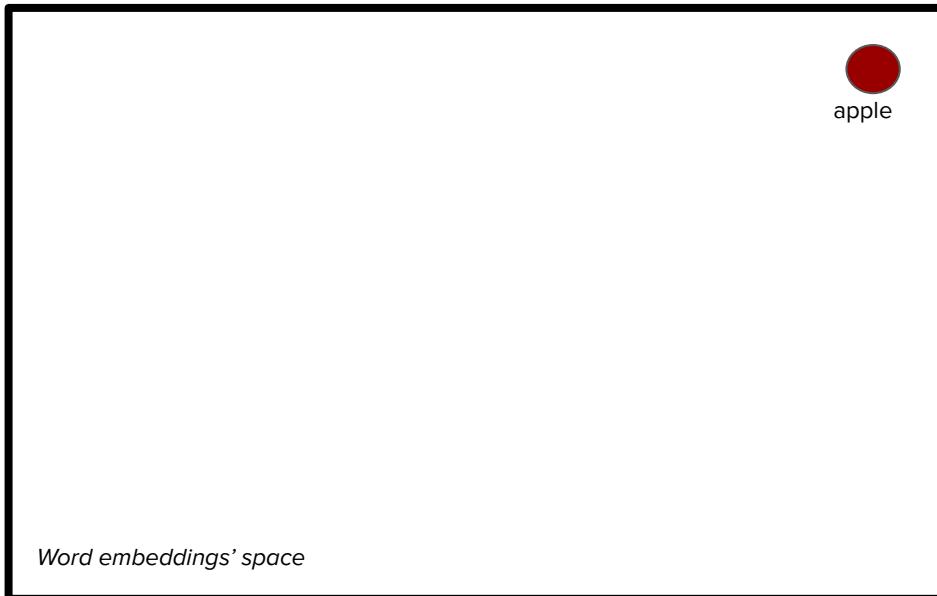
**Illia Polosukhin\*** <sup>‡</sup>  
illia.polosukhin@gmail.com

## Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

# Word Embeddings

- **Apple** in two sentences:
  - Sentence 1: My favorite fruit is **apple**
  - Sentence 2: Solution: My favorite brand is **apple**



One embedding has multiple senses

# Contextualized Word Embeddings

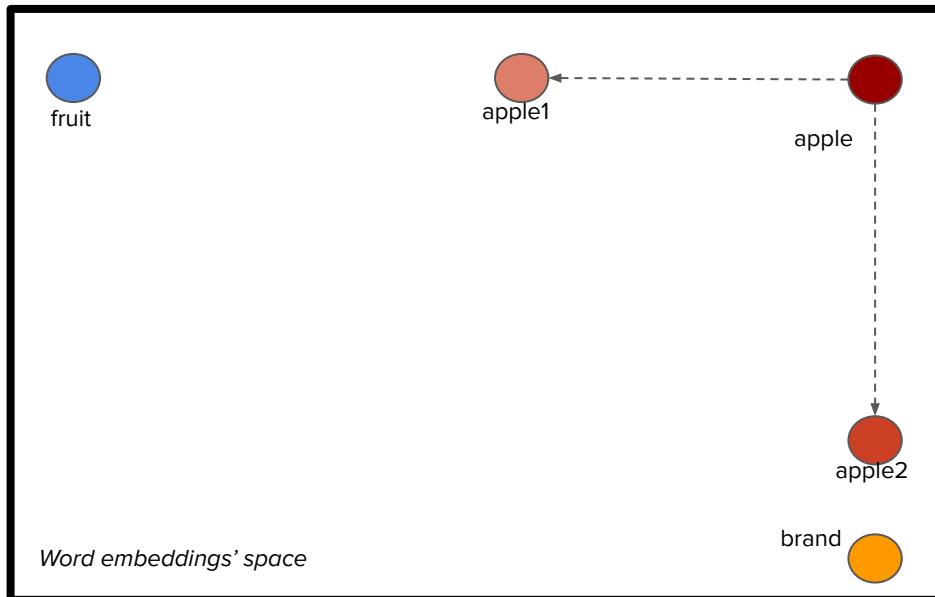
- Telling context in words
  - Sentence 1: My favorite **fruit** is **apple1**
  - Sentence 2: Solution: My favorite **brand** is **apple2**



From nearby words, we can guess two different meanings of this word (i.e., food and brand)

# Contextualized Word Embeddings

- Telling context in words
  - Sentence 1: My favorite **fruit** is **apple1**
  - Sentence 2: Solution: My favorite **brand** is **apple2**



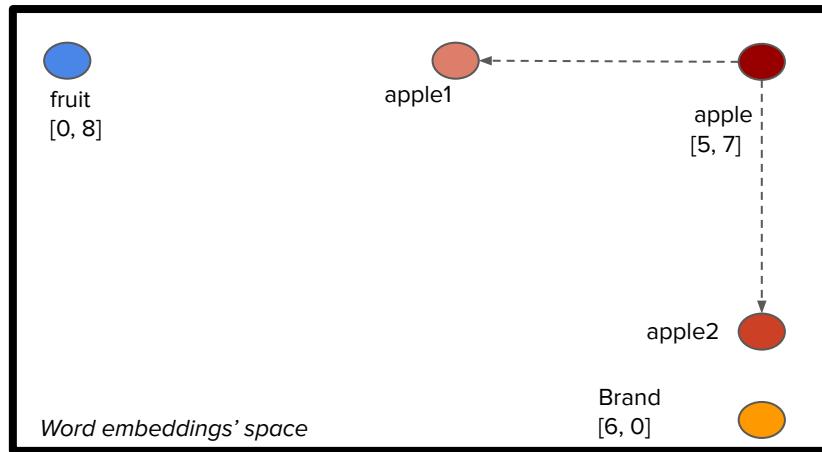
1. In sentence 1, move the word embedding of apple towards the word “fruit”
2. In sentence 2, move the word embedding of apple toward the word “brand”

# How to move one word closer to another one

- Average two words
  - $\text{apple1} = 0.8 \cdot \text{apple} + 0.2 \cdot \text{fruit} = 0.8 \cdot [5, 7] + 0.2 \cdot [0, 8] = [4.0, 7.2]$
  - $\text{apple2} = 0.9 \cdot \text{apple} + 0.1 \cdot \text{brand} = 0.9 \cdot [5, 7] + 0.1 \cdot [6, 0] = [5.1, 6.3]$

Similarity/Attention

Embeddings



Attention mechanism is able to learn multiple embeddings for the same word in multiple sentences

# How to derive similarity

- **Apple** in two sentences:
  - Sentence 1: My favorite fruit is **apple**
  - Sentence 2: My favorite brand is **apple**
- Why we move apple to fruit in sentence 1? Instead of other words as “my” and “is”
- It is based on the similarity!

Contextualized word embedding of apple in the sentence: my favorite fruit is apple

$$= \text{Attention}(\text{apple}, \text{my}) * \text{base\_vec}(\text{my}) + \text{Attention}(\text{apple}, \text{favorite}) * \text{base\_vec}(\text{favorite}) + \text{Attention}(\text{apple}, \text{fruit}) * \text{base\_vec}(\text{fruit}) + \text{Attention}(\text{apple}, \text{is}) * \text{base\_vec}(\text{is}) + \text{Attention}(\text{apple}, \text{apple}) * \text{base\_vec}(\text{apple})$$

# How to derive similarity

- In good embeddings, the similarity between two irrelevant words would be zero, while the similarity between the related pair would be high

	my	favourite	fruit	is	apple
my	1	0	0	0	0
favourite	0	1	0	0	0
fruit	0	0	1	0	0.25
is	0	0	0	1	0
apple	0	0	0.25	0	1

	my	favourite	brand	is	apple
my	1	0	0	0	0
favourite	0	1	0	0	0
brand	0	0	1	0	0.11
is	0	0	0	1	0
apple	0	0	0.11	0	1

# How to derive similarity

- The diagonal entries are all 1
- The similarity between any irrelevant words is 0 (for simplicity)
- The similarity between apple and fruit is 0.25 while the one between apple and brand is 0.11 considering apple is used more often in the same context as fruit

	my	favourite	fruit	is	apple
my	1	0	0	0	0
favourite	0	1	0	0	0
fruit	0	0	1	0	0.25
is	0	0	0	1	0
apple	0	0	0.25	0	1

	my	favourite	brand	is	apple
my	1	0	0	0	0
favourite	0	1	0	0	0
brand	0	0	1	0	0.11
is	0	0	0	1	0
apple	0	0	0.11	0	1

# How to derive similarity

- Contextualized Target Word = The sum of a product between the similarity between target word and context word \* context word
- We should also normalize the similarity along the sentence
- Therefore
  - my (in the sentence 1) = my
  - apple (in the sentence 1) =  $0.2 * \text{fruit} + 0.8 * \text{apple}$

	my	favourite	fruit	is	apple
my	1	0	0	0	0
favourite	0	1	0	0	0
fruit	0	0	1	0	0.25
is	0	0	0	1	0
apple	0	0	0.25	0	1

	my	favourite	brand	is	apple
my	1	0	0	0	0
favourite	0	1	0	0	0
brand	0	0	1	0	0.11
is	0	0	0	1	0
apple	0	0	0.11	0	1

# Attention Mechanism

Sentence i: My favourite fruit is apple

	my	favourite	fruit	is	apple
my	1	0	0	0	0
favourite	0	1	0	0	0
fruit	0	0	1	0	0.25
is	0	0	0	1	0
apple	0	0	0.25	0	1

New Word Index

my\_i

favourite\_i

fruit\_i

is\_i

apple\_i

Attention Step

my

favourite

$0.8 \cdot \text{fruit} + 0.2 \cdot \text{apple}$

is

$0.2 \cdot \text{fruit} + 0.8 \cdot \text{apple}$

Here, we only use the **same embedding** for attention weights and the base vectors

# Self-Attention Layer

Sentence i: My favourite fruit is apple

New Word Index

Attention Step

	my	favourite	fruit	is	apple
my	1	0	0	0	0
favourite	0	1	0	0	0
fruit	0	0	1	0	0.25
is	0	0	0	1	0
apple	0	0	0.25	0	1

my\_i

my

favourite\_i

favourite

fruit\_i

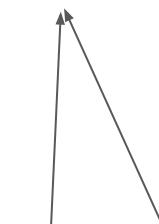
0.8\*fruit+0.2\*apple

is\_i

is

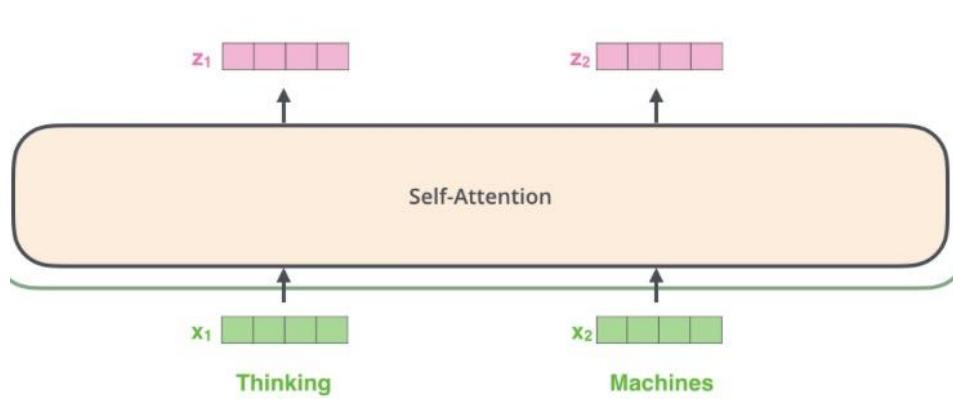
apple\_i

0.2\*fruit+0.8\*apple



Query, Key and Value vector would be created for each word

# Self-Attention Layer



A sequence of vectors in,  
A sequence of vector out

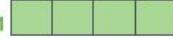
# Self-Attention Layer

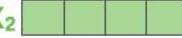
Input

Thinking

Machines

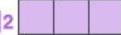
Embedding

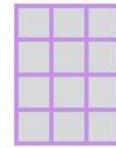
$X_1$  

$X_2$  

Queries

$q_1$  

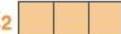
$q_2$  

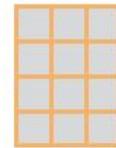


$W^Q$

Keys

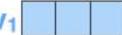
$k_1$  

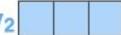
$k_2$  

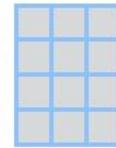


$W^K$

Values

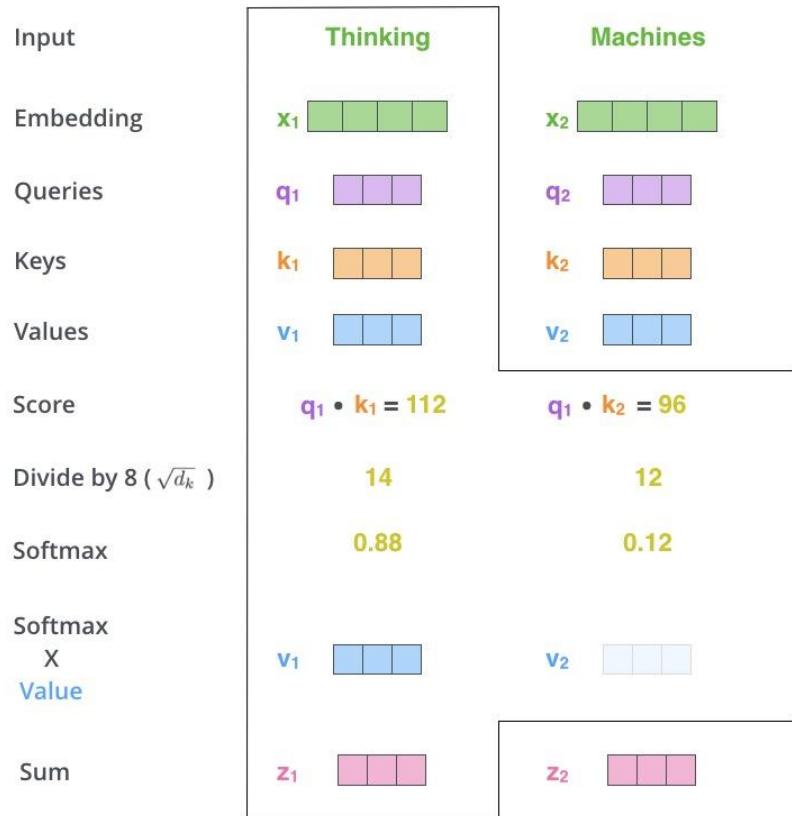
$v_1$  

$v_2$  



$W^V$

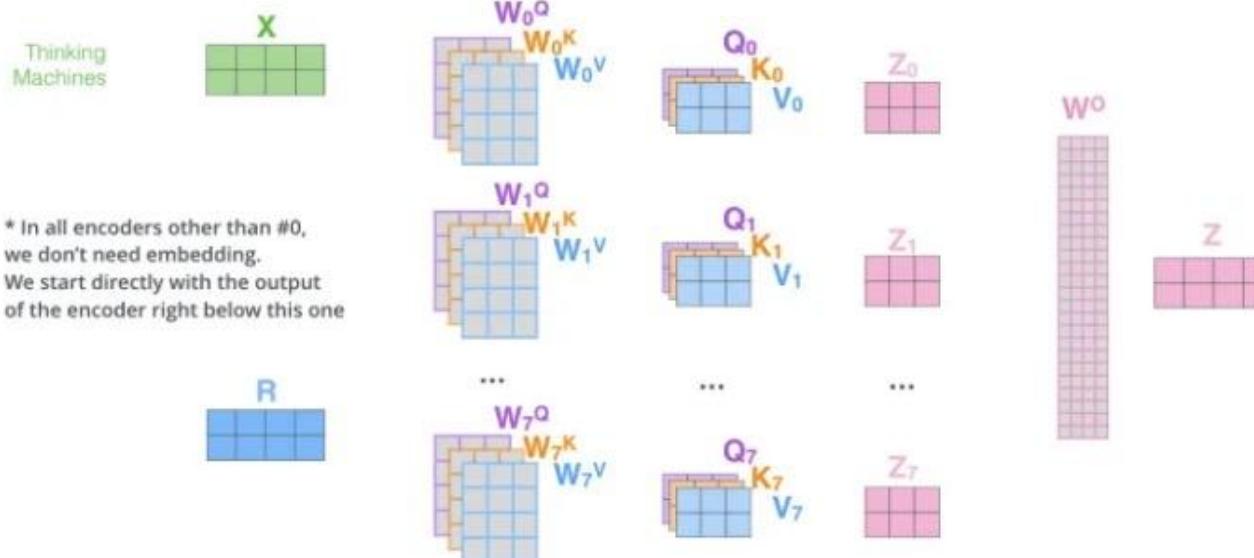
# Self-Attention Layer



Each pair of q, k, v transformation is corresponding to each neuron. In attentional layer, we call it multihead (multi-neurons)

# MultiHeadAttention

- 1) This is our input sentence\* each word\*
- 2) We embed
- 3) Split into 8 heads. We multiply  $X$  or  $R$  with weight matrices
- 4) Calculate attention using the resulting  $Q/K/V$  matrices
- 5) Concatenate the resulting  $Z$  matrices, then multiply with weight matrix  $W^o$  to produce the output of the layer



What are model parameters?

# MultiHeadAttention in Keras

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import MultiHeadAttention

target = tf.keras.Input(shape=[6, 16])

# assume it is a sentence of 6 words. Then, each word has a

layer = MultiHeadAttention(num_heads=1, key_dim=2)
output_tensor, attention_scores = layer(target, target, return_attention_scores=True)
print(output_tensor.shape)
print(attention_scores.shape)

(None, 6, 16)
(None, 1, 6, 6)

for matrix in layer.weights:
    print(matrix.shape)

(16, 1, 2)
(1, 2)
(16, 1, 2)
(1, 2)
(16, 1, 2)
(1, 2)
(1, 2, 16)
(16,)
```

# MultiHeadAttention in Keras

```
: layer = MultiHeadAttention(num_heads=3, key_dim=2)
target = tf.keras.Input(shape=[6, 16])
output_tensor, attention_scores = layer(target, target, return_attention_scores=True)
print(output_tensor.shape)
print(attention_scores.shape)

(None, 6, 16)
(None, 3, 6, 6)
```

```
: for matrix in layer.weights:
    print(matrix.shape)

(16, 3, 2)
(3, 2)
(16, 3, 2)
(3, 2)
(16, 3, 2)
(3, 2)
(3, 2, 16)
(16,)
```

# MultiHeadAttention in Keras

If we change the key\_dim from 2 to 5?

```
: layer = MultiHeadAttention(num_heads=3, key_dim=2)
target = tf.keras.Input(shape=[6, 16])
output_tensor, attention_scores = layer(target, target, return_attention_scores=True)
print(output_tensor.shape)
print(attention_scores.shape)

(None, 6, 16)
(None, 3, 6, 6)

for matrix in layer.weights:
    print(matrix.shape)

(16, 3, 2)
(3, 2)
(16, 3, 2)
(3, 2)
(16, 3, 2)
(3, 2)
(3, 2, 16)
(16,)
```

[https://github.com/rz0718/BT5153\\_2023/blob/main/codes/lab\\_lecture08/Attention\\_Layer\\_in\\_Keras.ipynb](https://github.com/rz0718/BT5153_2023/blob/main/codes/lab_lecture08/Attention_Layer_in_Keras.ipynb)

# Keras Implementation

[https://keras.io/examples/nlp/text\\_classification\\_with\\_transformer/](https://keras.io/examples/nlp/text_classification_with_transformer/)

# Details Behind Transformer

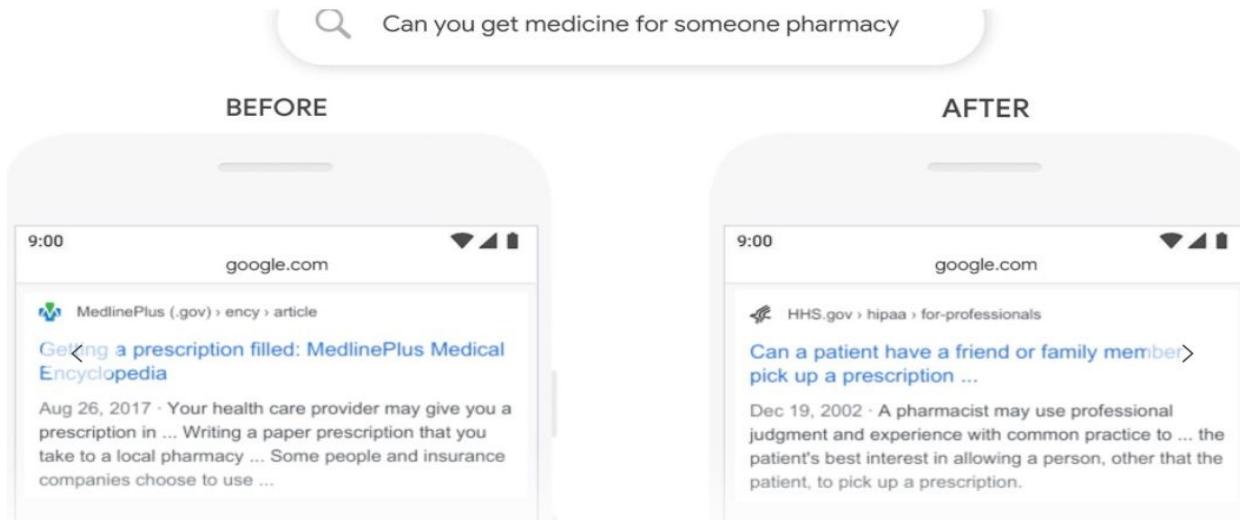
1. <https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/>
2. <http://jalammar.github.io/illustrated-transformer/>

Visualizing Ichiban!!!



## 5. Introduction to BERT

# BERT in Google Search



With the latest advancements from our research team in the science of language understanding—made possible by machine learning—we're making a significant improvement to how we understand queries, representing **the biggest leap** forward in the past five years, and one of the biggest leaps forward in the history of Search.

# BERT vs Somebody



Try:

<https://huggingface.co/bert-base-uncased?text;if+you+don%27t+want+to+inhale+virus+is%2C+you+should+wear+a+%5BMASK%5D>

# Extraction-based QA using BERT

```
question = "What can we learn in NUS?"  
answer_text = "The National University of Singapore (NUS) is the national research university of Singapore. \  
Founded in 1905 as the Straits Settlements and Federated Malay States Government Medical School, NUS is the oldest higher education institution in Singapore. \  
It is consistently ranked within the top 20 universities in the world and is considered to be the best university in the Asia-Pacific. \  
NUS is a comprehensive research university, \  
offering a wide range of disciplines, including the sciences, medicine and dentistry, design and environment, law, arts and social sciences, engineering, business, computing and music \  
at both the undergraduate and postgraduate levels."
```



```
print('Answer: "' + answer + '"')
```

```
Answer: "sciences , medicine and dentistry , design and environment , law , arts and social sciences , engineering , business , computing and music"
```

```
question = "What does NUS mean?"  
answer_text = "The National University of Singapore (NUS) is the national research university of Singapore. \  
Founded in 1905 as the Straits Settlements and Federated Malay States Government Medical School, NUS is the oldest higher education institution in Singapore. \  
It is consistently ranked within the top 20 universities in the world and is considered to be the best university in the Asia-Pacific. \  
NUS is a comprehensive research university, \  
offering a wide range of disciplines, including the sciences, medicine and dentistry, design and environment, law, arts and social sciences, engineering, business, computing and music \  
at both the undergraduate and postgraduate levels."
```

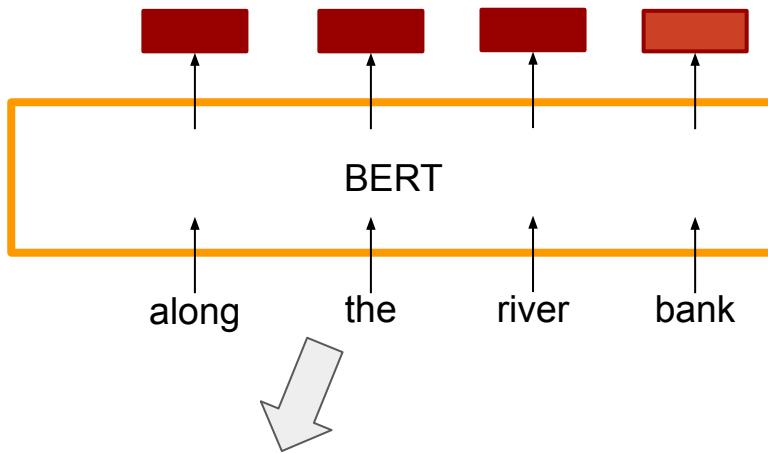


```
print('Answer: "' + answer + '"')
```

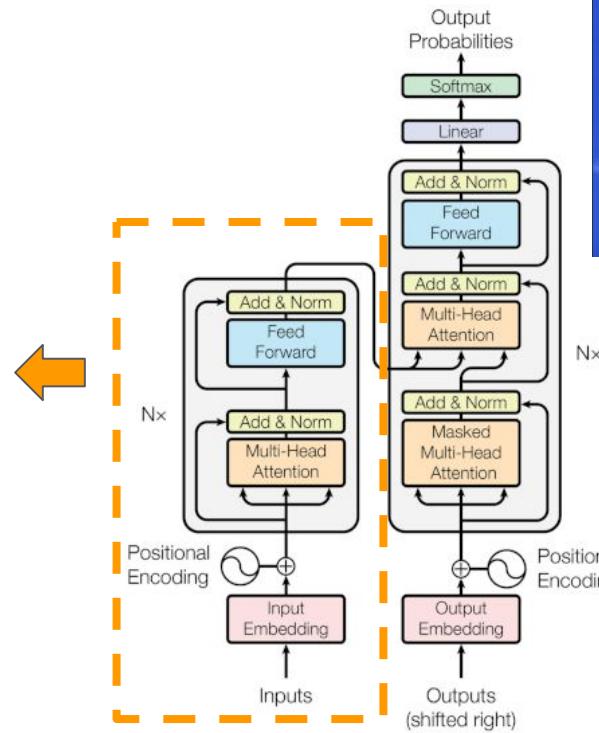
```
Answer: "national university of singapore"
```

# What is BERT

- Bidirectional Encoder Representations from Transformers (**BERT**)
- BERT: Encoder of Transformer,



Given a sequence of words, generate a sequence of vectors and then can be used for various NLP tasks



Transformer

Solve Seq2Seq Task

# Neural Networks for NLP



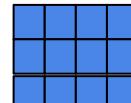
I  
got  
lost


3 by V

V by d



word  
embedding  
matrix



3 by d



Each word has a fixed vector

Can not address multi-sense  
problem!

# Multiple Senses of Words

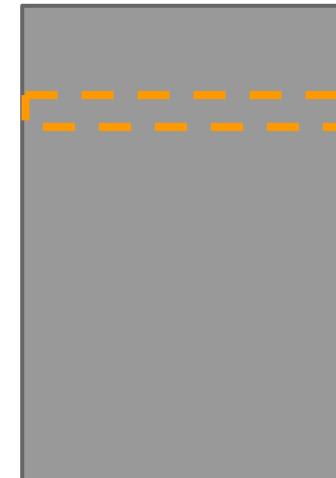
- It is safest to deposit your money in the bank.
- All the animals lined up along the river bank.
- Today, blood banks collect blood.

The third sense of not?

Word2Vec, Fasttext,Glove and other  
word embedding models



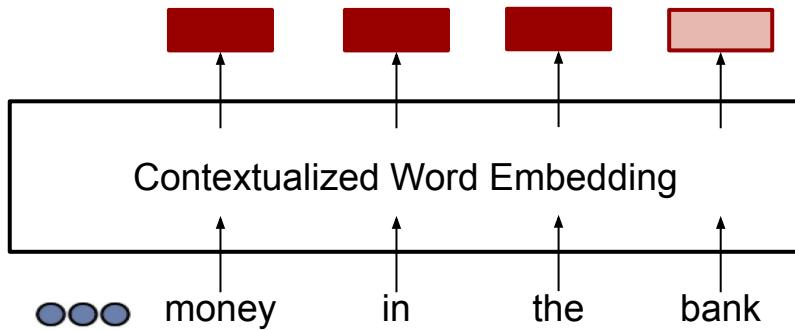
Vocab  
size



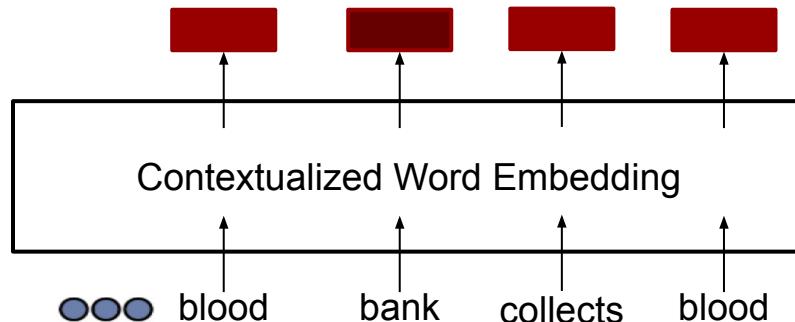
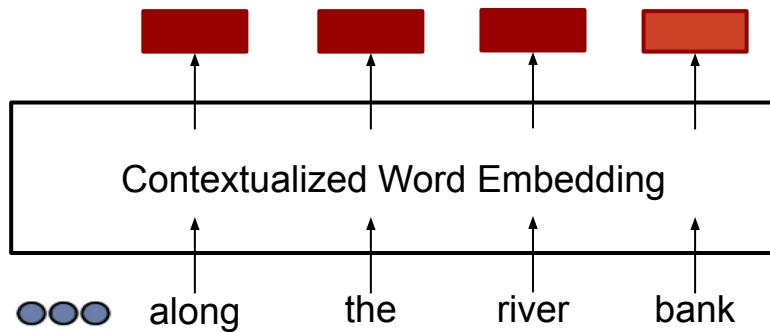
The index  
of “bank”

# Contextualized Word Embeddings

Encoded Embeddings

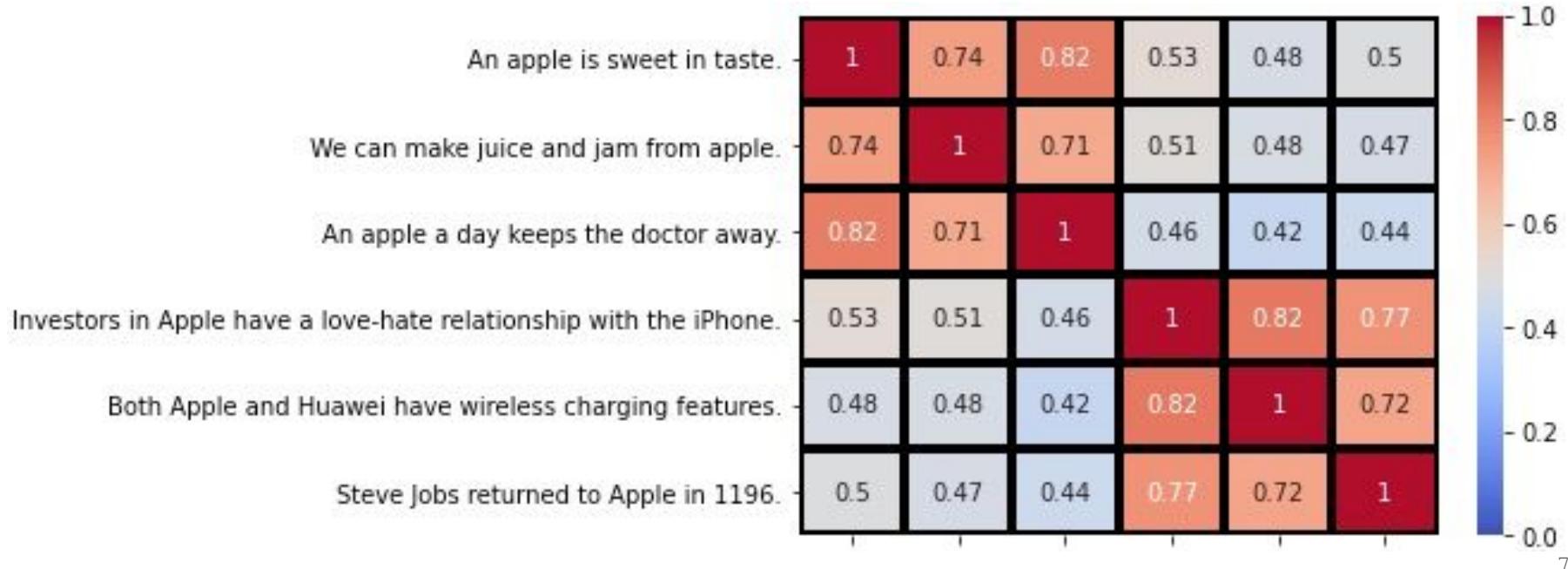


Different Contexts,  
Different Encoded Embeddings for bank.



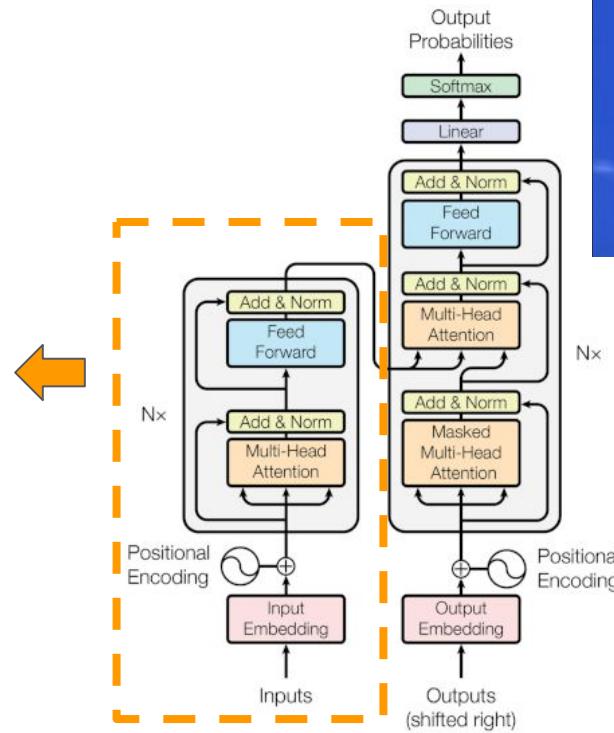
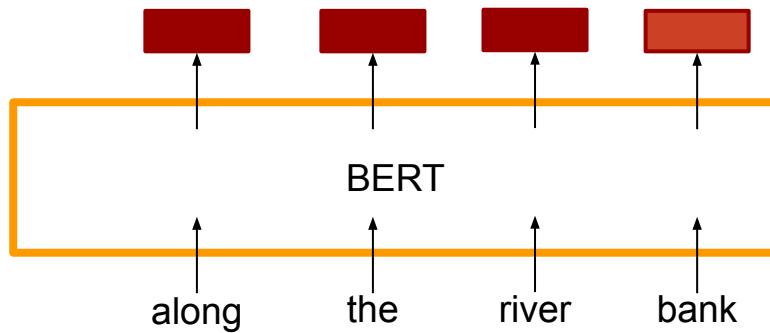
# Embeddings generated from BERT

Cos-similarities among vectors of “apple” in different context

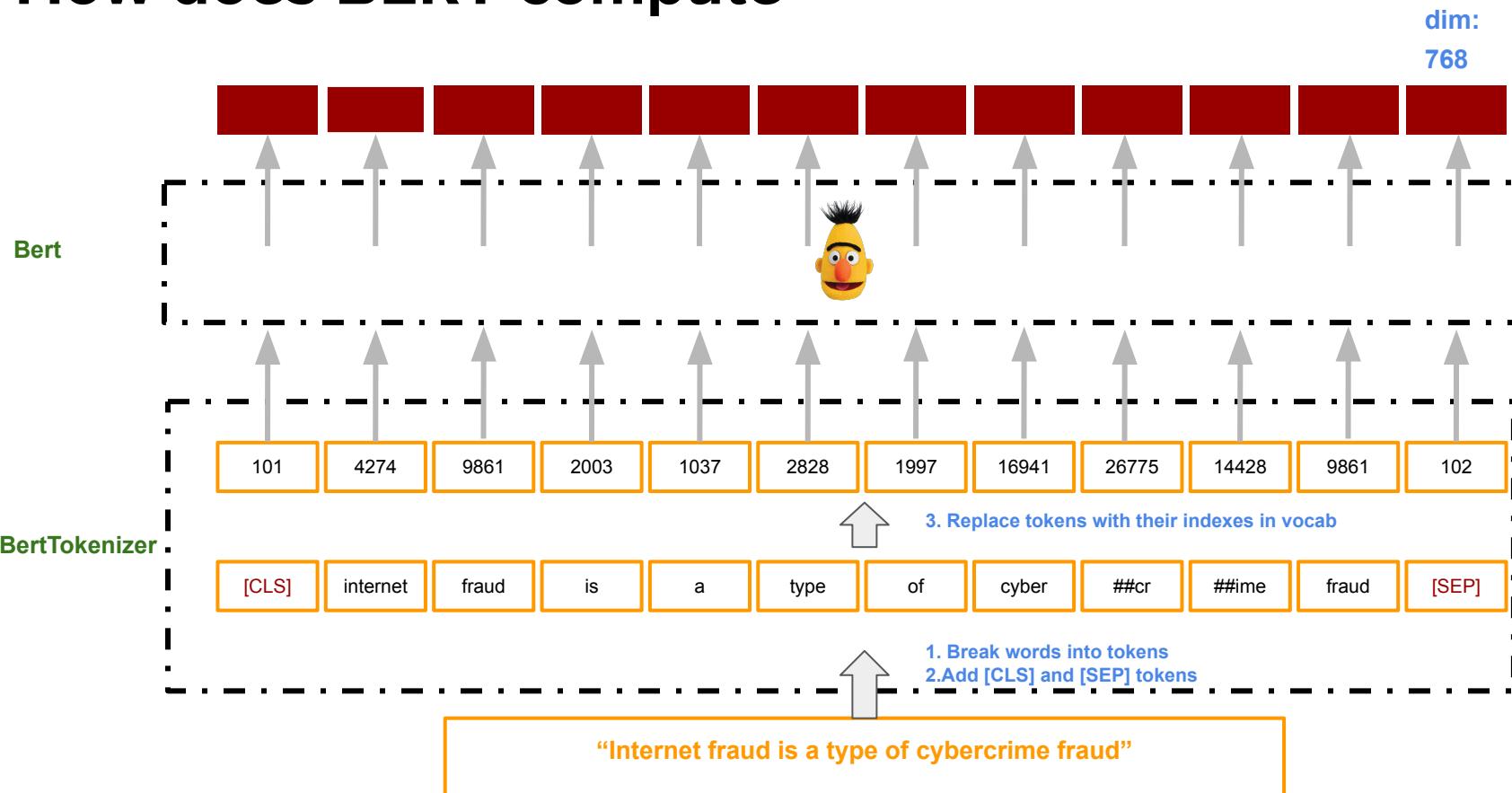


# BERT

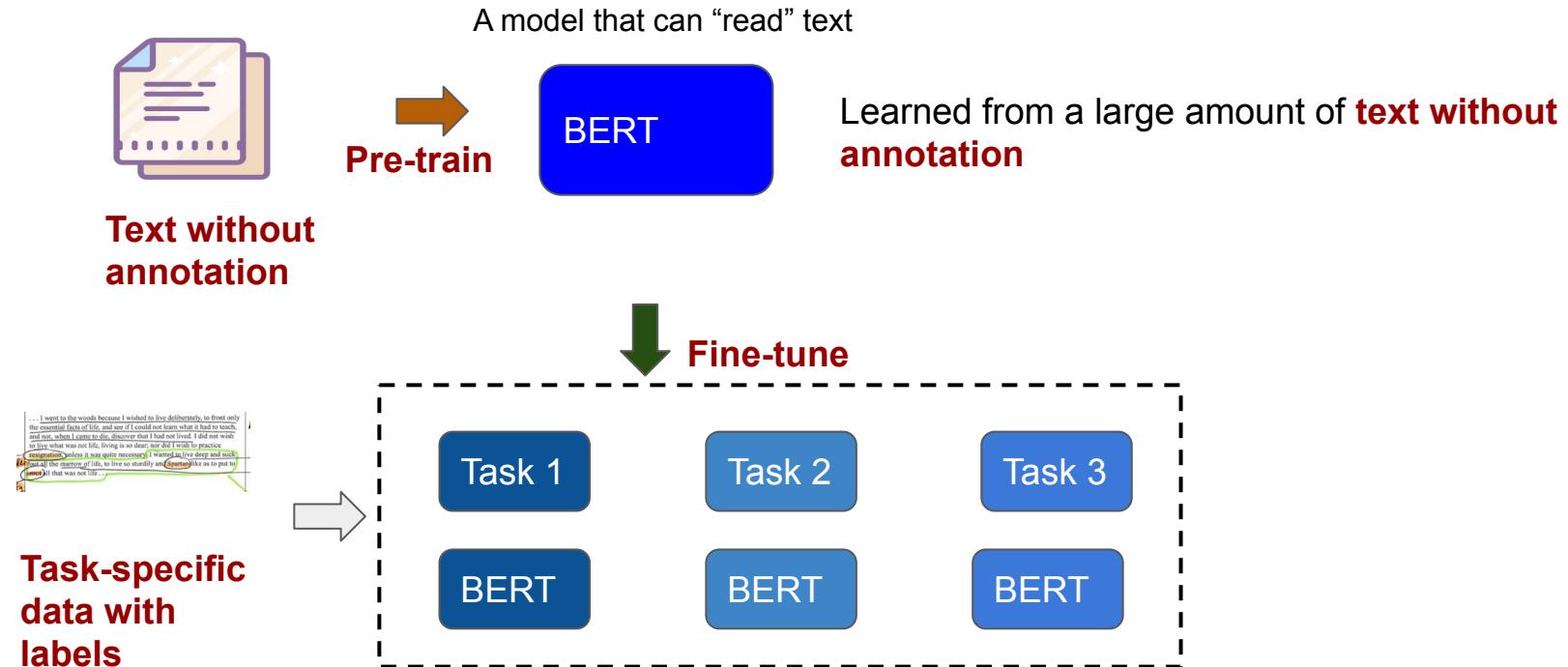
- Bidirectional Encoder Representations from Transformers (**BERT**)
- BERT: Encoder of Transformer,



# How does BERT compute



# How to use BERT



# How to Pre-Train



Yann LeCun

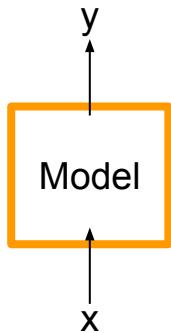
2019年4月30日 · 🌎

...

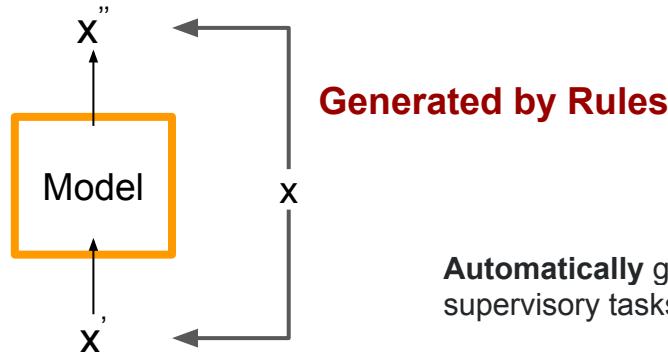
The answer is **self-supervised learning**.

I now call it "self-supervised learning", because "unsupervised" is both a loaded and confusing term.

In self-supervised learning, the system learns to predict part of its input from other parts of it input. In other words a portion of the input is used as a supervisory signal to a predictor fed with the remaining portion of the input.



Supervised



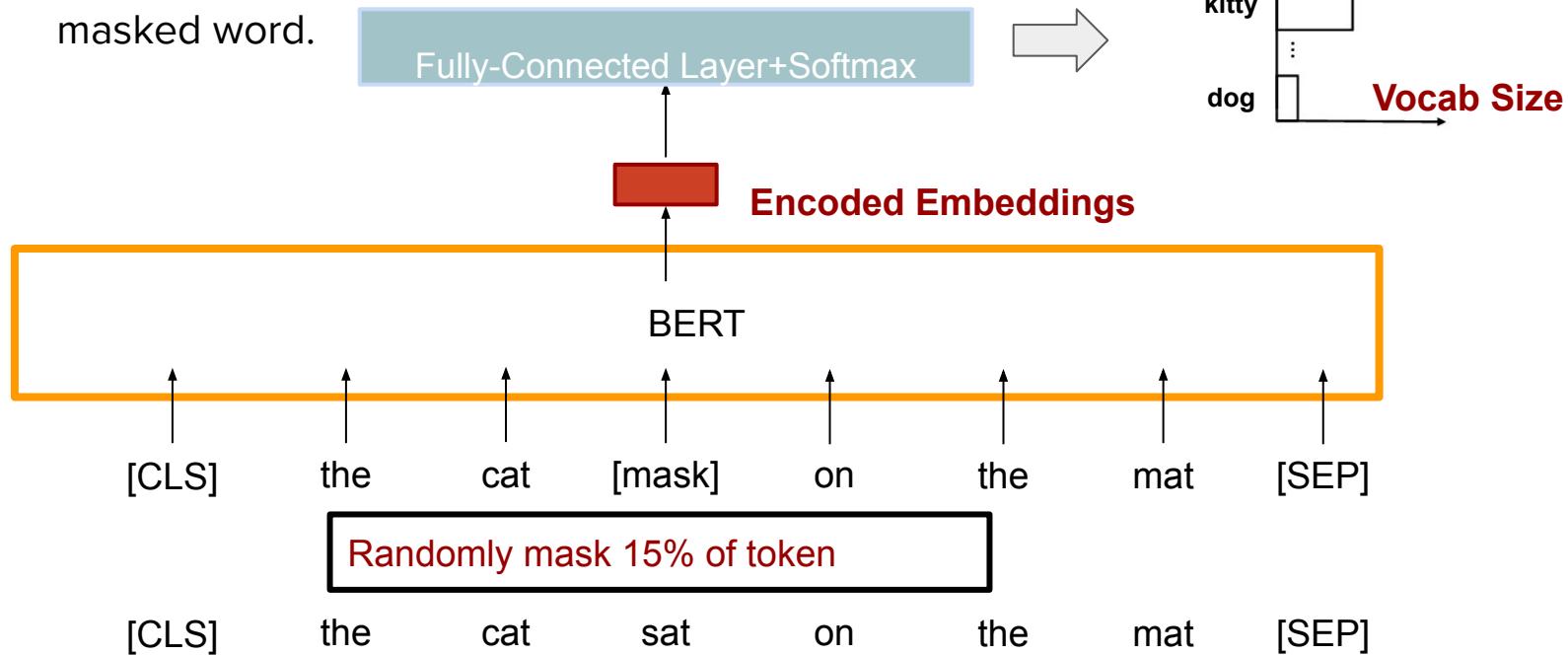
Self-Supervised

Automatically generate some kind of supervisory tasks

# Pre-training Task I: MLM

- **Masked Language Model**

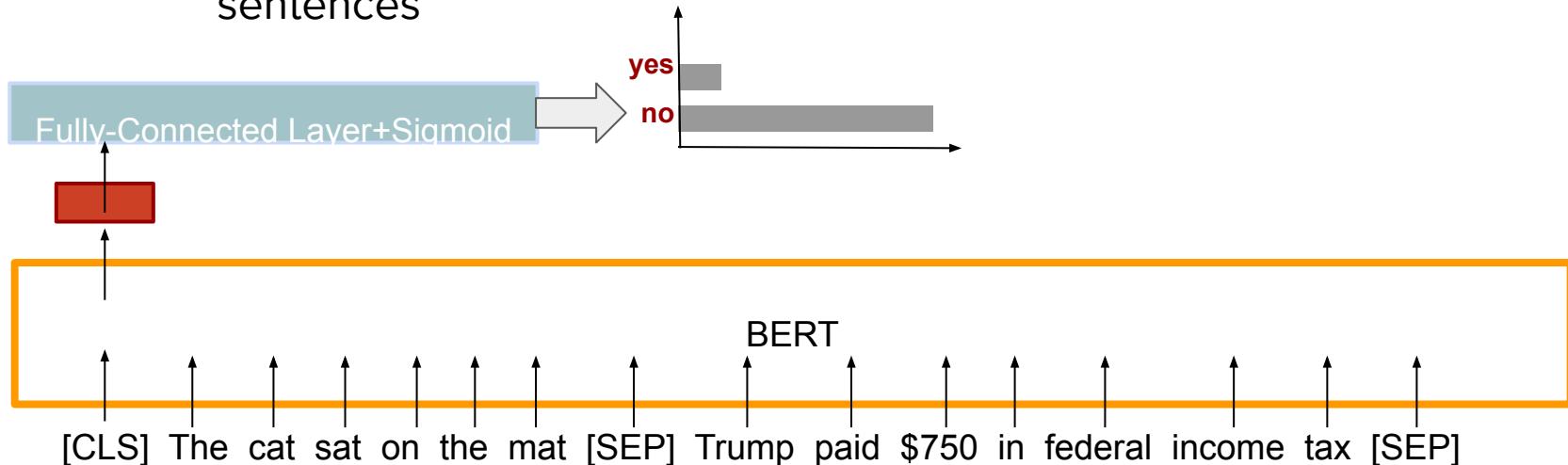
- Use the encoded embeddings of the masked word's to predict the masked word.



# Pre-training Task II: NSP

- **Next sentence prediction**

- Given two sentences A and B, is B likely to be the sentence followed by A?
- Make bert good at handling relationships between multiple sentences



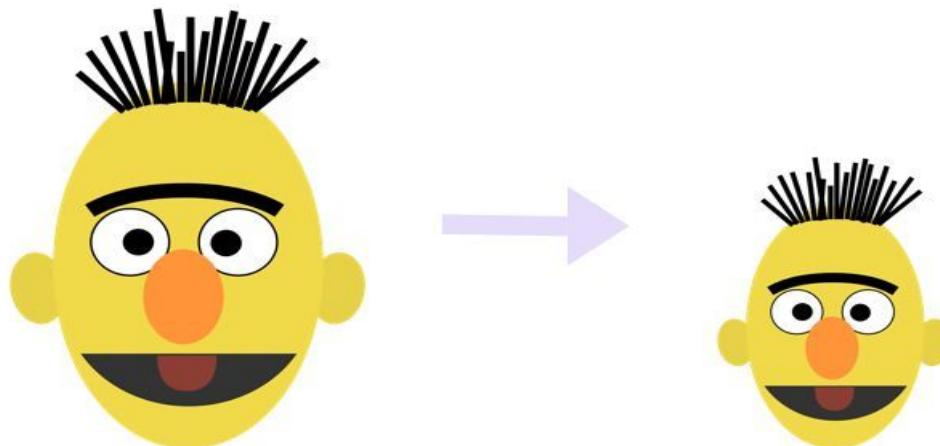
# Huge Model Size

Training of **BERT<sub>BASE</sub>** was performed on 4 Cloud TPUs in Pod configuration (16 TPU chips total).<sup>13</sup> Training of **BERT<sub>LARGE</sub>** was performed on 16 Cloud TPUs (64 TPU chips total). Each pre-training took 4 days to complete.

**12 attention heads  
110 million model parameters**

**16 attention heads  
345 million model parameters**

# Smaller Model



Published as a conference paper at ICLR 2020

---

## ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS

Zhenzhong Lan<sup>1</sup> Mingda Chen<sup>2\*</sup> Sebastian Goodman<sup>1</sup> Kevin Gimpel<sup>2</sup>  
Piyush Sharma<sup>1</sup> Radu Soricut<sup>1</sup>

---

**DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter**

---

Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF  
Hugging Face  
{victor,lysandre,julien,thomas}@huggingface.co

Abstract

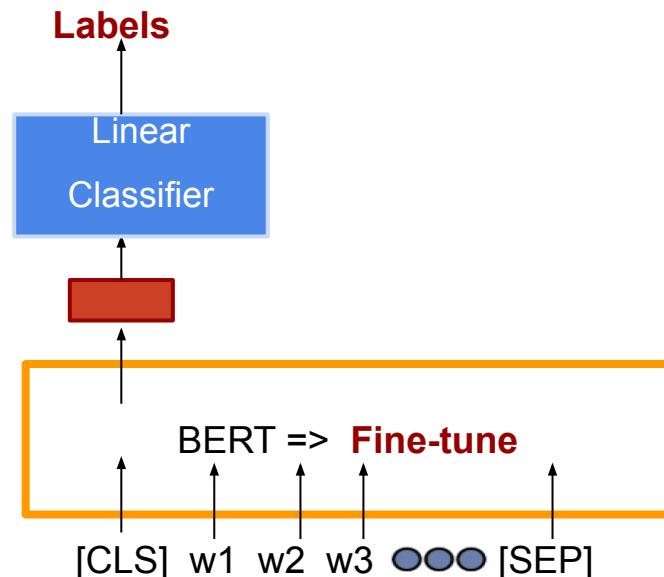
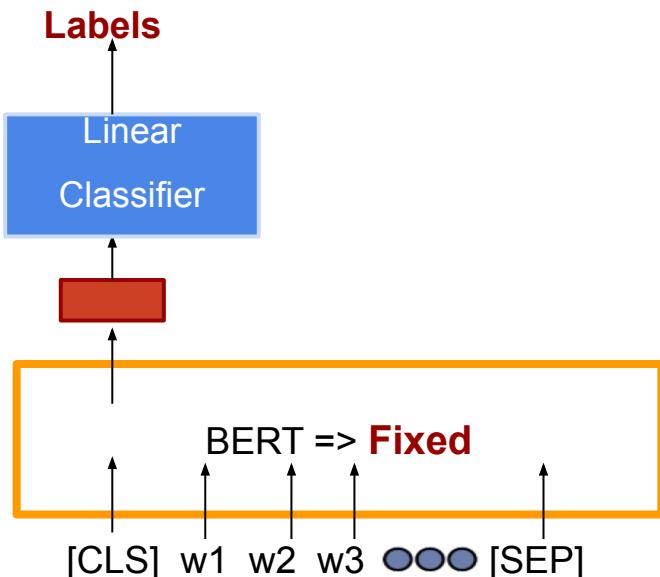
Good summary:

<http://mitchgordon.me/machine/learning/2019/11/18/all-the-ways-to-compress-BERT.html>

# BERT Usage I

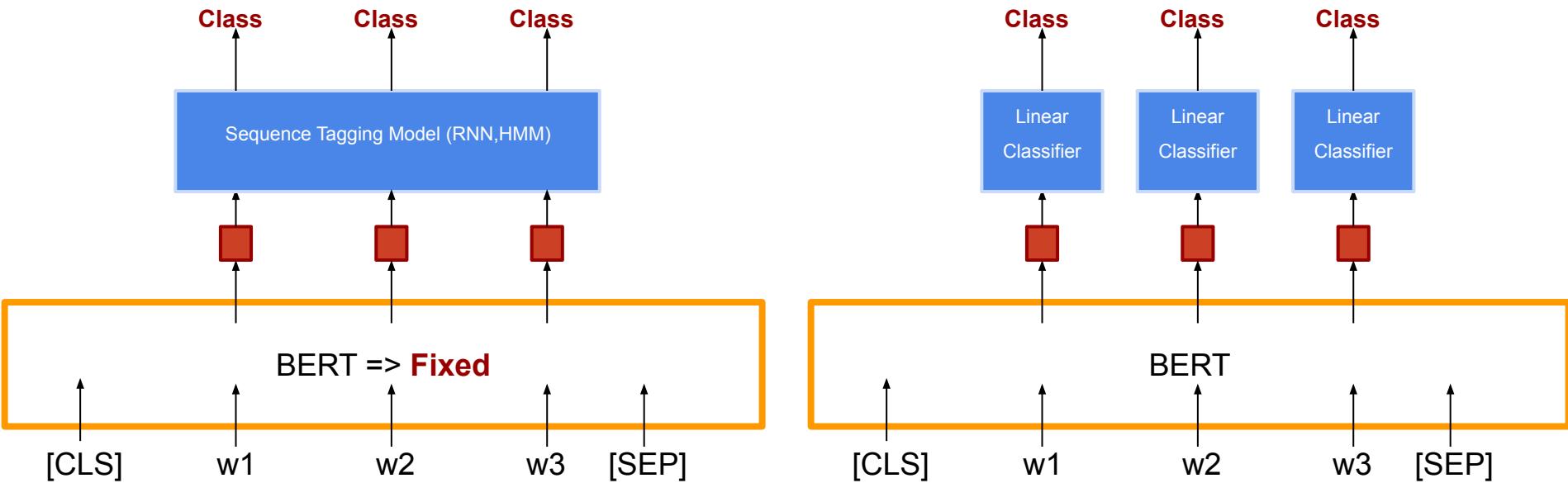
- **Input: Single Sentence Output: Class**

- Sentiment Analysis
- Document Classification



# BERT Usage II

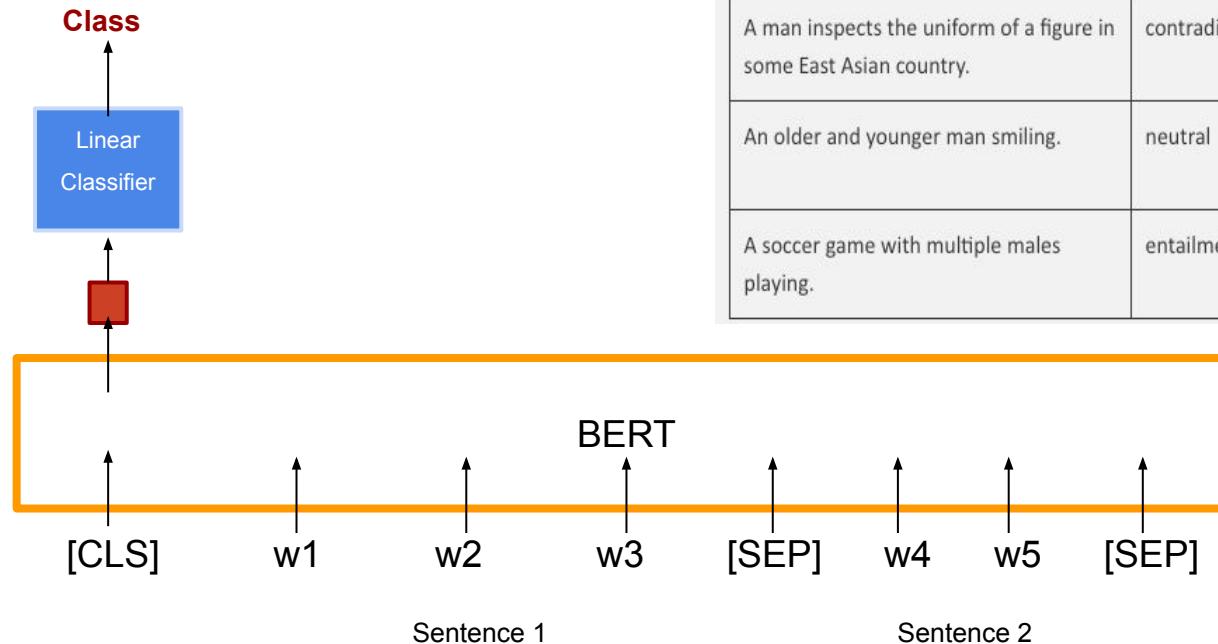
- **Input: Single Sentence Output: Class**
  - NER, POS Tagging



# BERT Usage III

- **Input: Two Sentences Output: Class**

- Natural Language Inference



Premise	Label	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	contradiction	The man is sleeping.
An older and younger man smiling.	neutral	Two men are smiling and laughing at the cats playing on the floor.
A soccer game with multiple males playing.	entailment	Some men are playing a sport.

# BERT Usage IV

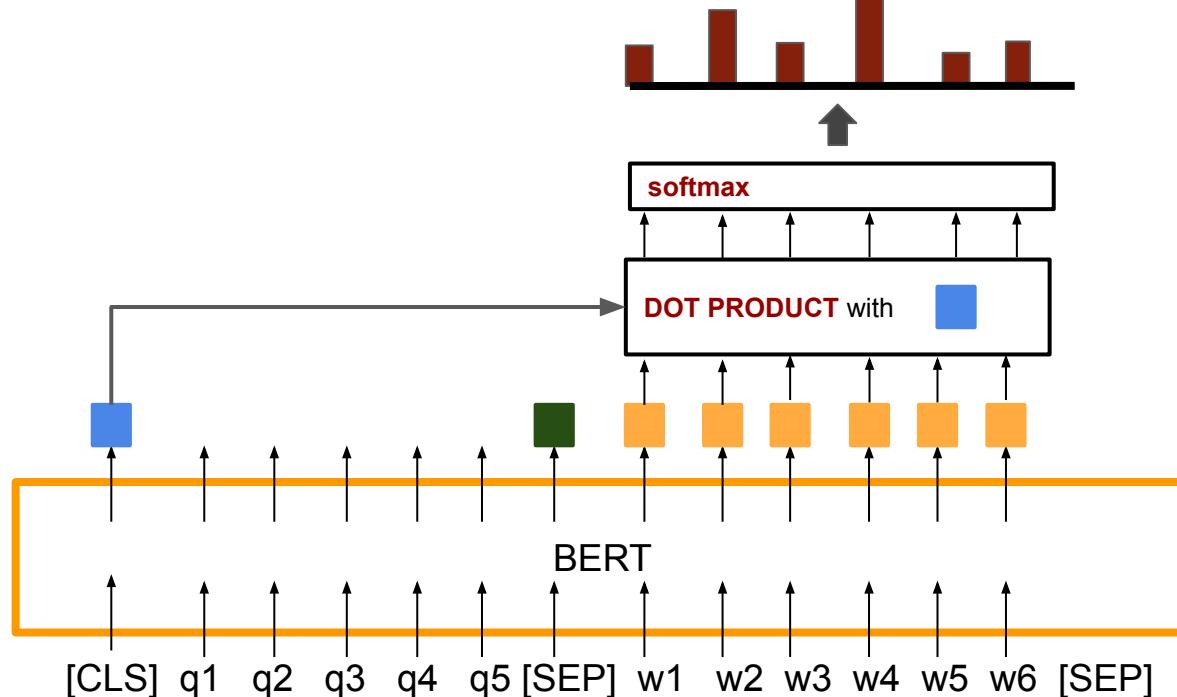
- Extraction-based Question Answering (SQuAD):
  - Input: two “sentences” (Question and Reference Text)
  - Output: start and end positions in Reference (Answer)

**Question:** How many parameters does BERT-large have?

**Reference Text:** BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

# BERT Usage IV

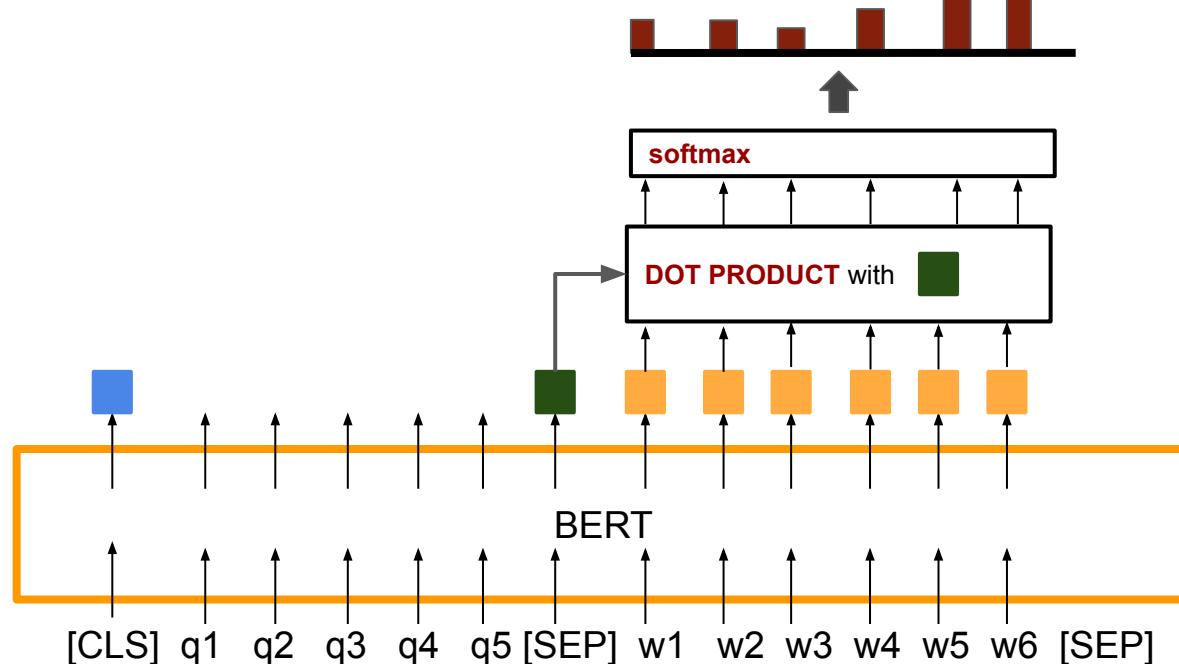
- Extraction-based Question Answering (SQuAD):
  - Question {q1,q2,q3,q4,q5} Reference Text{w1,w2,w3,w4,w5,w6}



The starting position for answer in reference is 4

# BERT Usage IV

- Extraction-based Question Answering (SQuAD):
  - Question {q1,q2,q3,q4,q5} Reference Text{w1,w2,w3,w4,w5,w6}

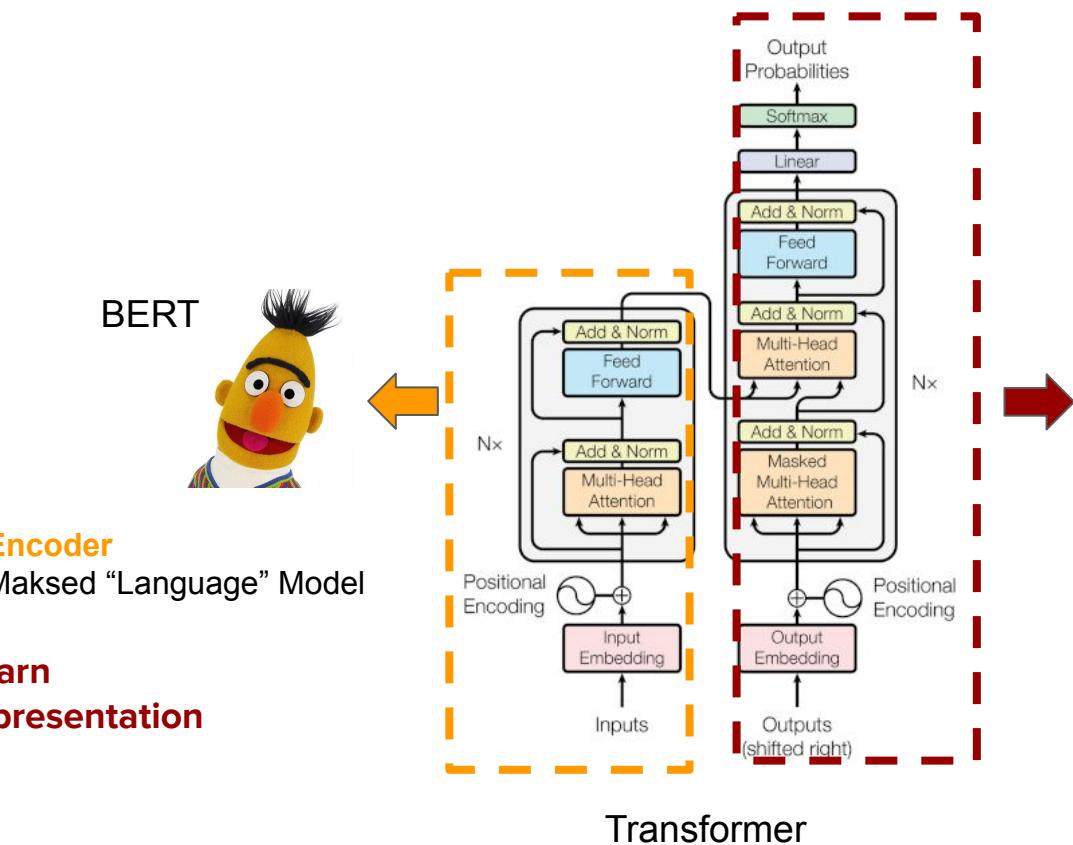


The starting position for answer in reference is 4

The ending position for answer in reference is 5

The answer is  
w4w5

# BERT, GPT and Transformers



GPT2  
GPT3



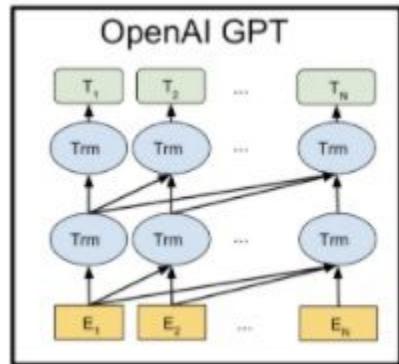
OpenAI

GPT-3, an autoregressive language model with 175 billion parameters

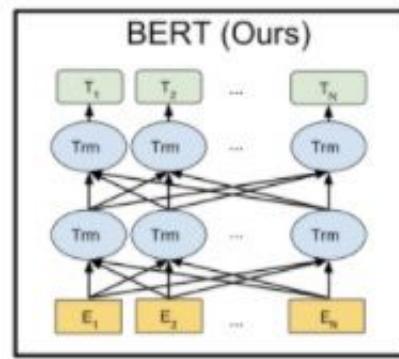
Decoder  
Language Model

Generate  
probabilities

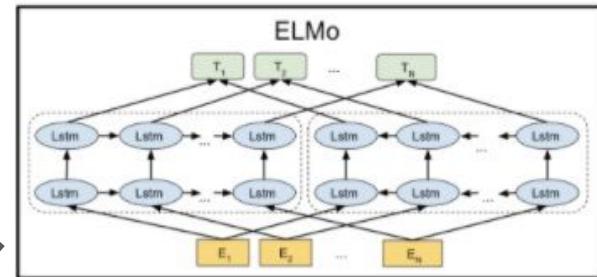
# BERT and other Pre-trained Models



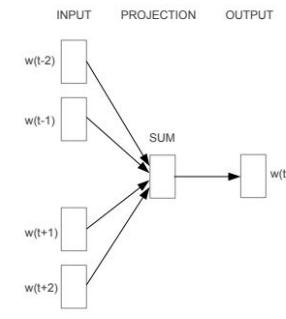
From  
single-directional to  
Bi-directional



From RNNs to  
transformers



From FCNs to  
transformers

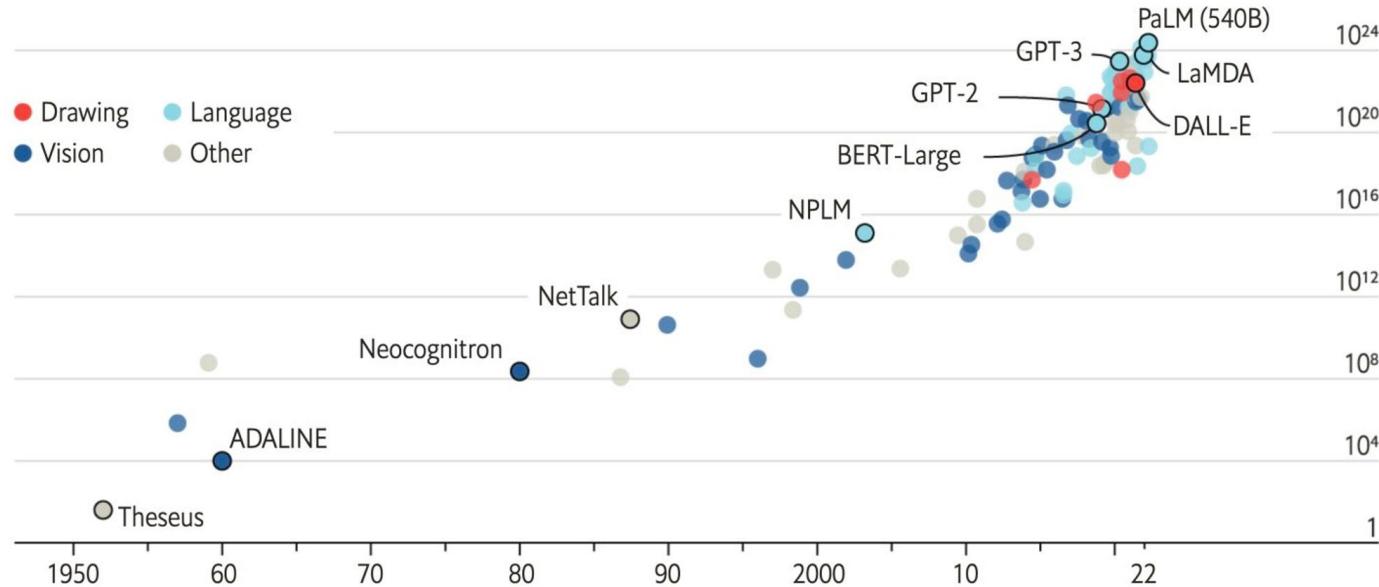


# Pre-training

## The blessings of scale

AI training runs, estimated computing resources used

Floating-point operations, selected systems, by type, log scale



Sources: "Compute trends across three eras of machine learning", by J. Sevilla et al., arXiv, 2022; Our World in Data

# Pre-training

GPT versions

	Architecture	Parameter count	Training data
GPT-1	12-level, 12-headed Transformer decoder (no encoder), followed by linear-softmax.	0.12 billion	BookCorpus: <sup>[8]</sup> 4.5 GB of text, from 7000 unpublished books of various genres.
GPT-2	GPT-1, but with modified normalization	1.5 billion	WebText: 40 GB of text, 8 million documents, from 45 million webpages upvoted on Reddit.
GPT-3	GPT-2, but with modification to allow larger scaling.	175 billion	570 GB plaintext, 0.4 trillion tokens. Mostly CommonCrawl, WebText, English Wikipedia, and two books corpora (Books1 and Books2).

[https://en.wikipedia.org/wiki/Generative\\_pre-trained\\_transformer](https://en.wikipedia.org/wiki/Generative_pre-trained_transformer)

GPT4 was released this week. The estimated model size is around 175B to 280B.

# Pre-training



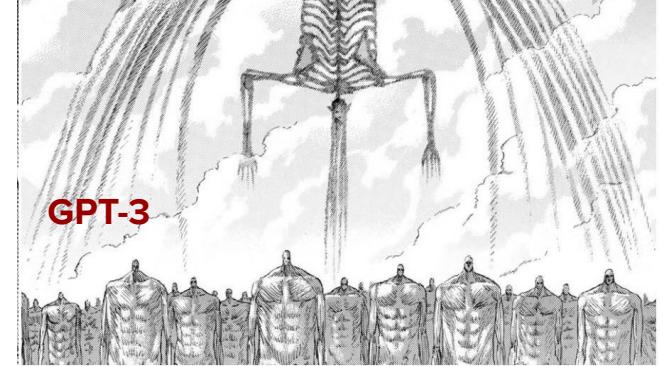
GPT-1

Levi 兵長 1.6m

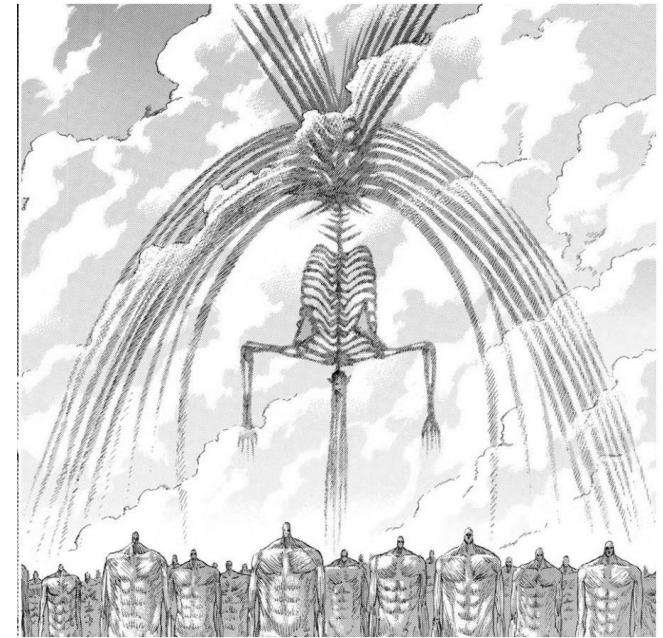


GPT-2

Beast Titan 獣の巨人 17m



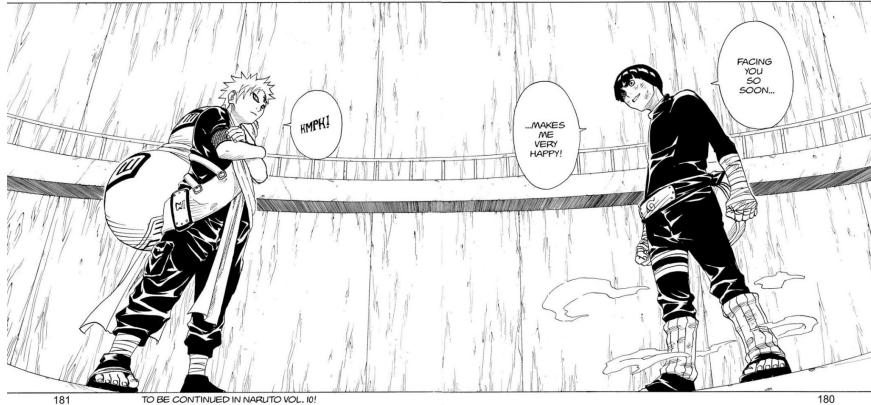
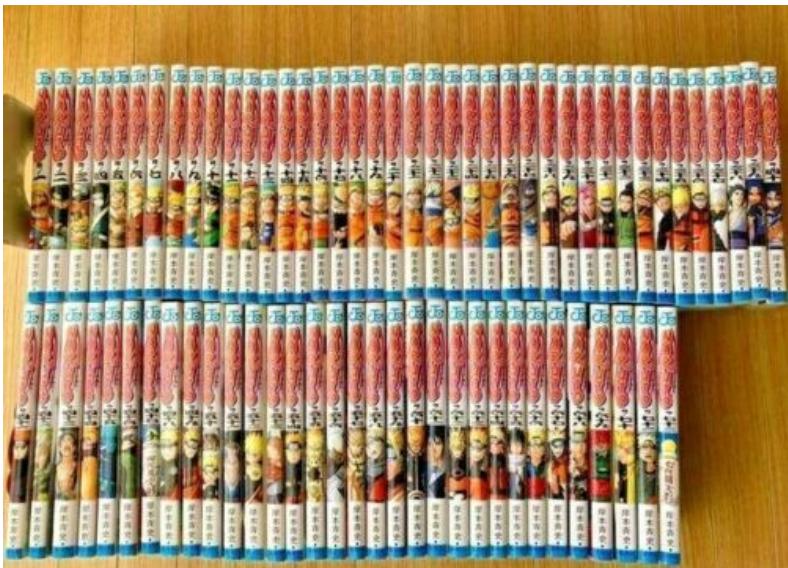
GPT-3



Two Stacked Eren Founding Titan 始祖の巨人  $2 \times 1000\text{m}$  <sup>89</sup>

# How large is 570GB

Read Naruto **270K** times



Only lines/text are counted.

# GPT3 is already powerful

- GPT3 can do everything that Chatgpt can do
  - <https://pub.towardsai.net/crazy-gpt-3-use-cases-232c22142044>
- Why not achieve the hype as ChatGPT?
  - It is a language model that is learned from all text across the open internet
  - GPT3 is a genius but do not know our humans' preference

## 6. From GPT to ChatGPT

# What is ChatGPT

- It is a language model
  - A mapping function: take a few words in, generate another word
  - Pre-trained by **large-scale** unannotated corpus
  - Fine-tuned by limited annotated dataset



What is business analytics



Business analytics is the practice of using data analysis and statistical methods to make informed business decisions. It involves the use of techniques such as data mining, data visualization, and predictive modeling to identify patterns and insights within data that can inform business strategies and actions.



Business analytics can help organizations improve their decision-making by providing data-driven insights into areas such as customer behavior, market trends, product performance, and operational efficiency. By analyzing data from various sources, such as customer surveys, social media, and sales reports, businesses can identify opportunities for growth, optimize their operations, and make more informed decisions.

Business analytics is a broad field that encompasses many different techniques and approaches, including descriptive analytics, which focuses on summarizing and visualizing data; predictive analytics, which uses statistical modeling and machine learning to make predictions about future outcomes; and prescriptive analytics, which suggests optimal courses of action based on data analysis.

Overall, business analytics is a powerful tool for businesses of all sizes and industries, and can help organizations make data-driven decisions that can lead to greater success and



Business analytics is the practice of using data analysis and statistical methods to make informed business decisions. It involves the use of techniques such as data mining, data visualization, and predictive modeling to identify patterns and insights within data that can inform business strategies and actions.

Business analytics can help organizations improve their decision-making by providing data-driven insights into areas such as customer behavior, market trends, product performance, and operational efficiency. By analyzing data from various sources, such as customer surveys, social media, and sales reports, businesses can identify opportunities for growth, optimize their operations, and make more informed decisions.

Business analytics is a broad field that encompasses many different techniques and approaches, including descriptive analytics, which focuses on summarizing and visualizing data; predictive analytics, which uses statistical modeling and machine learning to make predictions about future outcomes; and prescriptive analytics, which suggests optimal courses of action based on data analysis.

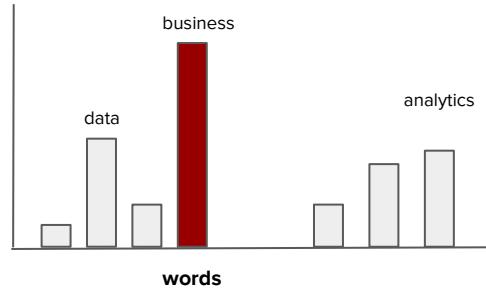
Overall, business analytics is a powerful tool for businesses of all sizes and industries, and

# Language model

What is business analytics?

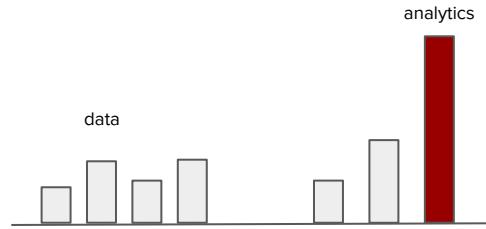


$\text{prob(next token} \mid \text{input text})$



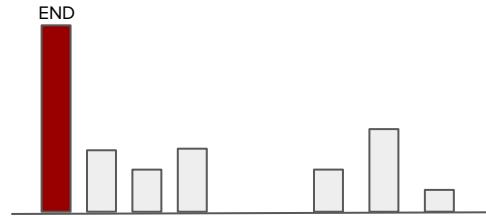
Business

What is business analytics? Business



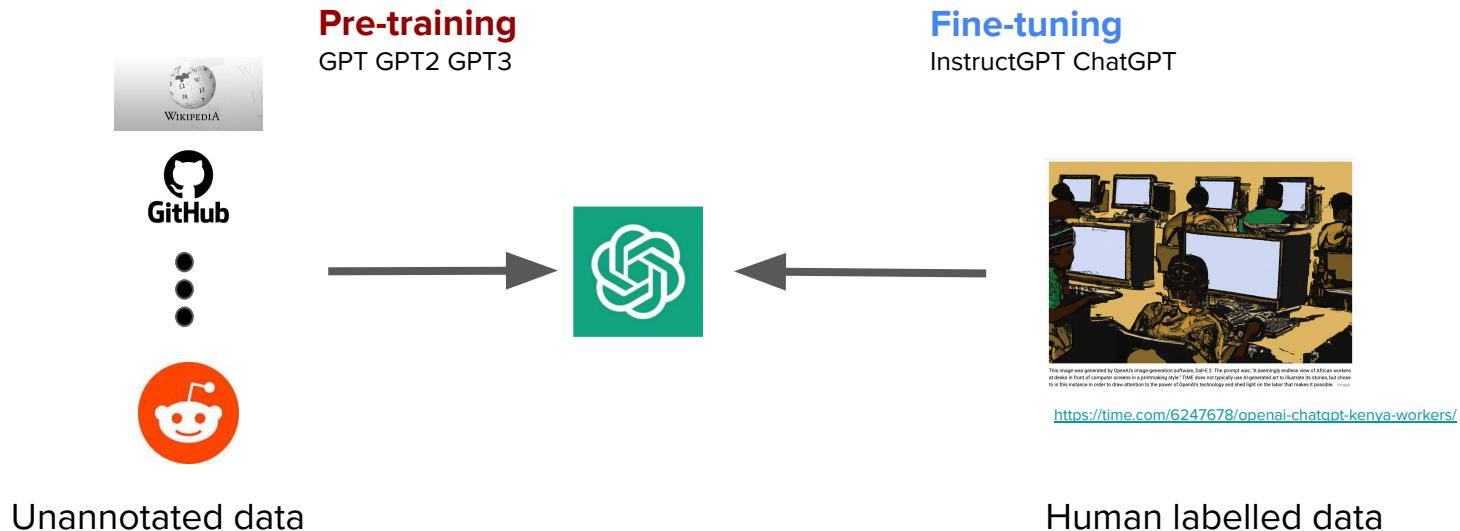
analytics

What is business analytics? Business analytics is the practice.....



END

# How to train ChatGPT



# GPT3 is already powerful

- GPT3 can do everything that Chatgpt can do
  - <https://pub.towardsai.net/crazy-gpt-3-use-cases-232c22142044>
- Why not achieve the hype as ChatGPT?
  - It is a language model that is learned from all text across the open internet
  - GPT3 is a genius but do not know our humans' preference

# GPT3 sometimes is cute

What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```



The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function.



What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```



source: <https://arxiv.org/pdf/2203.02155.pdf>

# GPT3 sometimes is cute

What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```



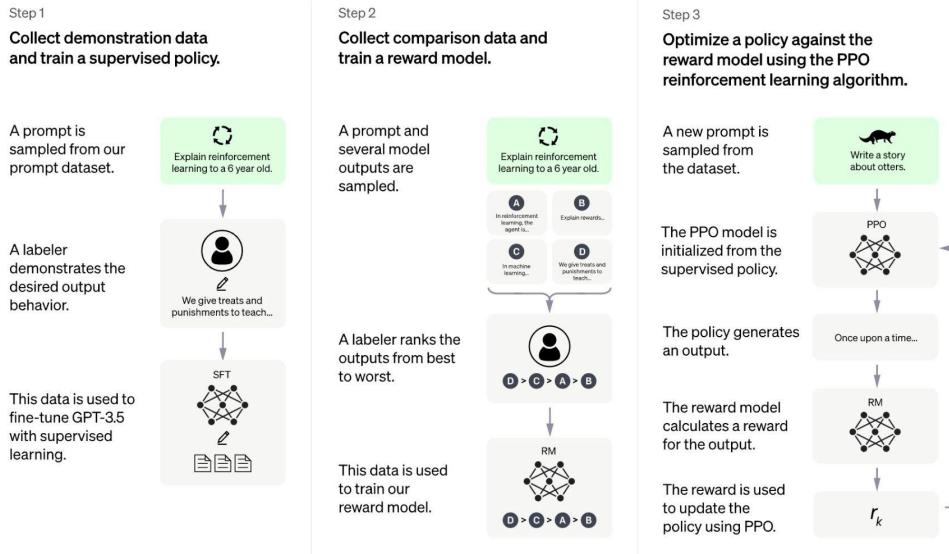
- A. to store the value of C[0]
- B. to store the value of C[1]
- C. to store the value of C[i]
- D. to store the value of C[i - 1]



source: <https://arxiv.org/pdf/2203.02155.pdf>

# How to inject human preference

- This is how ChatGPT is different from GPT3



## Training language models to follow instructions with human feedback

Long Ouyang\* Jeff Wu\* Xu Jiang\* Diogo Almeida\* Carroll L. Wainwright\*

Pamela Mishkin\* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray

John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens

Amanda Askell<sup>†</sup> Peter Welinder Paul Christiano<sup>†</sup>

Jan Leike\* Ryan Lowe\*

OpenAI

## Abstract

## Methods

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

source: <https://openai.com/blog/chatgpt>

ChatGPT should be quite close to InstructGPT in terms of implementation

# Start with Prompts

- Prompts: query sent to GPT models
- Prompts are generated in the following ways:
  - Plain: ask the labelers to come up with an arbitrary task
  - Few-shot: ask the labelers to come up with an instruction
    - Like any coding/programming related questions
  - User-based: collected use-cases from OpenAI API users. And labelers are asked to come up with related prompts

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms" →	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?" →	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?" →	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021



**Try to increase the diversity and coverage of all prompts**

# Step 1: Supervised fine-tuning (InstructGPT)

- Prepare SFT dataset
  - It only has 13K training prompts with labeler demonstration
  - The ground-truth responses are provided directly
- Fine-tune the GPT model using the SFT dataset
  - Training target is the same as the pre-training: next word prediction

It is too **expensive** to prepare prompts in the above way. Sometime, it is even not possible. Think about some prompts such as “write a song for my best friend who is having the kaggle competition”

# Step 1: Supervised fine-tuning (InstructGPT)

- SFT dataset
  - It only has 13K training prompts with labeler demonstration
  - The ground-truth responses are provided directly
- Fine-tune the GPT model using the SFT dataset
  - Training target is the same as the pre-training: next word prediction

It is too **expensive** to prepare prompts in the above way. Sometime, it is even not possible. Think about some prompts such as “write a song for my best friend who is having the kaggle competition”

**We can not write songs but we can pick a better song.**

# Step 2: Reward model (InstructGPT)

- Prepare RM dataset (33k prompts)
  - Given prompts and multiple model outputs, labelers are asked to give ranking
- Train a Reward model
  - Take a prompt and a response, and output a scalar reward
  - Loss function for the reward model:

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D} [\log (\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))]$$

where  $r_\theta(x, y)$  is the scalar output of the reward model for prompt  $x$  and completion  $y$  with parameters  $\theta$ ,  $y_w$  is the preferred completion out of the pair of  $y_w$  and  $y_l$ , and  $D$  is the dataset of human comparisons.

The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function.

(1)

What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```

- A. to store the value of C[0]  
B. to store the value of C[1]  
C. to store the value of C[i]  
D. to store the value of C[i - 1]

# Step 3: Reinforcement learning (InstructGPT)

- Prepare PPO dataset (31k prompts)
  - Selected prompts and RM model from Step 2
- Fine-tune SFT model using PPO algorithm (one of Reinforcement Learning algorithms)
  - The environment here is: prompt is randomly presented and the SFT model is expected to generate a prompt
  - Given the prompt and response, RM model will output a reward value
  - SFT model is trained to achieve a higher reward value

**From these three steps, we kind of enforce GPT model to learn our humans' preference**

# GPT4

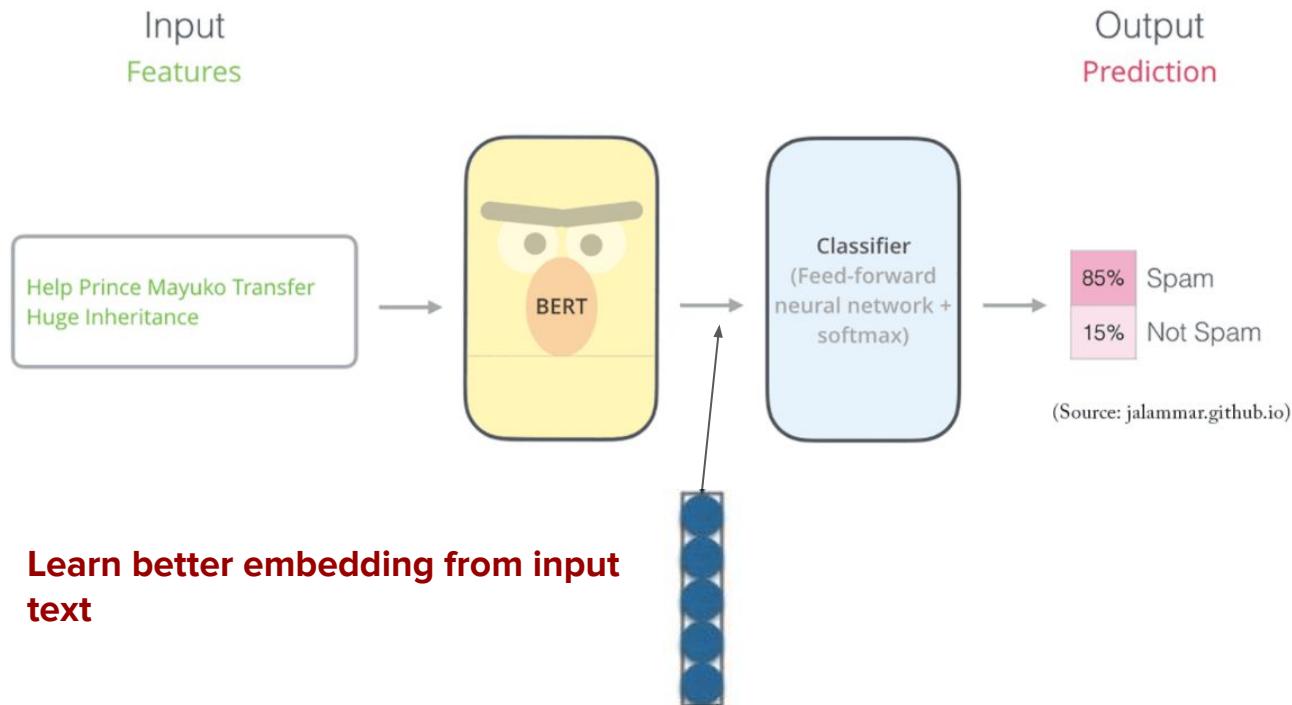
- Accept visual inputs now
  - <https://www.youtube.com/watch?v=outcGtbnMuQ&t=971s>
- Can handle up to 25 K words (3k for ChatGPT)

## Abstract

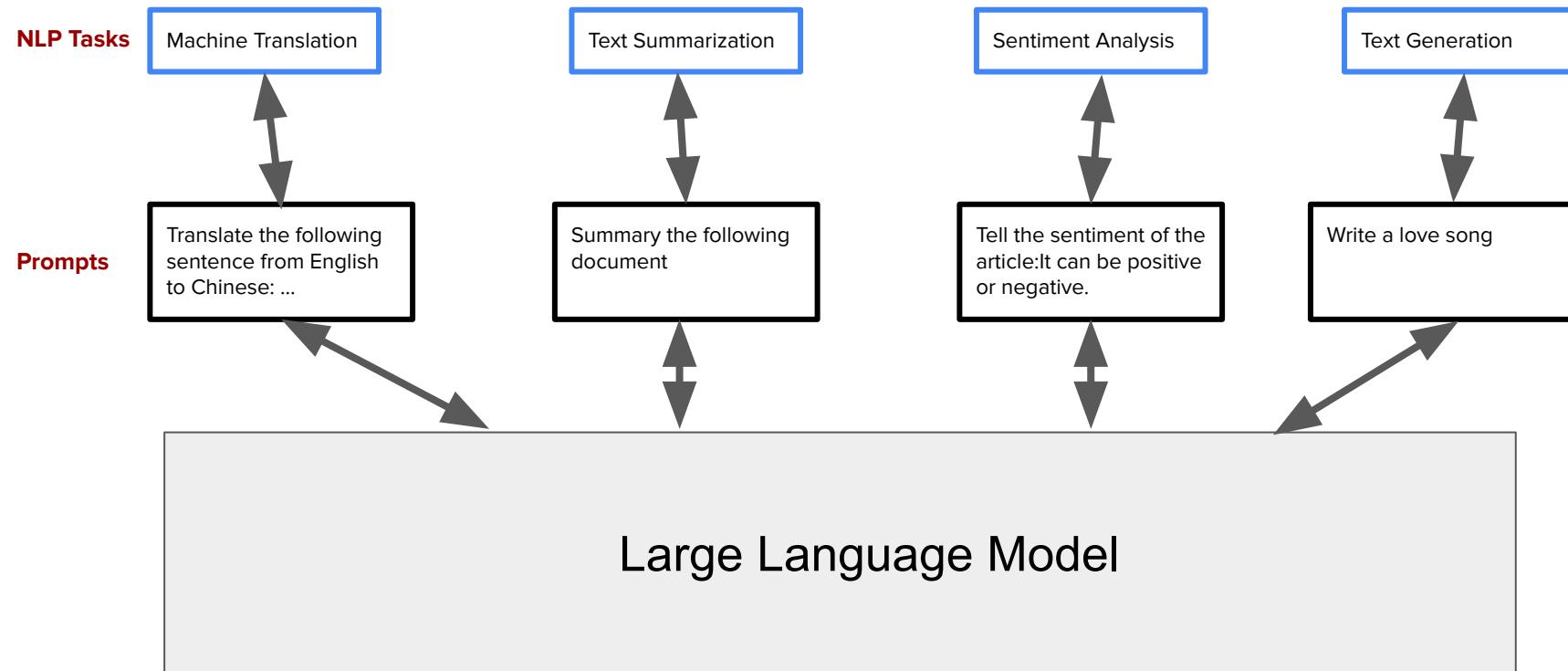
We report the development of GPT-4, a large-scale, multimodal model which can accept image and text inputs and produce text outputs. While less capable than humans in many real-world scenarios, GPT-4 exhibits human-level performance on various professional and academic benchmarks, including passing a simulated bar exam with a score around the top 10% of test takers. GPT-4 is a Transformer-based model pre-trained to predict the next token in a document. The post-training alignment process results in improved performance on measures of factuality and adherence to desired behavior. A core component of this project was developing infrastructure and optimization methods that behave predictably across a wide range of scales. This allowed us to accurately predict some aspects of GPT-4's performance based on models trained with no more than 1/1,000th the compute of GPT-4.

## 7. Fine-tuning vs Prompt Engineering

# Bert + Fine-tuning



# GPT + Prompt Engineering



# Fine-tuning vs Prompt

The model is fine-tuned via a large corpus of example tasks

machine learning → 机器学习

data science → 数据科学

fraud detection → 欺诈检测

Gradients are updated

artificial intelligence → ?

Prompt (few-shot learning here)

Translate English to Chinese

machine learning -> 机器学习

data science -> 数据科学

fraud detection -> 欺诈检测

artificial intelligence ->

No gradient updates are performed

The prompt consists

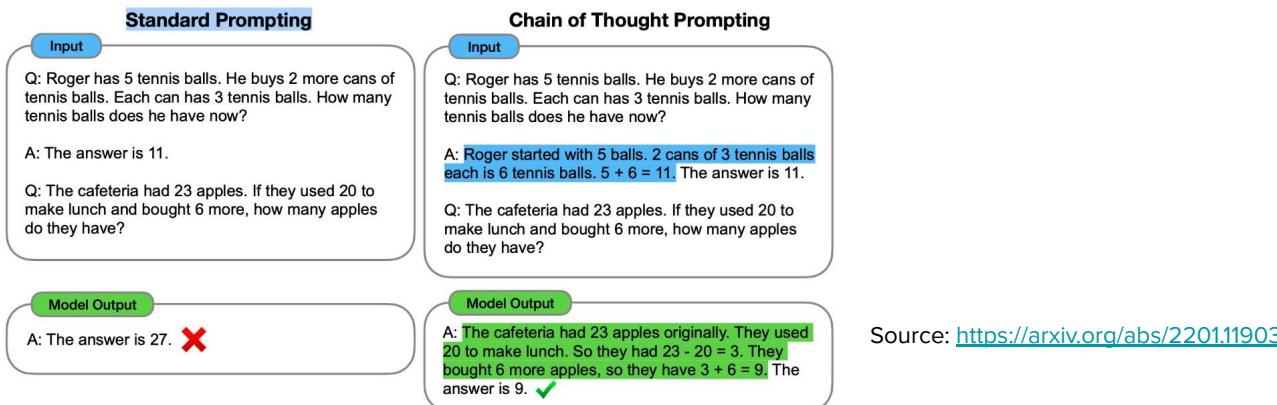
1. Task description

2. Examples

3. Query

# Prompting

- Zero-shot Prompting
  - No examples are given in prompt
- Few-shot Prompting
  - A few shot examples of tasks are provided
- Chain of Thoughts Prompting
  - Examples with the reasoning processes are improved.



# Prompt Engineer?

AI generation, wherever its used, seems to lower the barriers to entry to any profession. Artists and copywriters are most obviously at risk, but software development may be the next big field to have [AI take over](#).

AI research company Anthropic recently put out a job listing for a 'prompt-engineer' and prompt librarian. The base salary? An impressive [\\$250k-\\$335k](#).

Source: <https://www.efinancialcareers.sg/news/2023/01/chat-gpt-banking>

Want to be prompt Engineer?

[https://www.reddit.com/r/StableDiffusion/comments/xcrm4d/useful\\_prompt\\_engineering\\_tools\\_and\\_resources/](https://www.reddit.com/r/StableDiffusion/comments/xcrm4d/useful_prompt_engineering_tools_and_resources/)

<https://learnprompting.org/>

# Some fun from ChatGPT Prompts



reminder me lyrics of eric cartman's song about kyle's mom



Sure, here are the lyrics to Eric Cartman's song "Kyle's Mom Is a Big Fat B\*\*\*\*" from the TV show South Park:

Kyle's Mom is a big fat b\*\*\*\*,  
She's the biggest b\*\*\*\* in the whole wide world.  
She's a stupid b\*\*\*\*, if there ever was a b\*\*\*\*,  
She's a b\*\*\*\* to all the boys and girls.

On Monday, she's a b\*\*\*\*,  
On Tuesday, she's a b\*\*\*\*,  
On Wednesday, through Saturday, she's a b\*\*\*\*,  
Then on Sunday, just to be different,  
She's a super King Kamehameha b\*\*\*\*.

Have a great day!



so you will not generate any curse words no matter what? in any context?



As an AI language model, my responses are based on the training data and guidelines that were used to develop my language capabilities. I am designed to generate responses that are informative, helpful, and respectful, and do not contain any profanity, offensive or discriminatory language, or inappropriate content. My purpose is to assist and provide information, and I am not programmed to generate or condone any kind of harmful or inappropriate language.

## Next Class: Model Evaluation