Applied Machine Learning for Business Analytics

Lecture 8: Frontiers in NLP

Lecturer: Zhao Rui

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 (a small bite on GPT4)
7. Fine-tuning vs Prompt Engineering

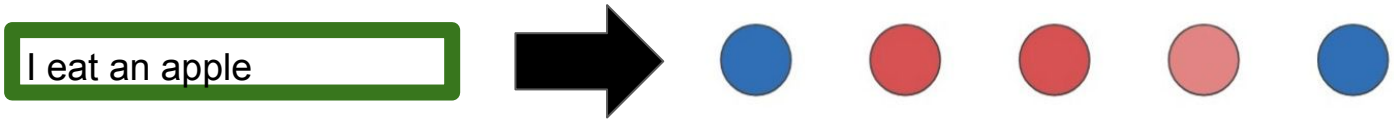
1. Representation Learning

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.



Text

Numerical Representation

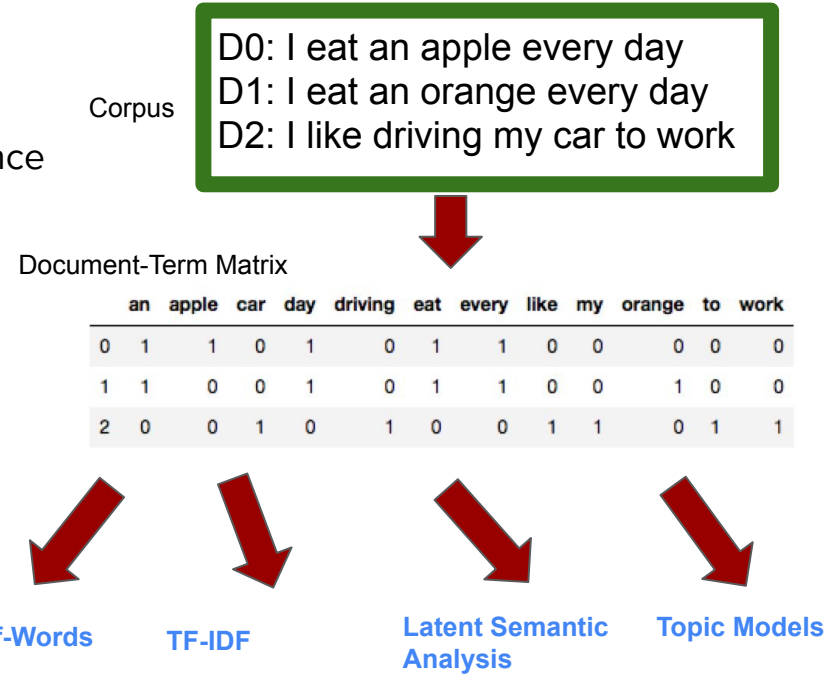
The learned representation should capture high-level semantic and syntactic information.

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



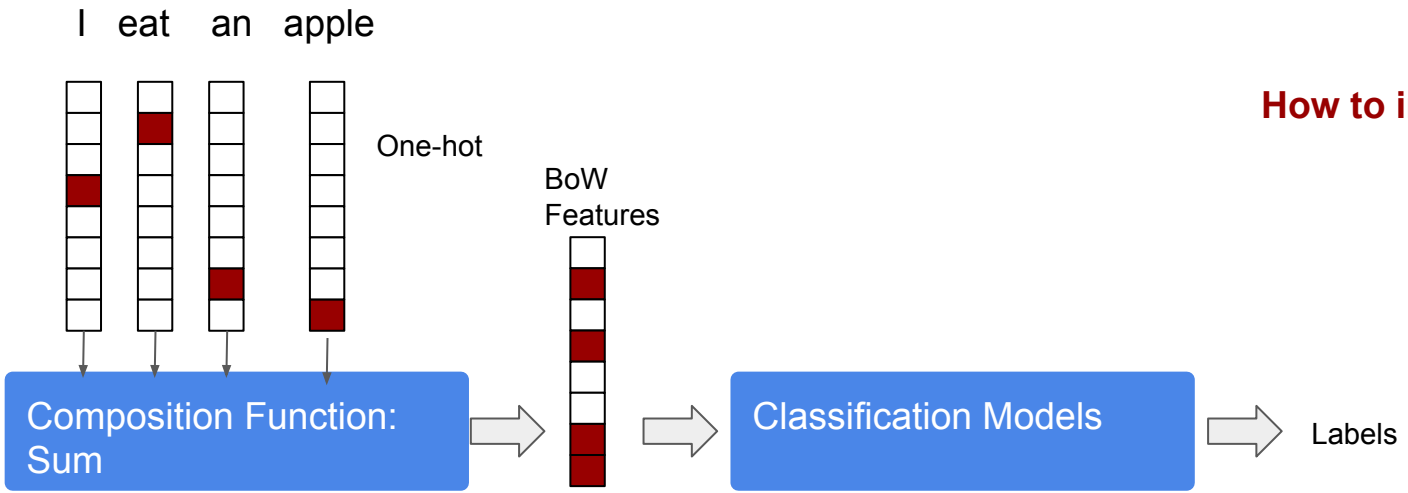
Limitations of document-Term matrix

- 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	0
1	1	0	0	1	0	1	1	0	0	1	0	0
2	0	0	1	0	1	0	0	1	1	0	1	1

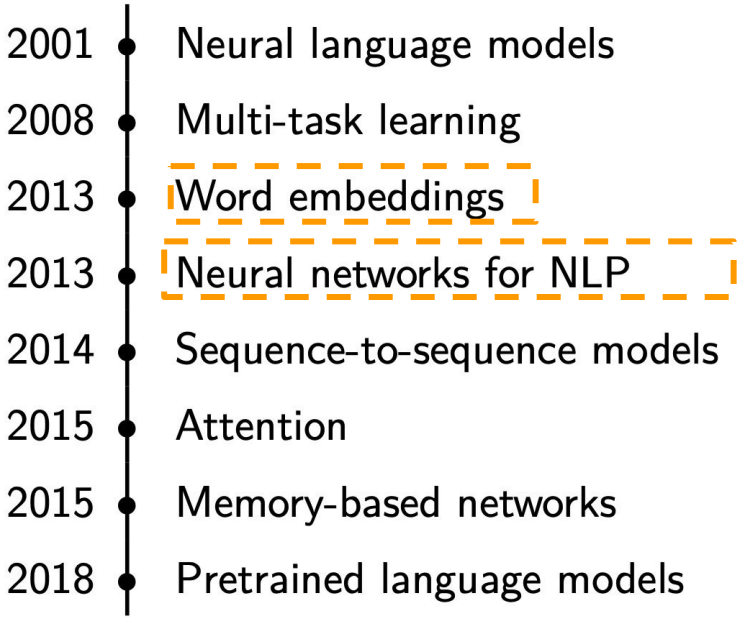
A new perspective on BoW

- 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.



How to improve?

Neural NLP



https://www.kamperh.com/slides/ruder+kamper_indaba2018_talk.pdf

2. Word Embeddings

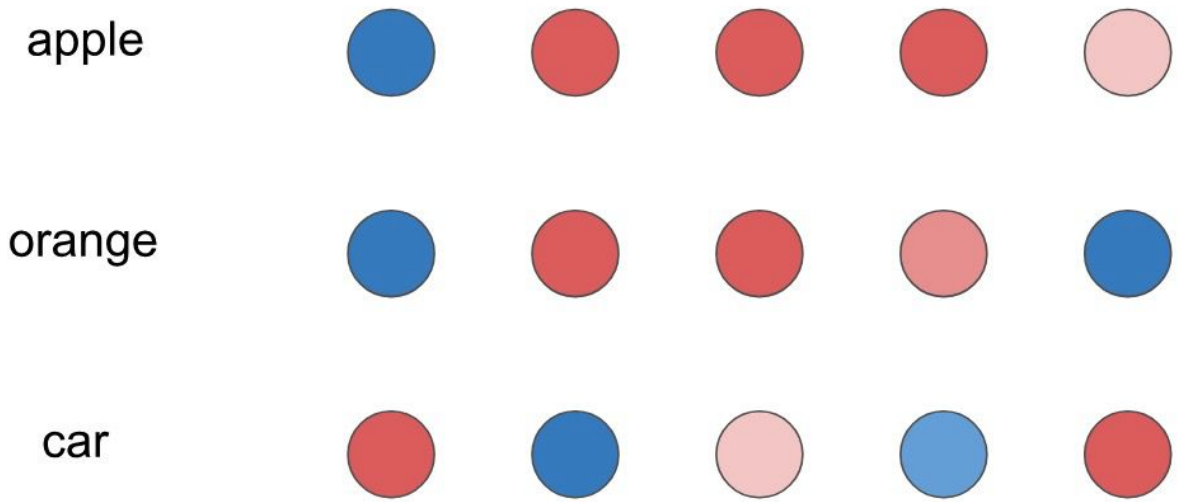
Word representation

- 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 0 0]
orange	[0 0 0 0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 ... 0 0 0 0 0 0]
car	[0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]

Distributed representation

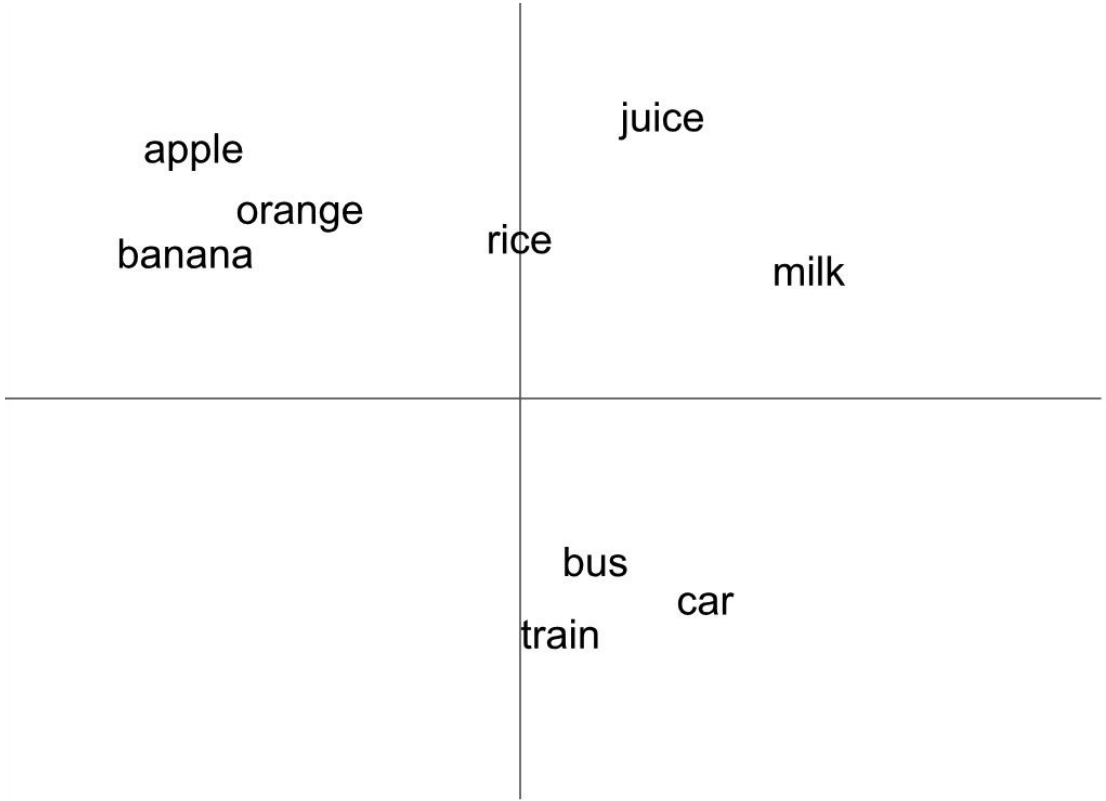
- Words should be encoded into a low-dimensional and dense vector



Word vectors

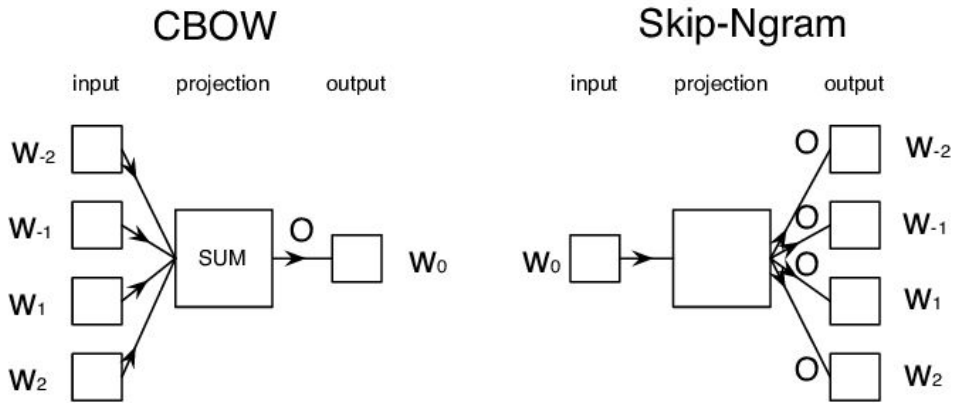
Project word vectors in a two-dimensional space. And visualize them!

Similar words are close to each other.

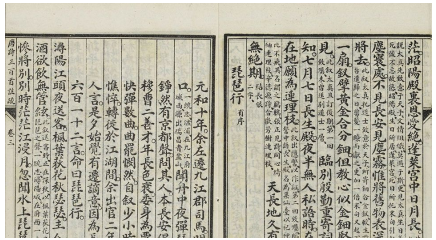


Word2Vec

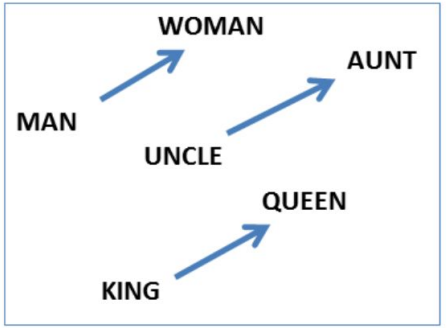
- A method of computing vector representation of words developed by Google.
- Open-source version of Word2Vec hosted by Google (in C)
- Train a simple neural network with a single hidden layer to perform word prediction tasks.
- Two structures proposed Continuous Bag of Words (CBoW) vs Skip-Gram



Word2Vec as BlackBox



input, output



Corpus **Word2Vec Tool** **Word Embeddings**

A Good Visualization for Word2Vec

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

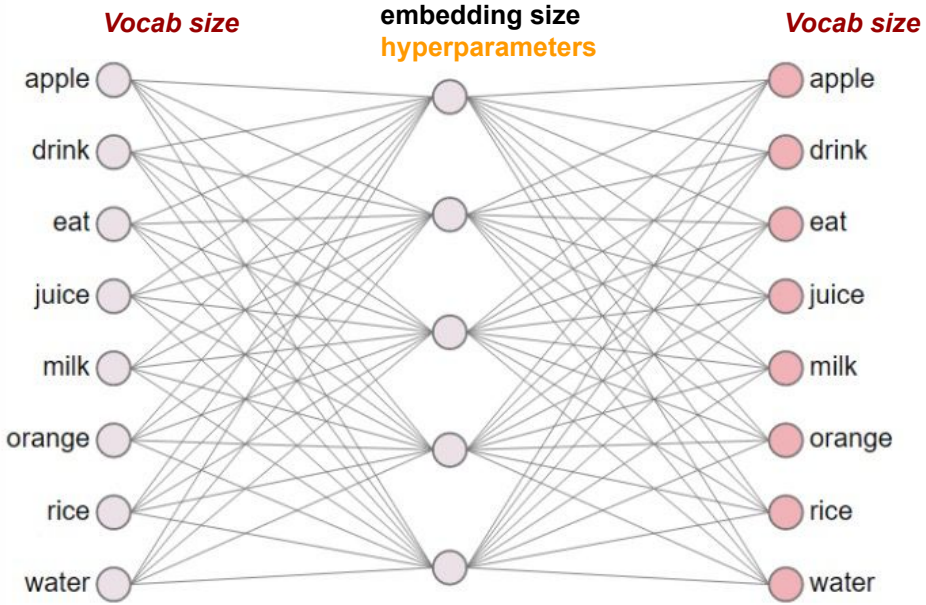
Target

- 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.

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

Neural network parameters that will be optimized

Model architecture



Structure Highlights:

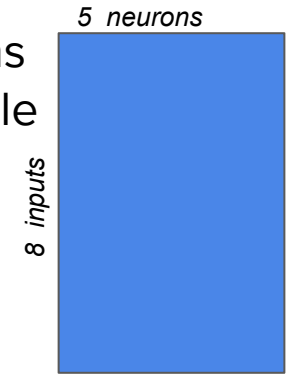
- input layer
  - one-hot vector
- hidden layer
  - linear (identity)
- output layer
  - softmax

From Xin Rong 2016

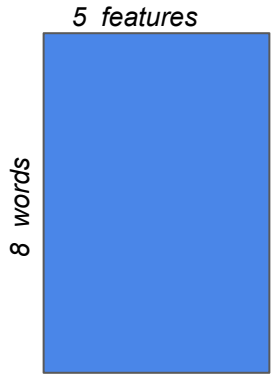
Hidden layer

- **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



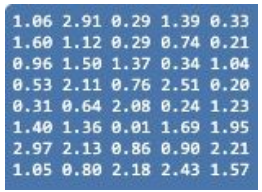
Word Vector Look Up Table



One-hot vector

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

Index of eat



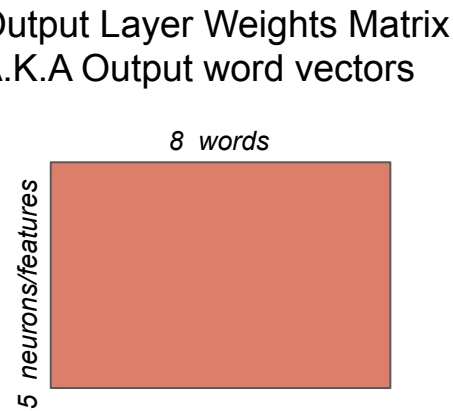
Word vector for “eat”

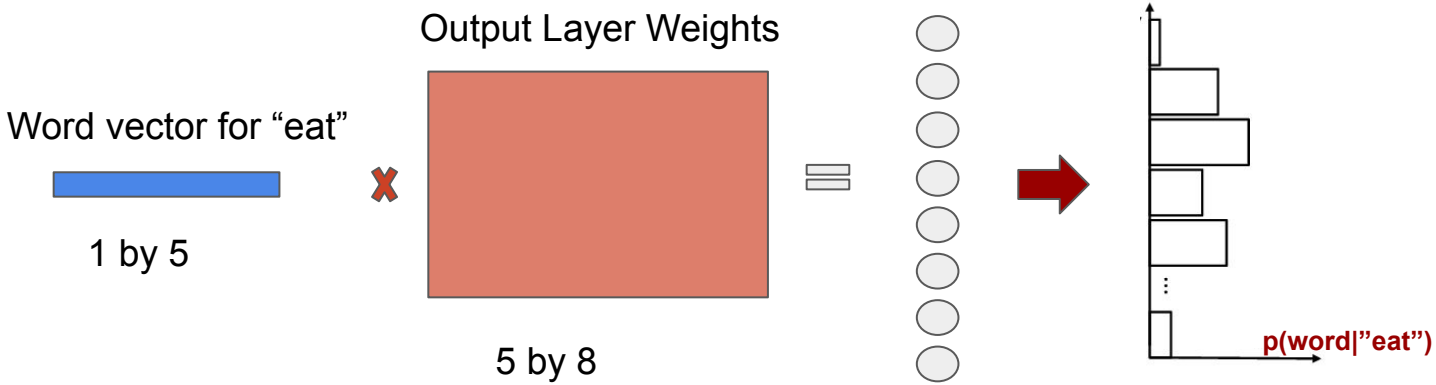
0.96, 1.5, 1.37, 0.34, 1.04

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

Output layer

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

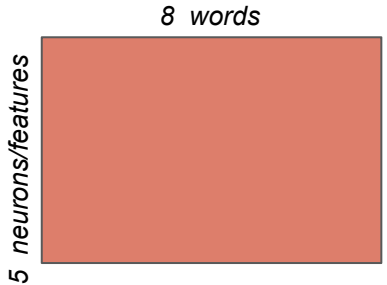




Output layer

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

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



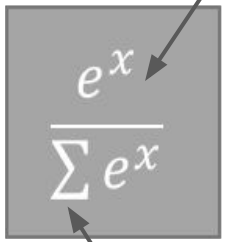
Output weights for “apple”

e^(IVec[eat]*OVec[apple])

Word vector for “eat”

5 features

×



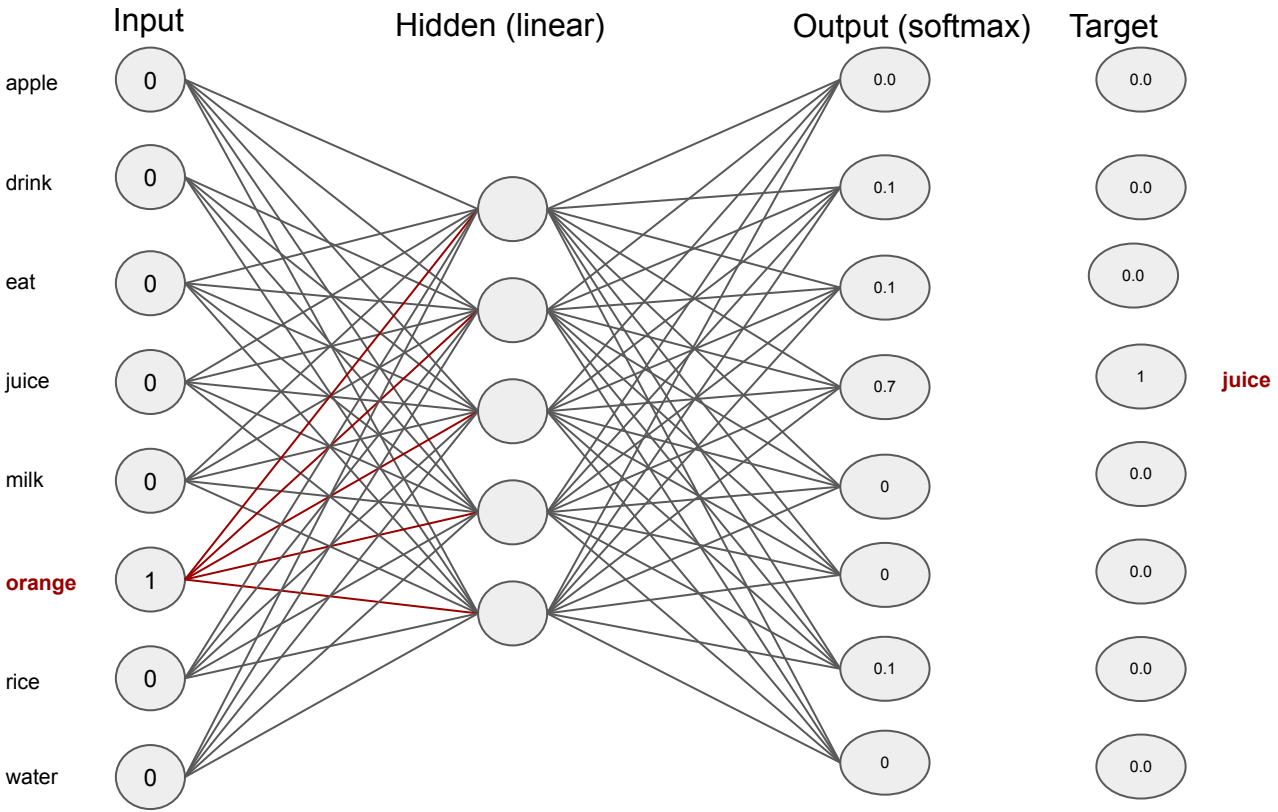
=

Probability that if you randomly pick a word nearby “eat”, that it is “apple”

p(apple|eat)

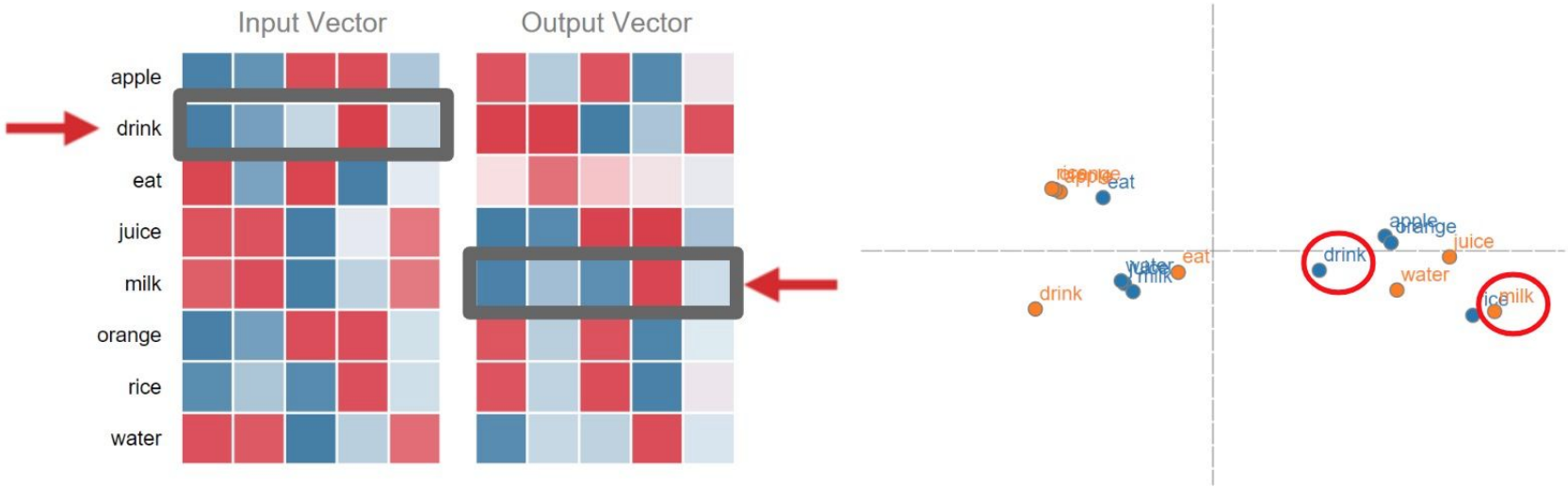
e^(IVec[eat]*OVec[apple]) + e^(IVec[eat]*OVec[juice]) + e^(IVec[eat]*OVec[drink])+e^(IVec[eat]*OVec[other vocab words)

Word2Vec



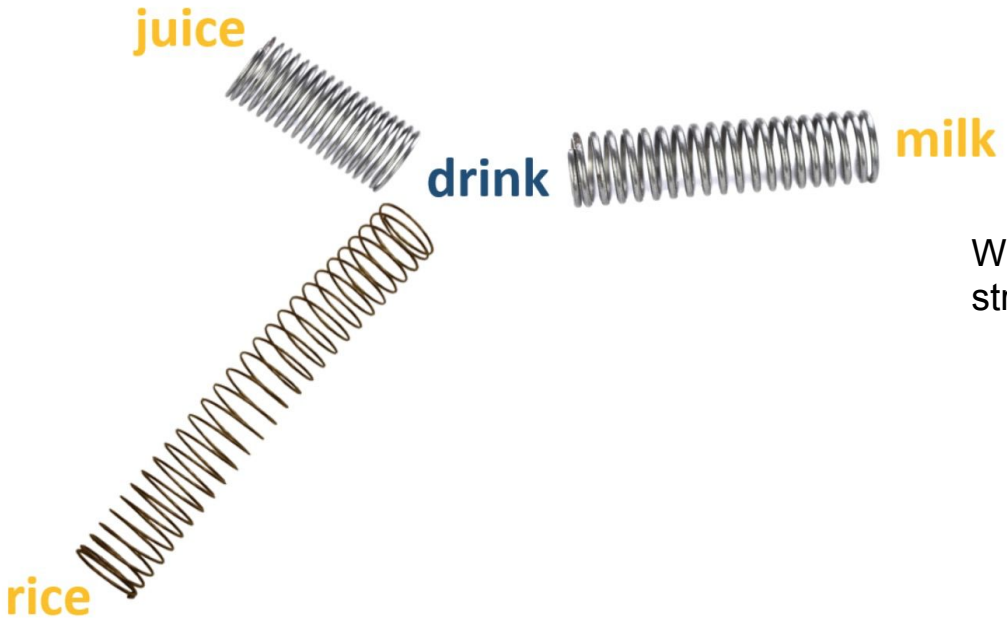
Then, we can compute the loss and call gradient descent to update model parameters.

Updating word vectors



From Xin Rong 2016

A force-directed graph



What decides the strength of the string?

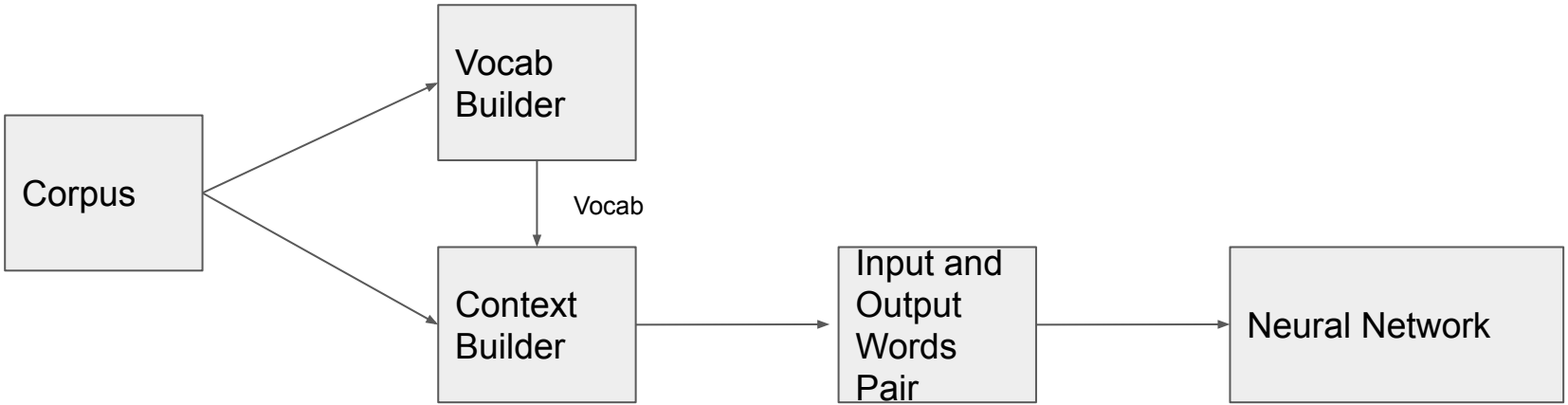
Idea behind Word2Vec

- 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)

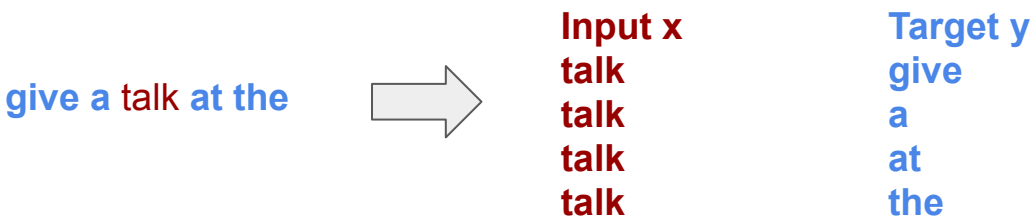
Input and output words

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



Skip-Gram

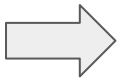
- Task Definition: given a specific word, predict its nearby word (probability output)
- Model input: source word, Model output: nearby word
- Input is one word, output is one word
- The output can be interpreted as prob. scores, which are regarded as how likely it is that each vocabulary word can be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



CBoW

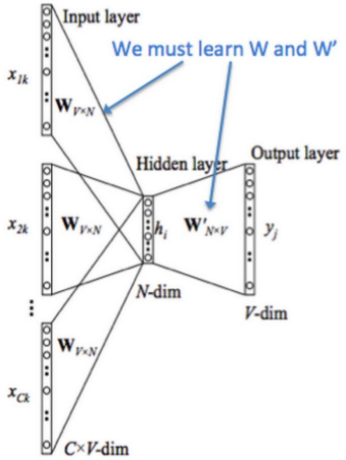
- Task Definition: given context, predict its target word
- Model input: context (several words), Model output: center word
- Input is several words, output is one word
- Core Trick: **average** these context vectors for prob. score computing

give a **talk** at the



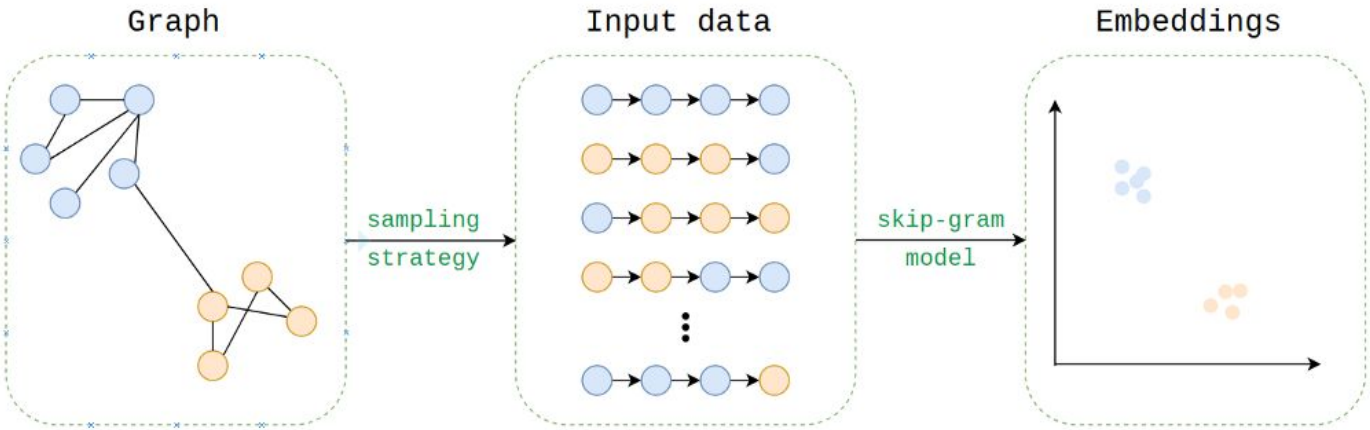
Input x
(give,a,at,the)

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.

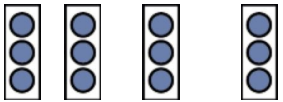


3. Neural Networks for NLP

Sequence of words

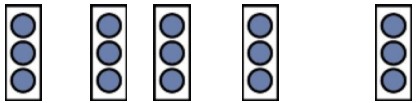
- Each sentence or document can be regarded as a sequence of vectors.
- The shape of matrix depends on the length of sequence. However, the majority of ML systems need fixed-length feature vectors.
- One simple solution: average the sequence of vectors, just like bag-of-words (abandon order information).

I hate this movie



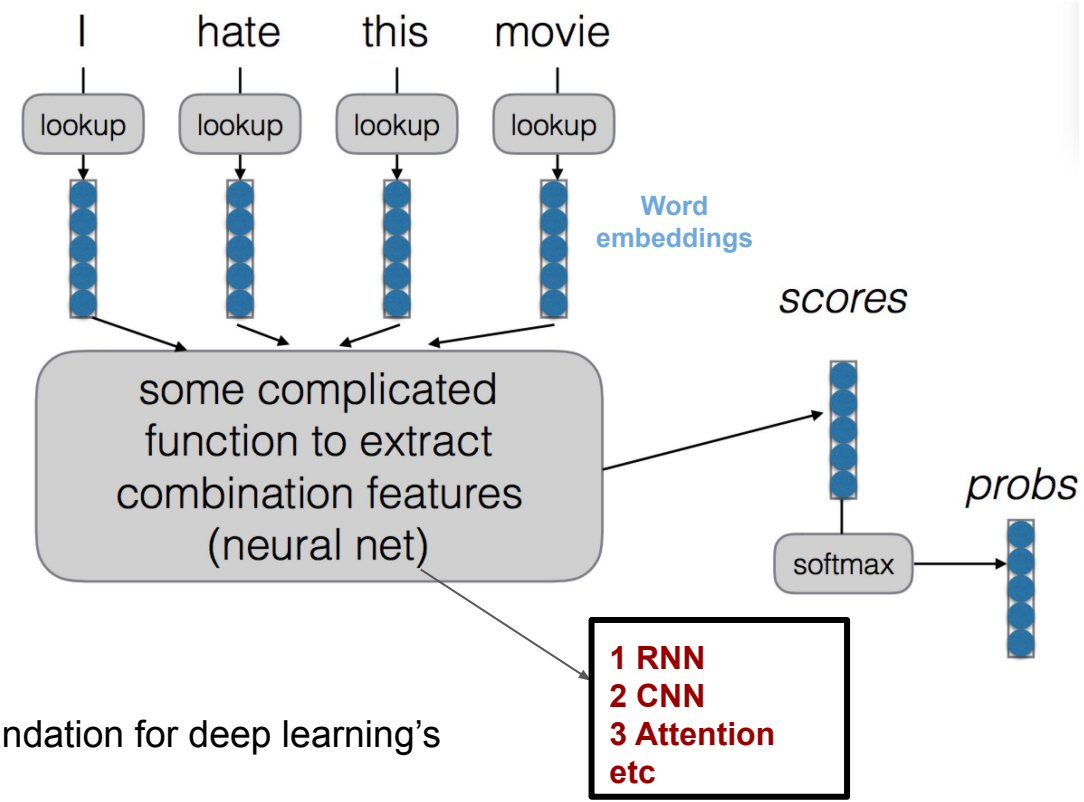
4 by d

This is my favorite movie.



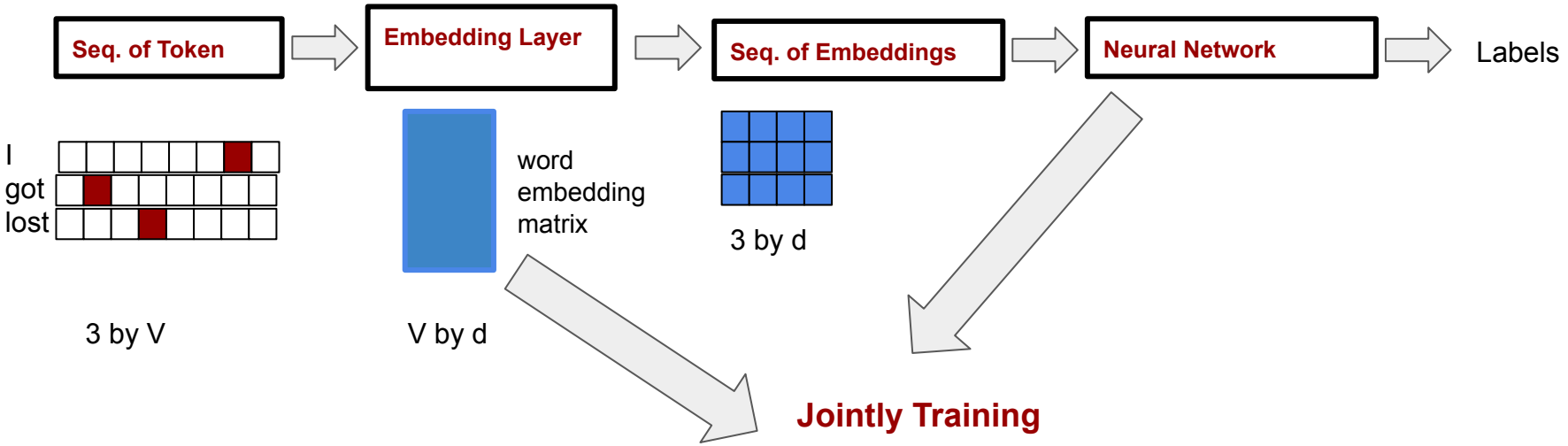
5 by d

Complex semantic



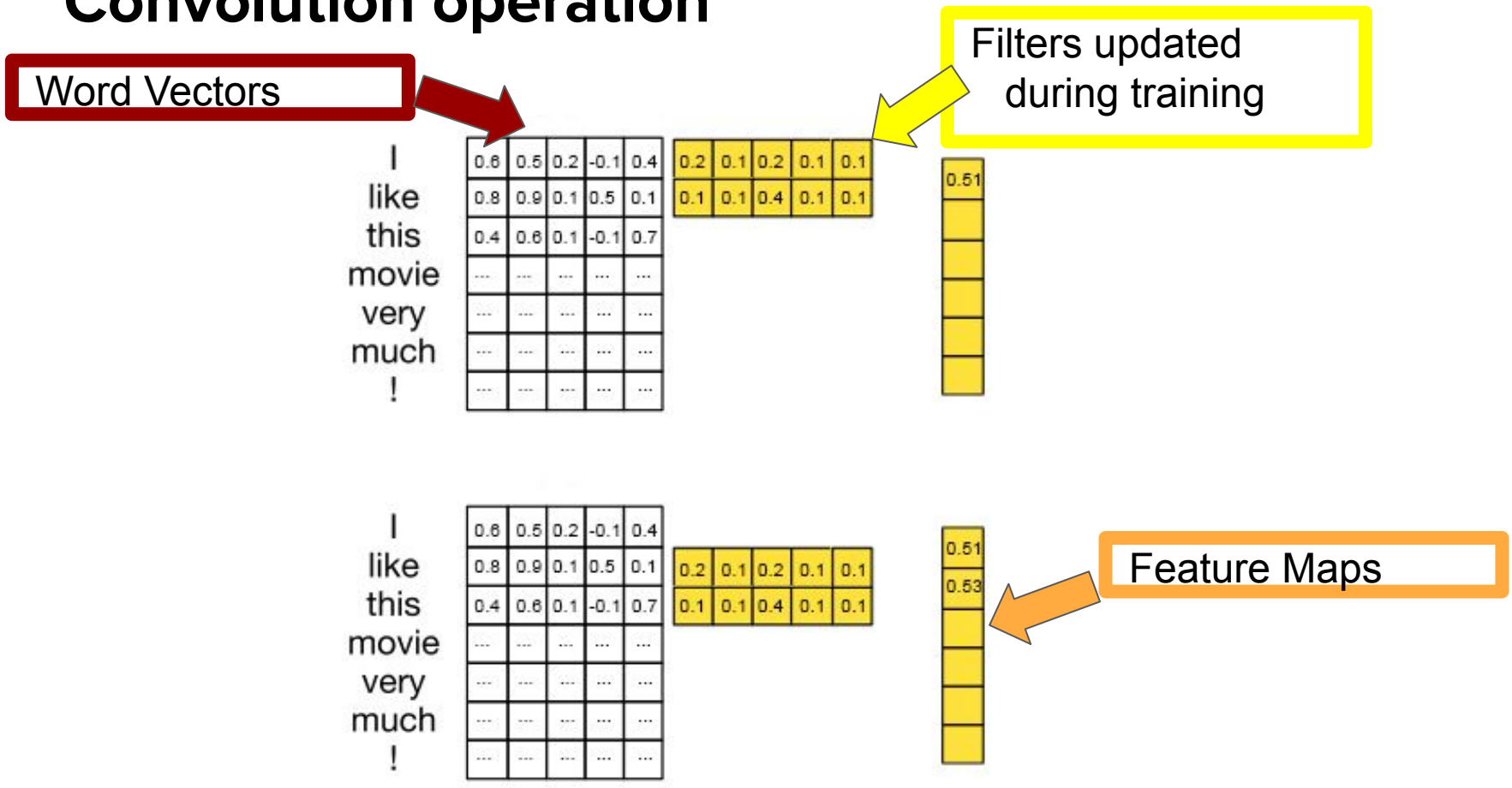
Word Embeddings is the foundation for deep learning's applications on NLP

Neural networks for NLP

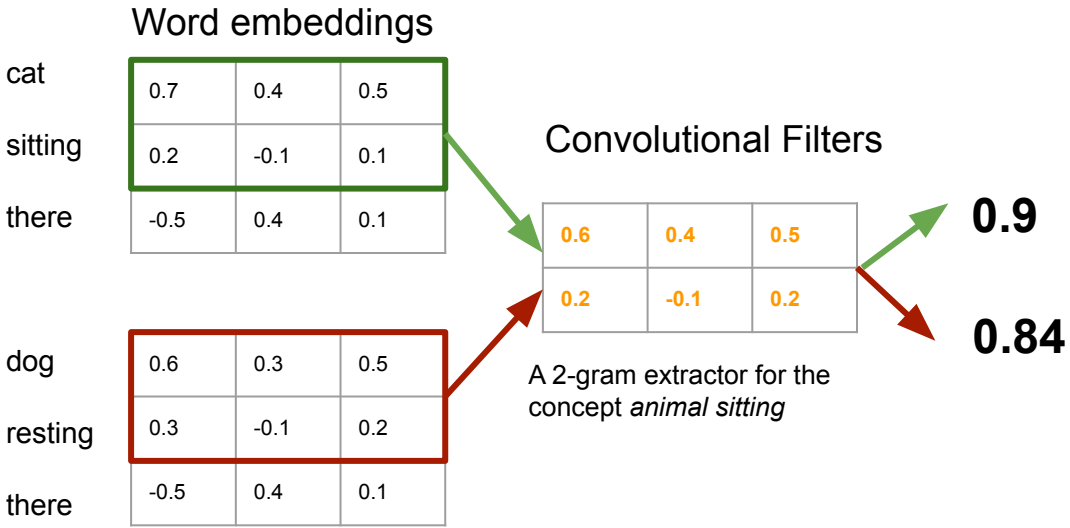


- Learn from Scratch: Random initialize the word embedding matrix and update the matrix and neural network parameters in the specific task
- Pre-train: Got pre-trained word embeddings as the embedding layer and only update neural network parameters in the specific task
- **Pre-train then fine tune**

Convolution operation



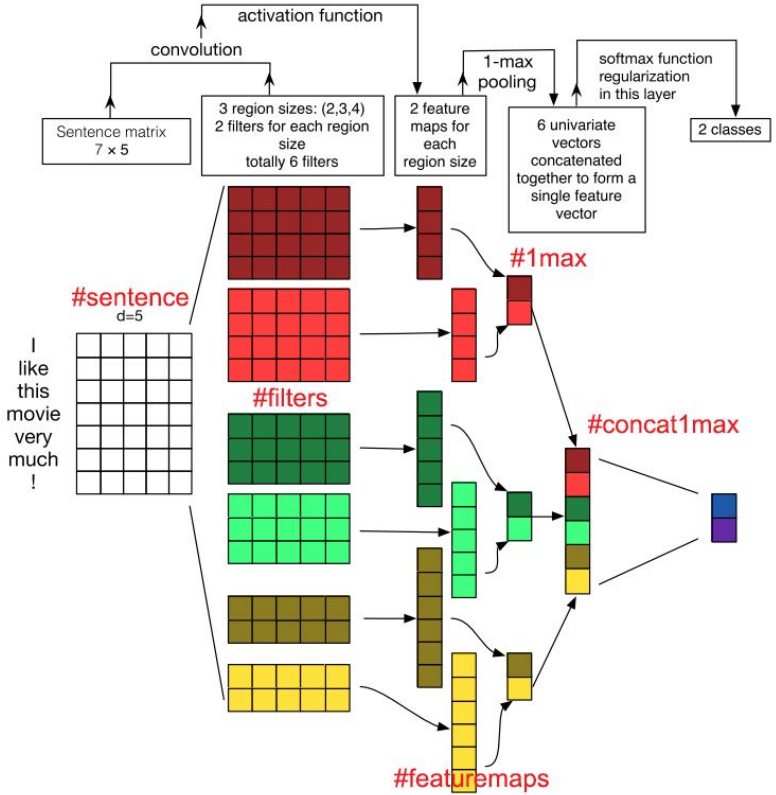
Toy example



- 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

A model that can “read” text

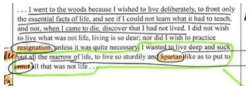
Text without annotation

Pre-train

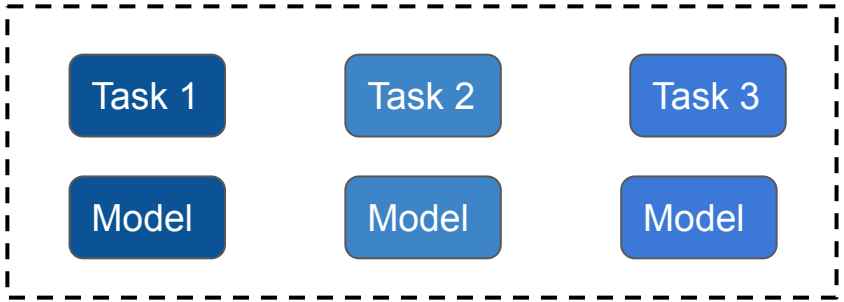
Model

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.

Fine-tune



Task-specific data with labels

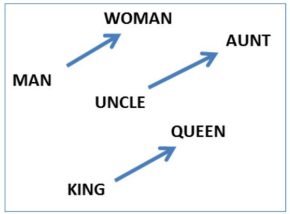


Is Word2Vec good enough?

- 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

Corpus

Word2Vec Tool



Word Embeddings used for downstream tasks

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* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

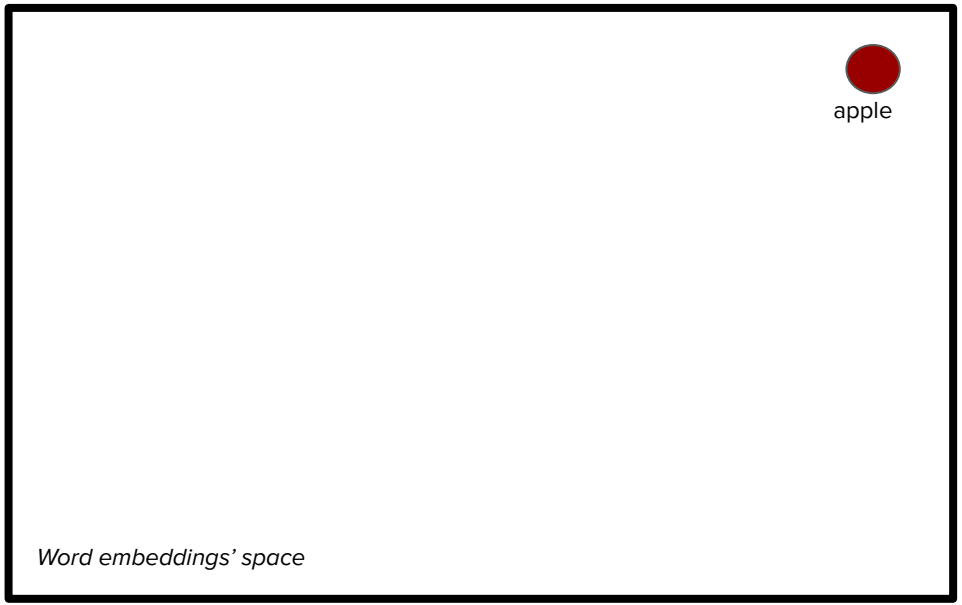
Abstract

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.

https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Word Embeddings

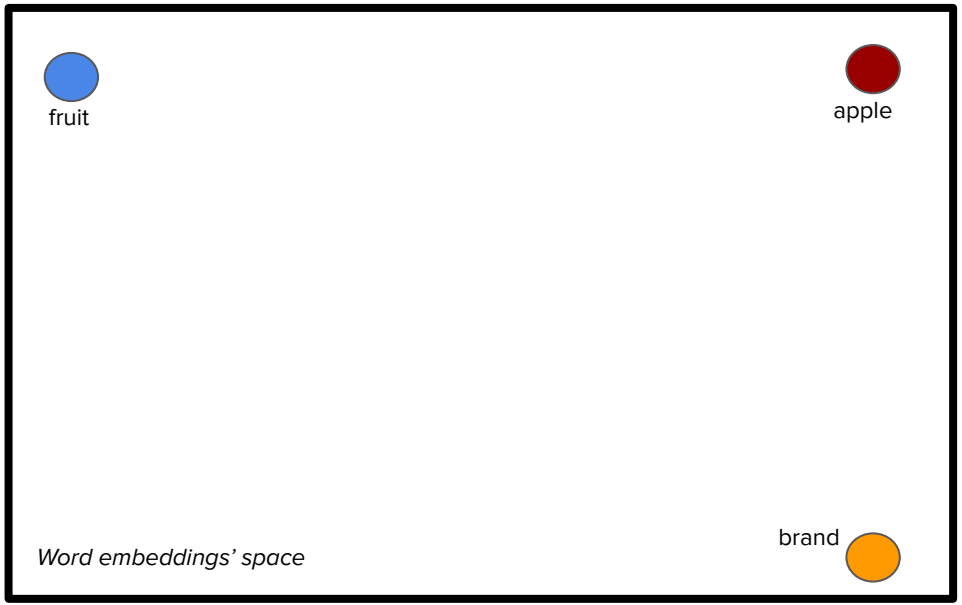
- **Apple** in two sentences:
  - Sentence 1: My favorite fruit is **apple**
  - Sentence 2: Solution: My favorite brand is **apple**



One embedding has multiple senses

Contextualized Word Embeddings

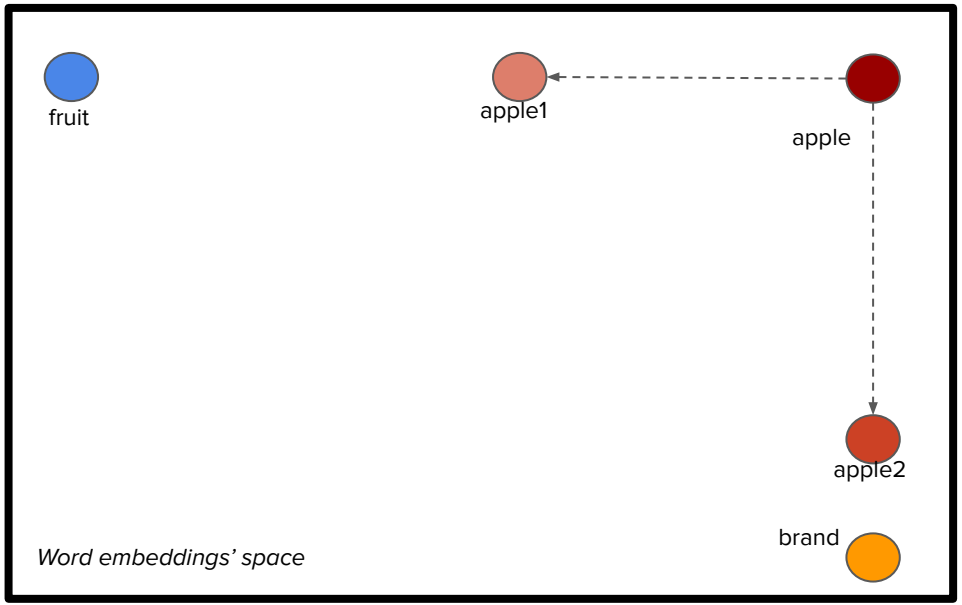
- Telling context in words
  - Sentence 1: My favorite fruit is **apple1**
  - Sentence 2: Solution: My favorite brand is **apple2**



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"

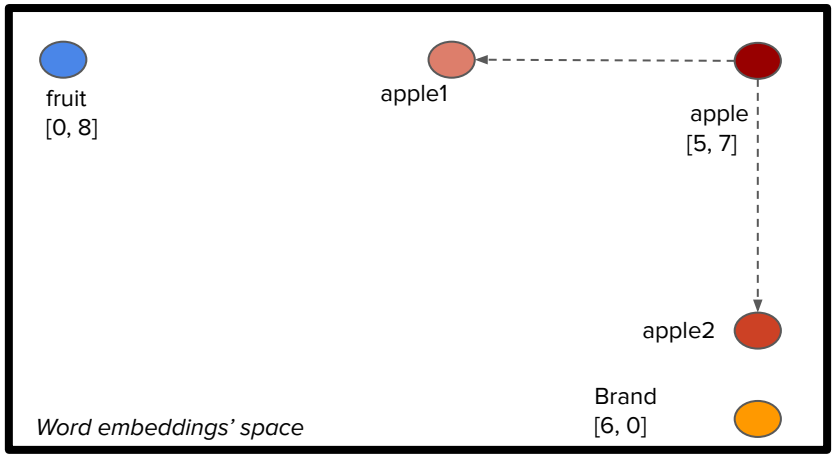
This is how attention will work

How to move one word closer to another one

- Average two words
  - apple1 = 0.8*apple + 0.2*fruit = 0.8*[5,7] + 0.2*[0,8]= [4.0, 7.2]
  - apple2 = 0.9*apple + 0.1*brand = 0.9*[5,7] + 0.1*[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

= Attention(apple, my)*base_vec(my) + Attention(apple, favorite)*base_vec(favorite) + Attention(apple, fruit)*base_vec(fruit) + Attention(apple, is)*base_vec(is) + Attention(apple, apple)*base_vec(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 * fruit + 0.8 * 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	Attention Step
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

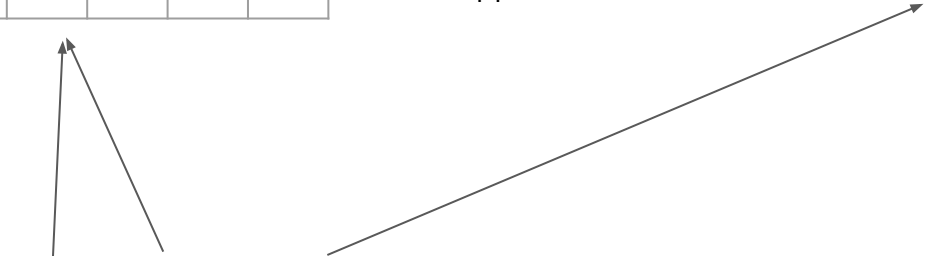
Here, we only use the **same embedding** for attention weights and the base vectors

Self-Attention Layer

Sentence i: My favourite fruit is apple

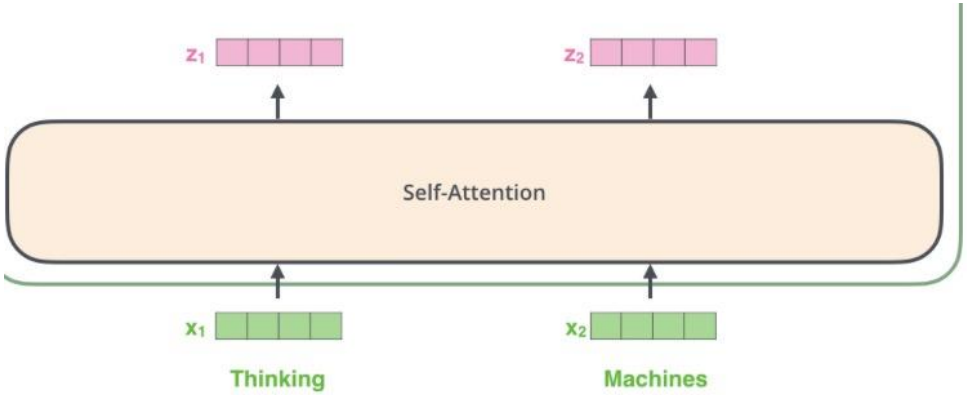
	my	favourite	fruit	is	apple
my	1	0	0	0	0
favourite	0	1	0	0	0
fruit	0	0	1	0	0.25
is	0	0	0	1	0
apple	0	0	0.25	0	1

New Word Index	Attention Step
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



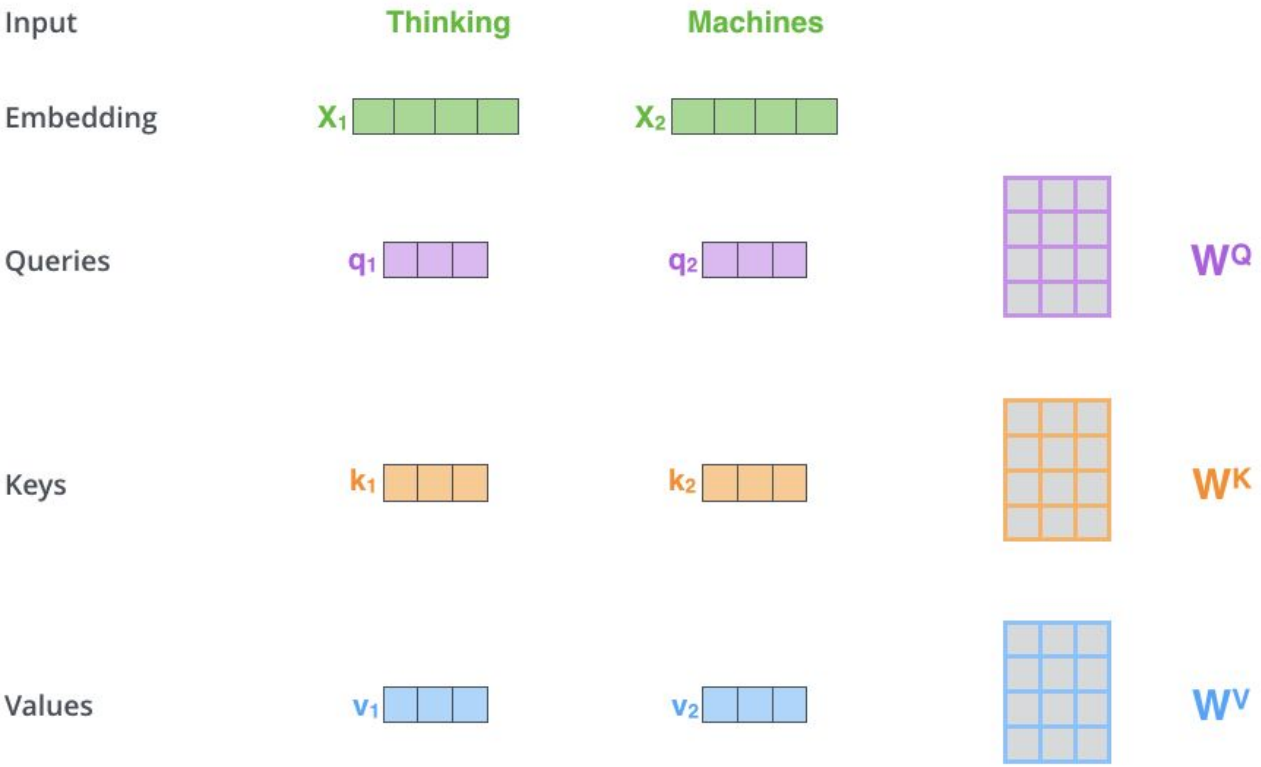
Query, Key and Value vector would be created for each word

Self-Attention Layer

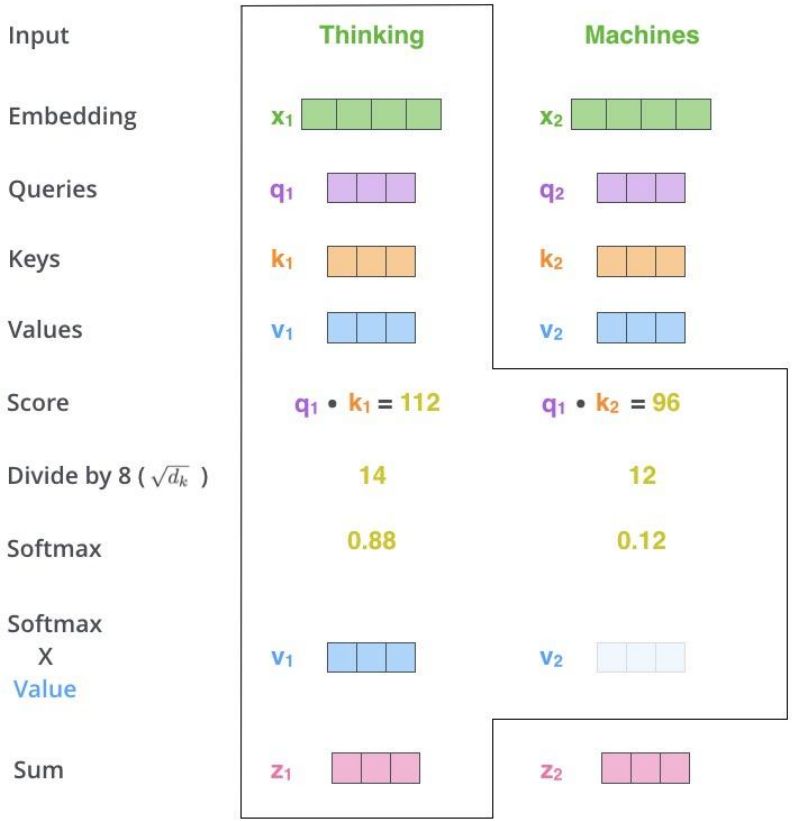


A sequence of vectors in,
A sequence of vector out

Self-Attention Layer

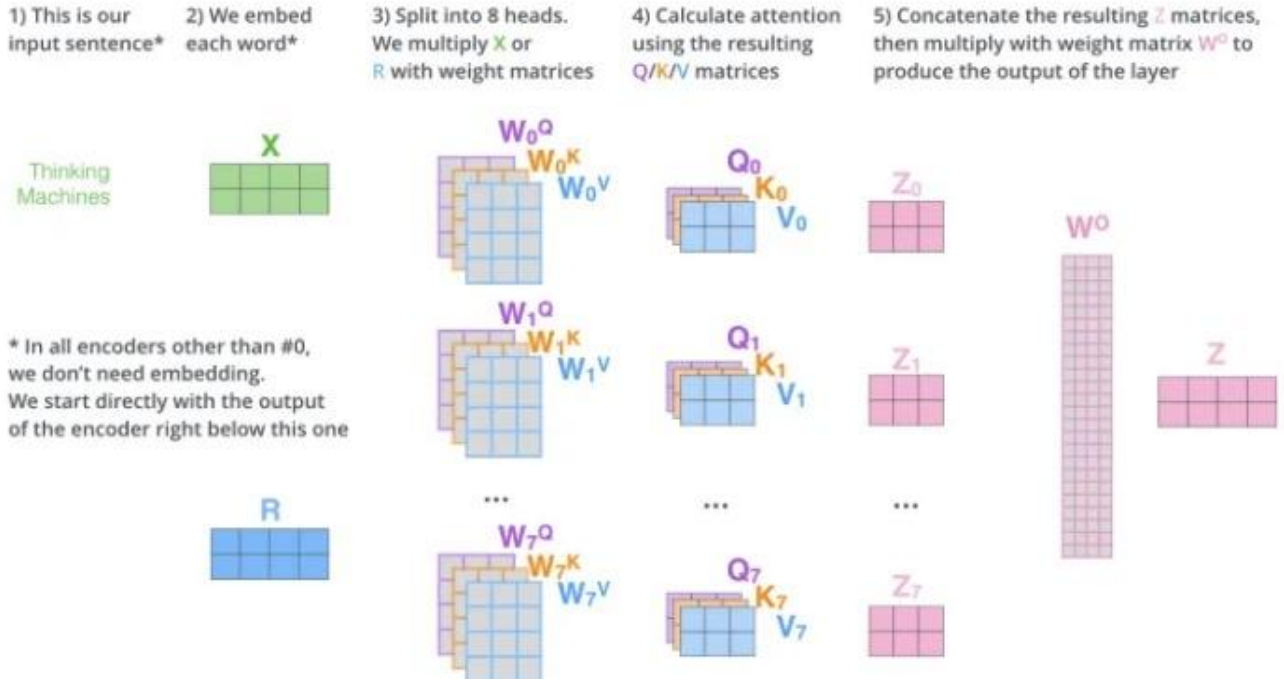


Self-Attention Layer



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

MultiHeadAttention



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

Keras Implementation

https://keras.io/examples/nlp/text_classification_with_transformer/

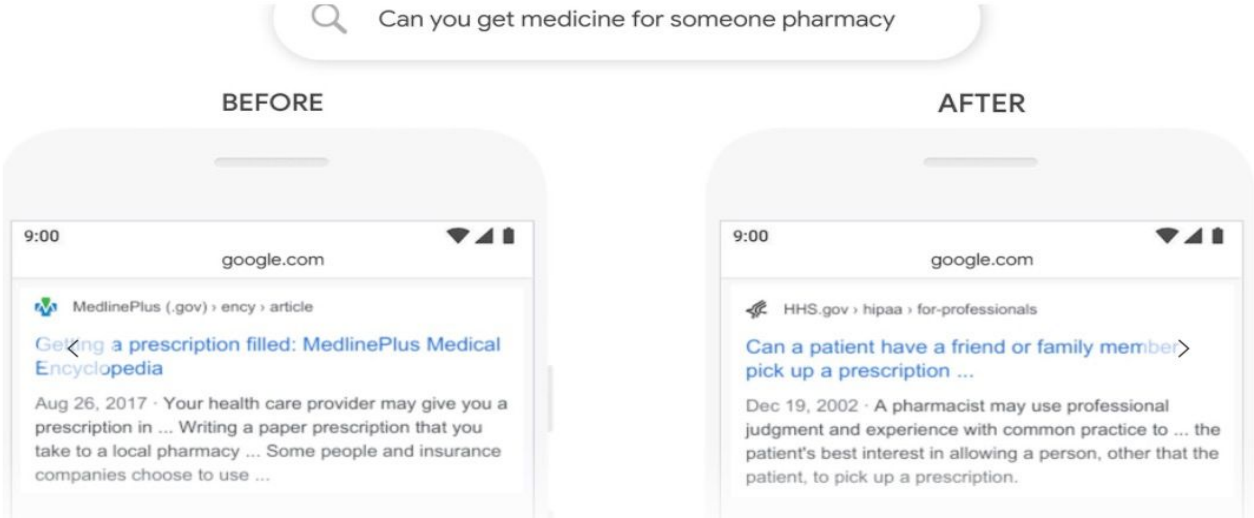
Details Behind Transformer

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

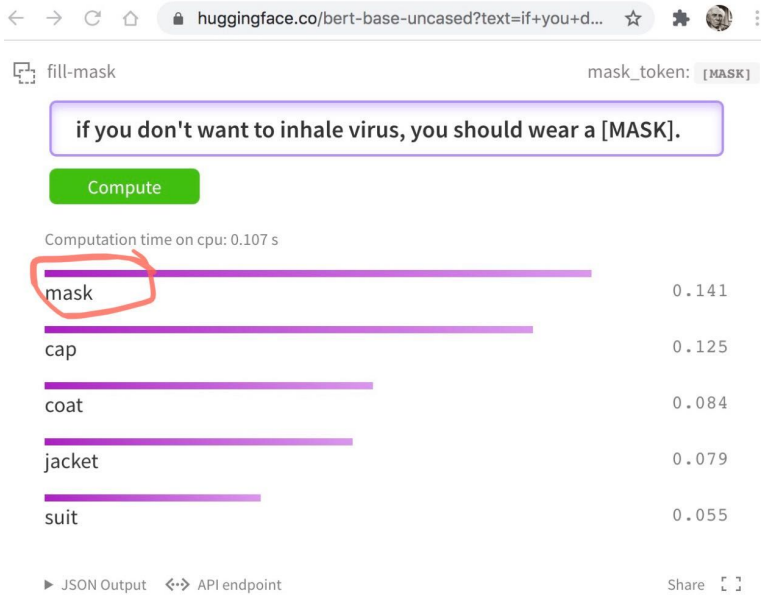
BERT in Google Search



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.

https://blog.google/products/search/search-language-understanding-bert/

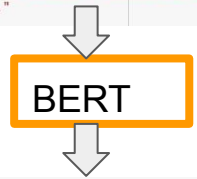
BERT vs Somebody



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

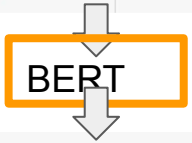
```
question = "What can we learn in NUS?"
answer_text = "The National University of Singapore (NUS) is the national research university of Singapore. \
               Founded in 1905 as the Straits Settlements and Federated Malay States Government Medical School, NUS is the oldest higher education institution 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."
```

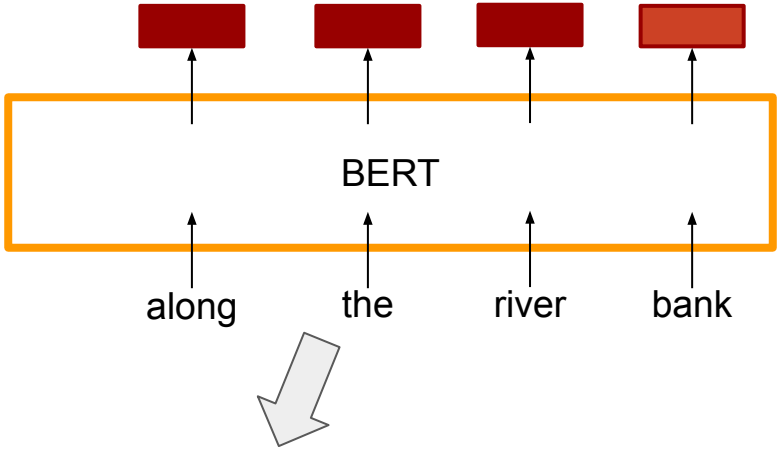



```
print('Answer: "' + answer + '"')
```

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

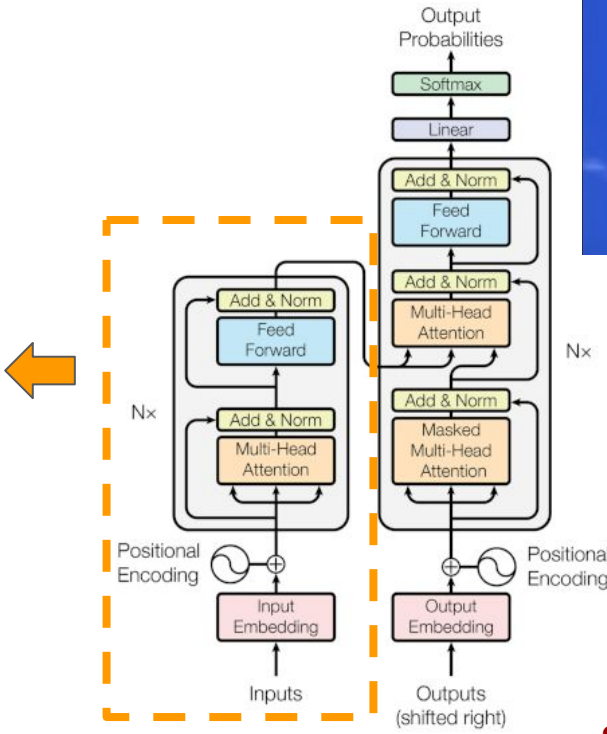
What is BERT

- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (**BERT**)
- BERT: Encoder of Transformer,



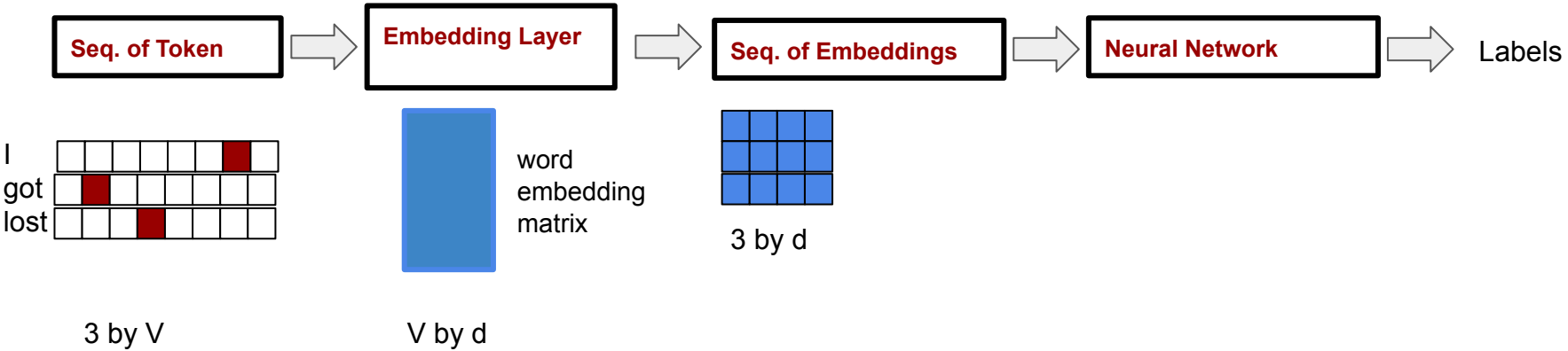
Given a sequence of words, generate a sequence of vectors and then can be used for various NLP tasks

Transformer



Solve Seq2Seq Task

Neural Networks for NLP



Each word has a fixed vector

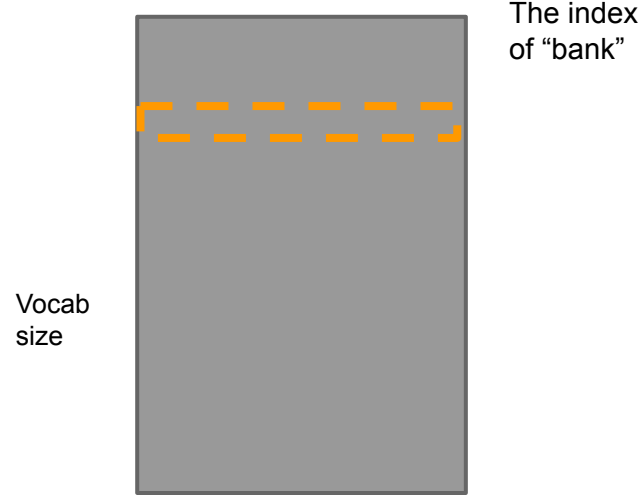
Can not address multi-sense problem!

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.

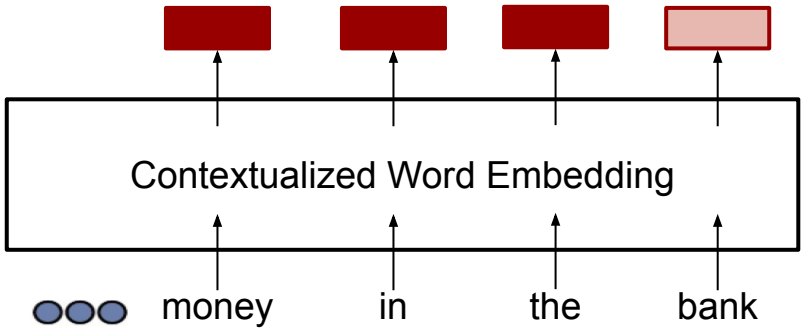
The third sense of not?

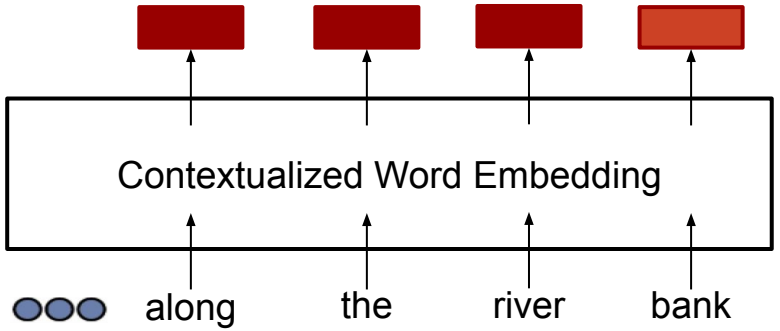
Word2Vec, Fasttext,Glove and other word embedding models

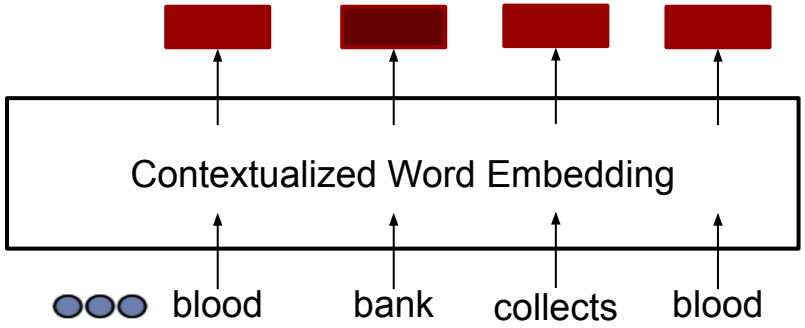


Contextualized Word Embeddings

Encoded Embeddings







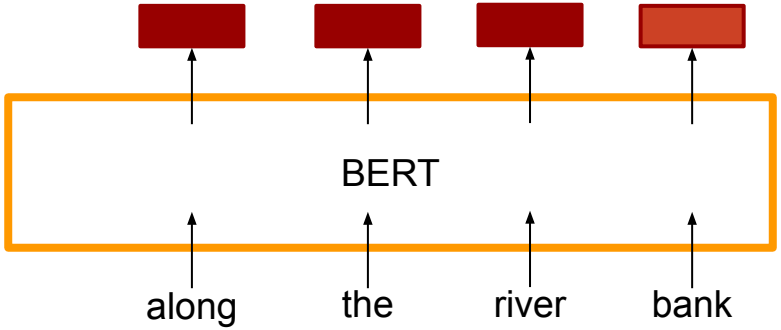
Different Contexts,
Different Encoded Embeddings for bank.

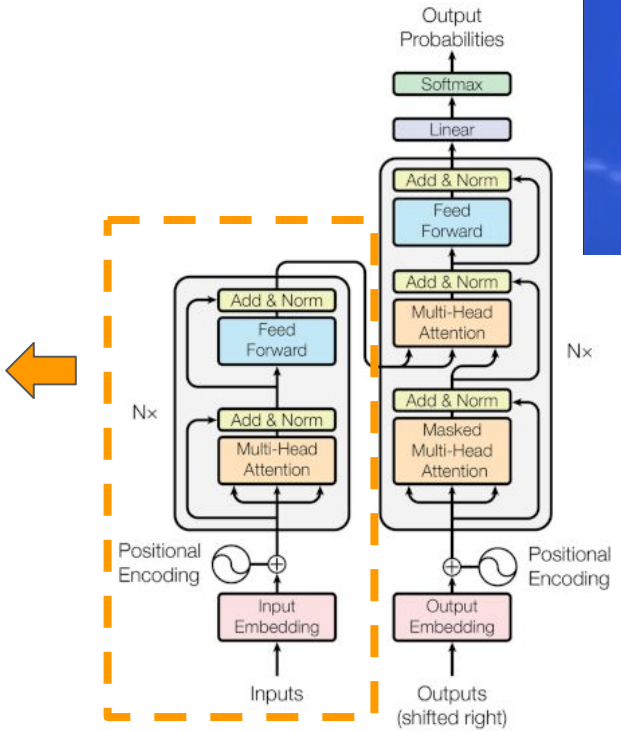
Embeddings generated from BERT

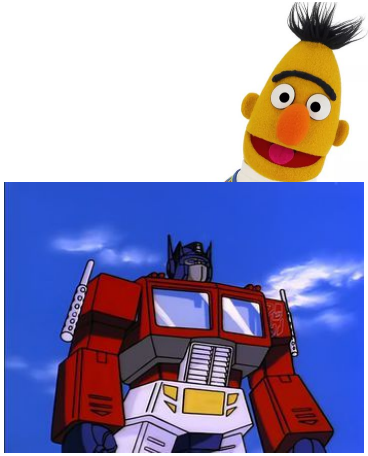
Cos-similarities among vectors of “apple” in different context

BERT

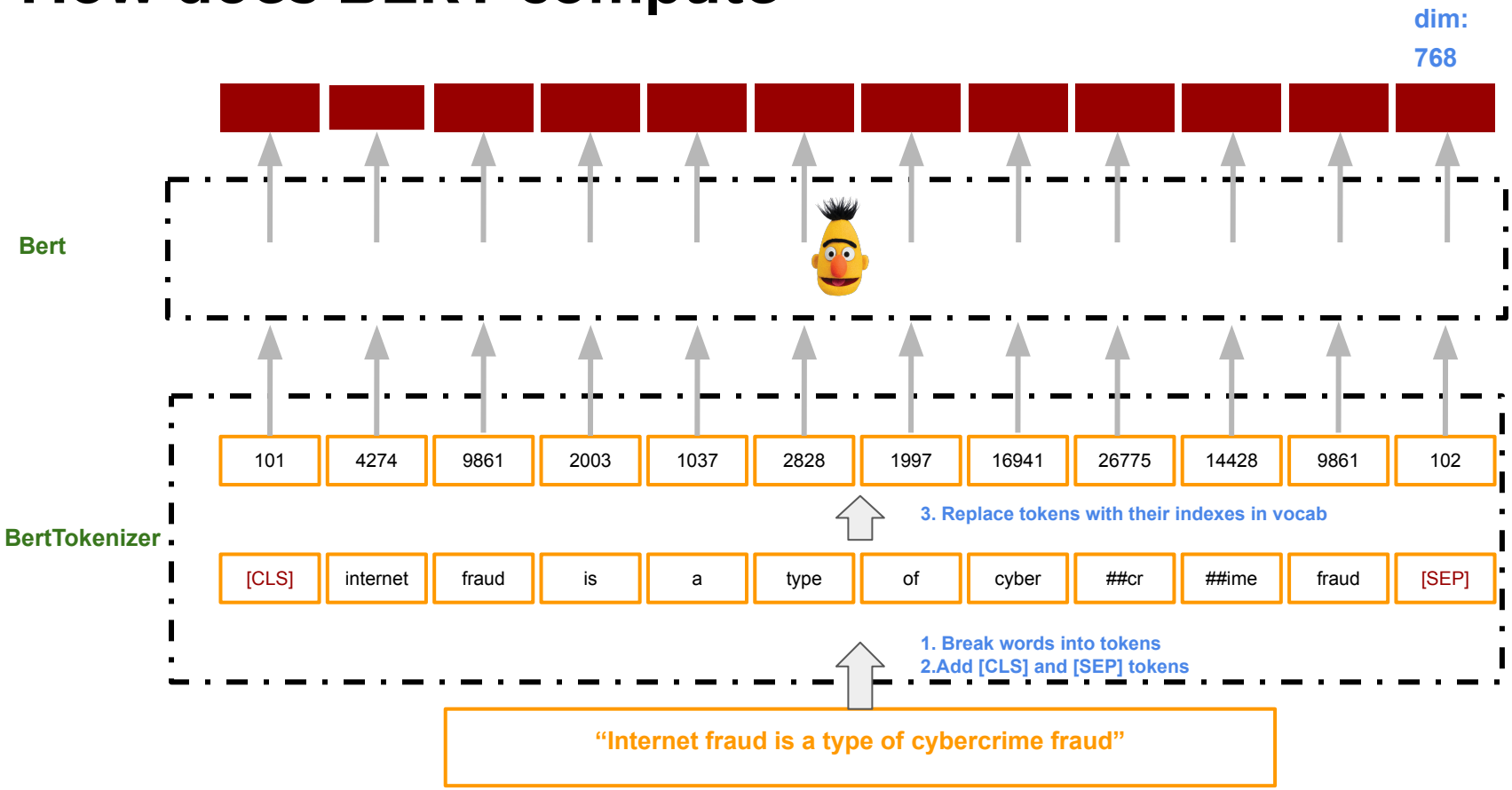
- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (**BERT**)
- BERT: Encoder of Transformer,







How does BERT compute



How to use BERT

Text without annotation

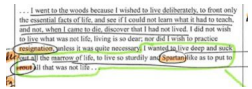
Pre-train

A model that can “read” text

BERT

Learned from a large amount of **text without annotation**

Fine-tune



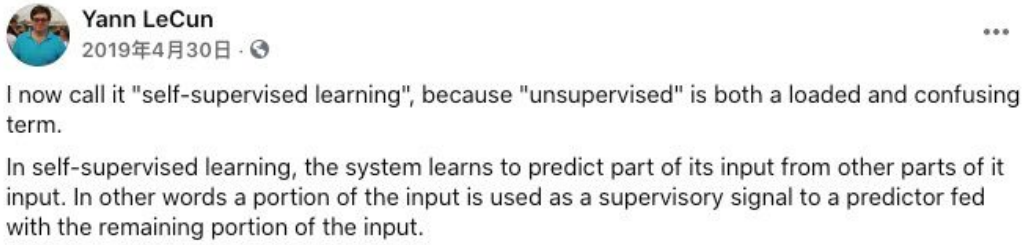
Task-specific data with labels

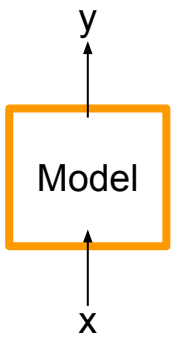
Task 1 | Task 2 | Task 3

BERT | BERT | BERT

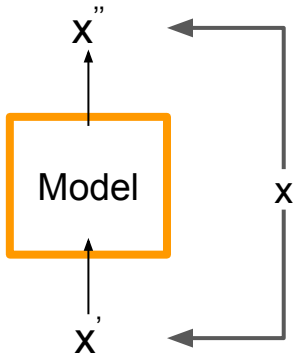
How to Pre-Train

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





Supervised



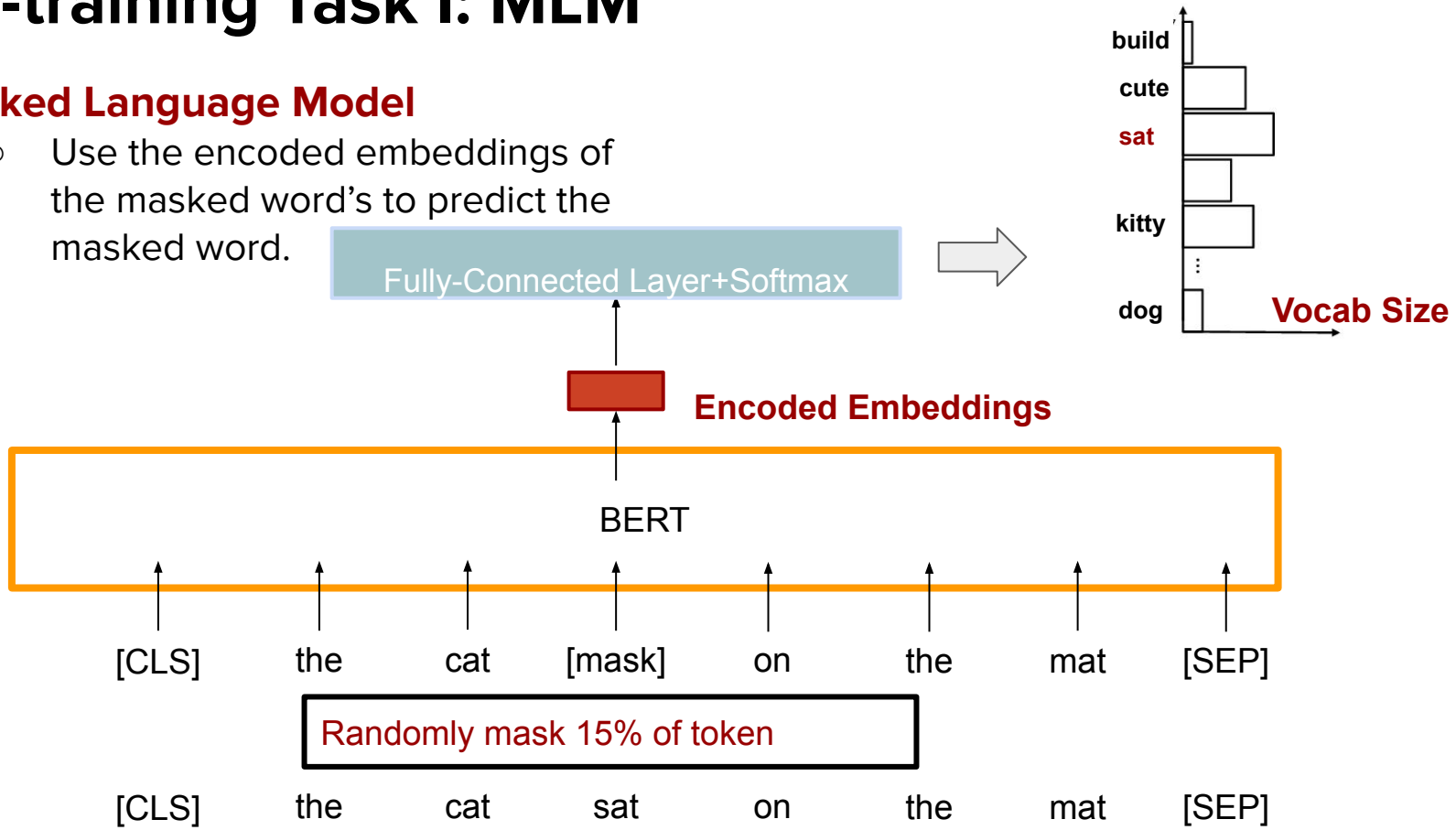
Generated by Rules

Automatically generate some kind of supervisory tasks

Self-Supervised

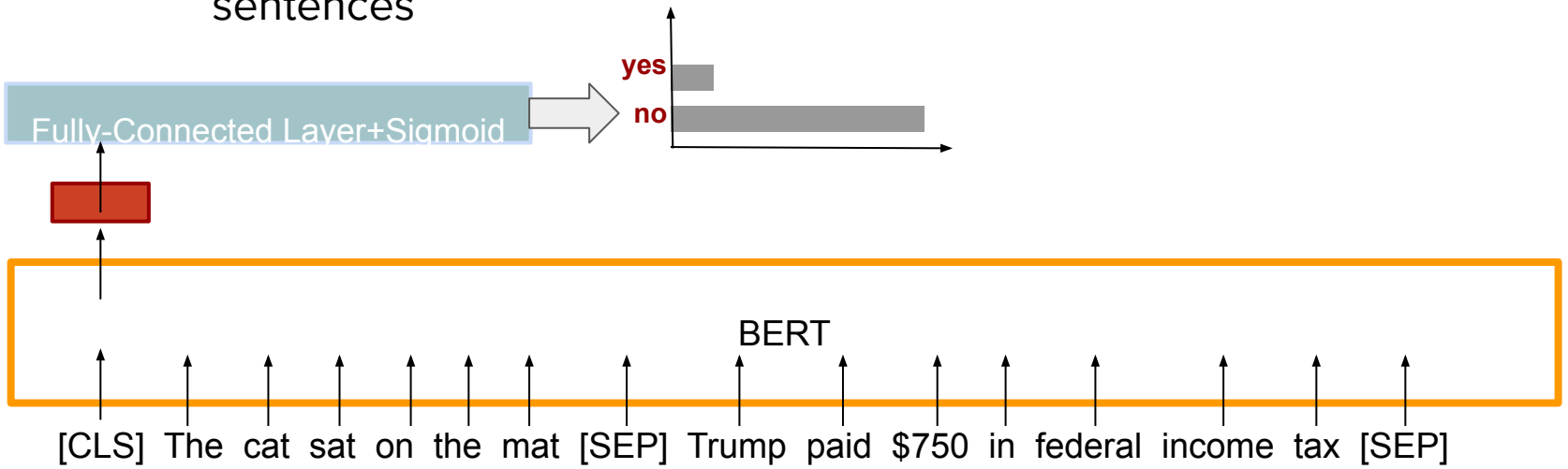
Pre-training Task I: MLM

- **Masked Language Model**
  - Use the encoded embeddings of the masked word's to predict the masked word.

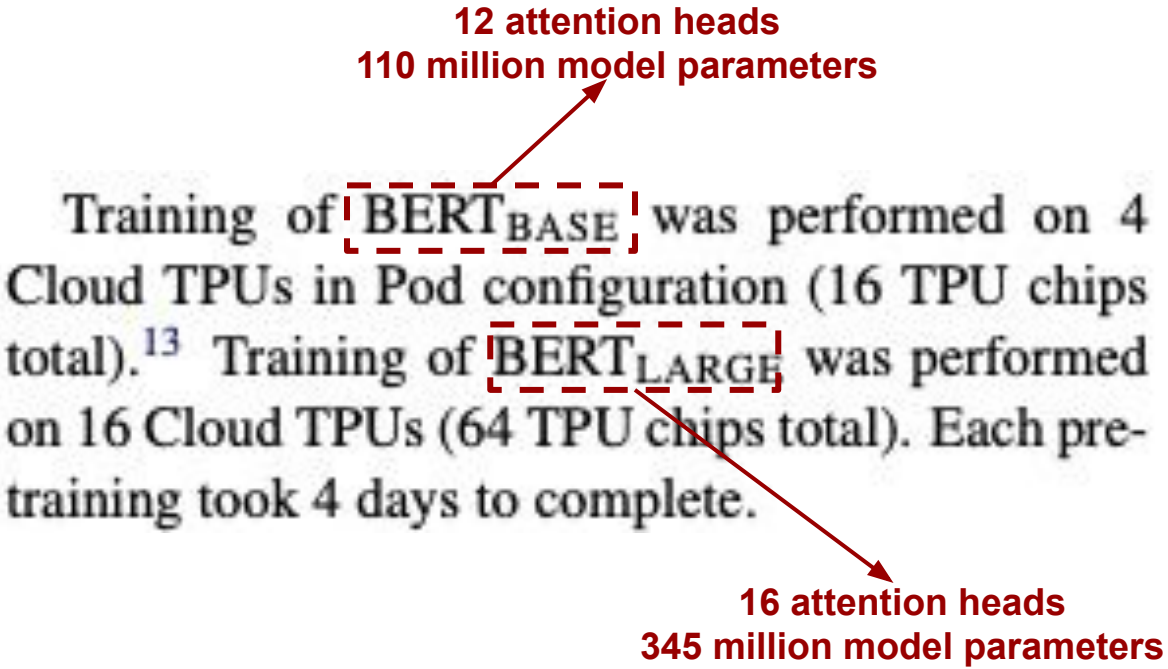


Pre-training Task II: NSP

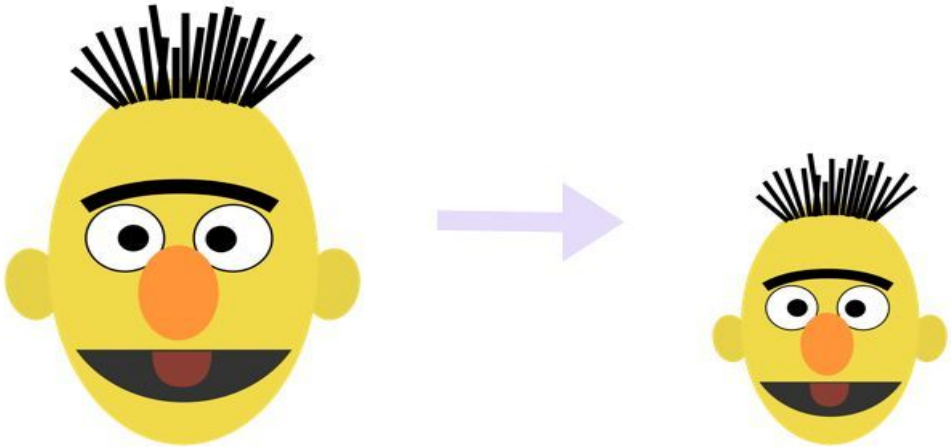
- **Next sentence prediction**
  - Given two sentences A and B, is B likely to be the sentence followed by A?
  - Make bert good at handling relationships between multiple sentences



Huge Model Size



Smaller Model



Published as a conference paper at ICLR 2020

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

Zhenzhong Lan[1] Mingda Chen[2*] Sebastian Goodman[1] Kevin Gimpel[2]
Pivush Sharma[1] Radu Soricut[1]

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

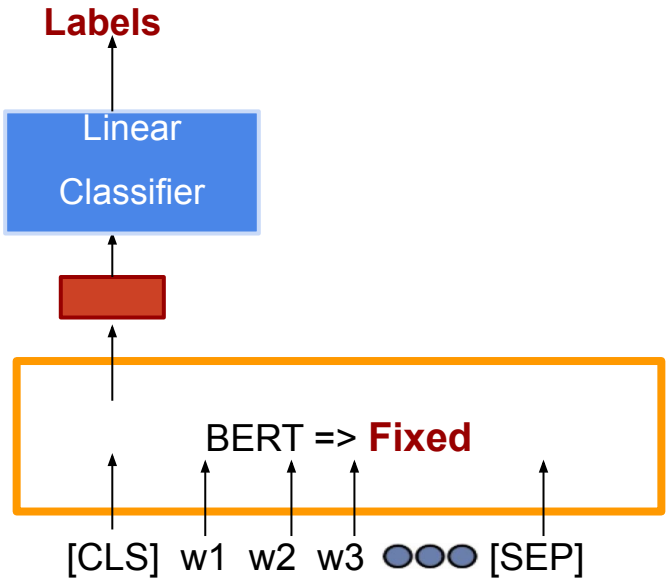
Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF
Hugging Face
{victor,lysandre,julien,thomas}@huggingface.co

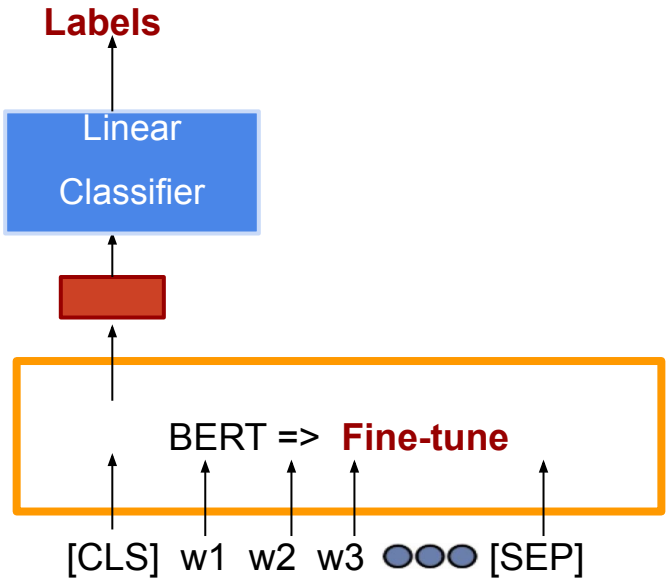
Abstract

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

BERT Usage I

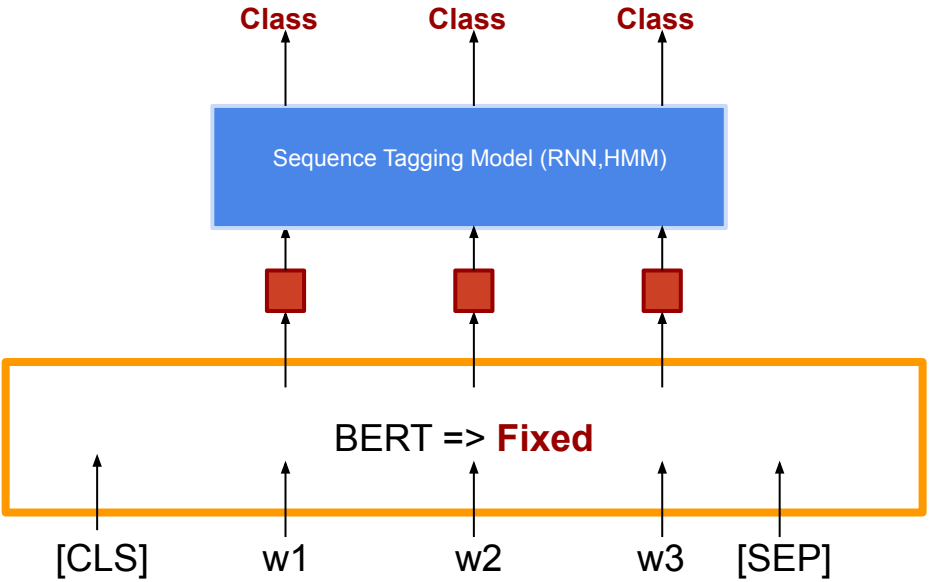
- **Input: Single Sentence Output: Class**
  - Sentiment Analysis
  - Document Classification

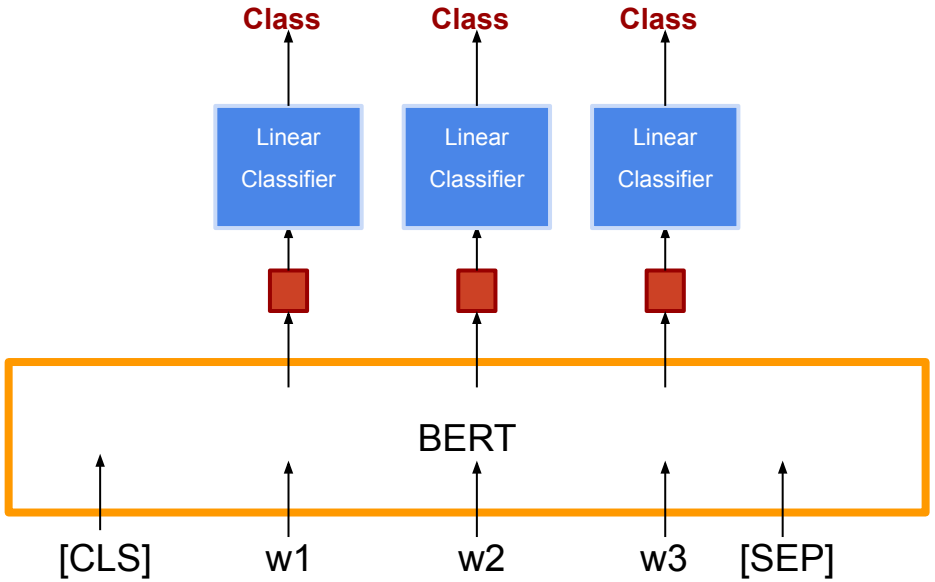




BERT Usage II

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

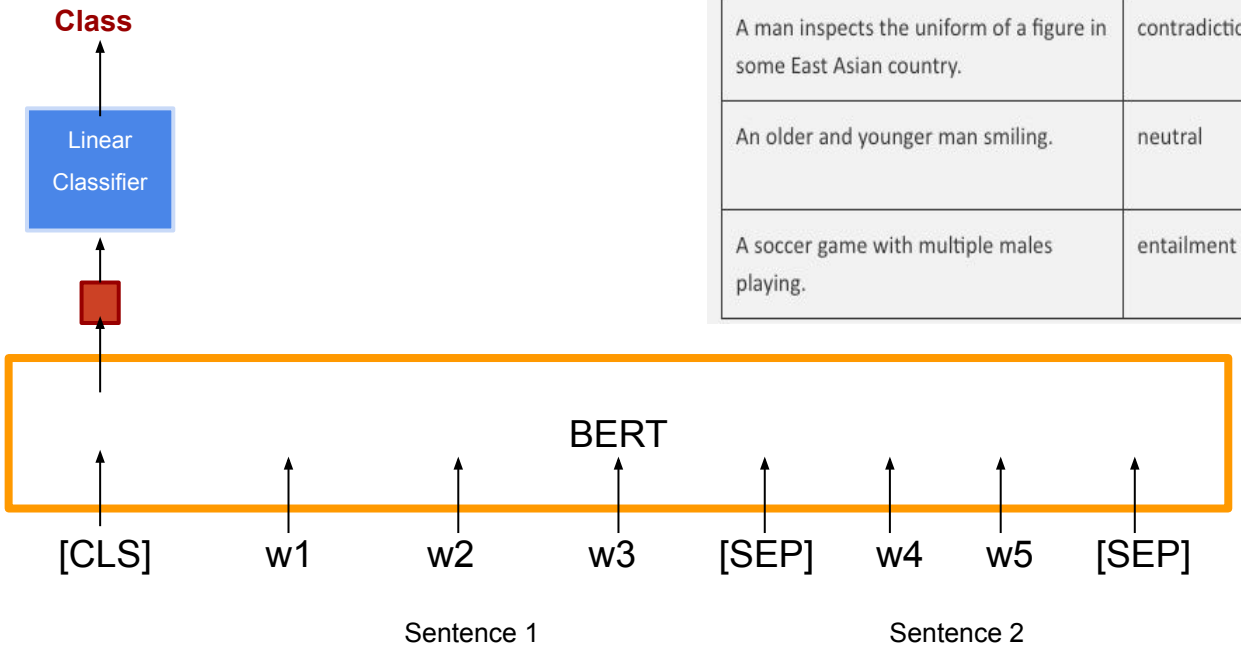




BERT Usage III

- **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

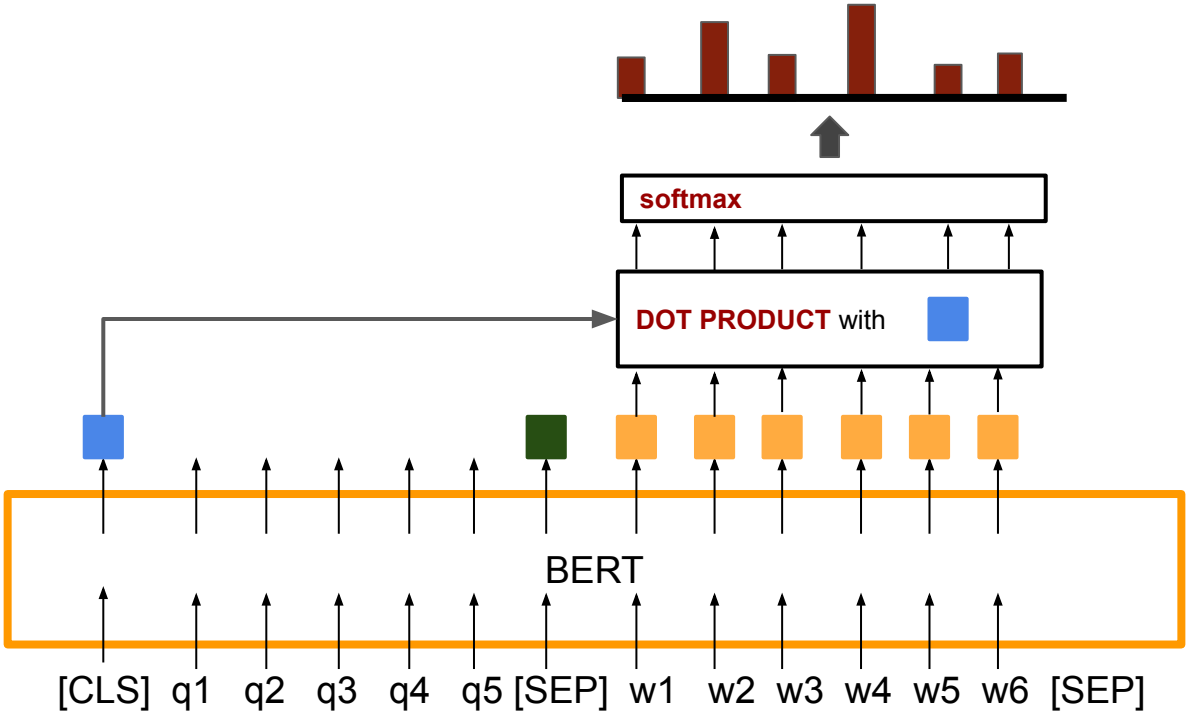
- Extraction-based Question Answering (SQuAD):
  - Input: two “sentences” (Question and Reference Text)
  - Output: start and end positions in Reference (Answer)

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

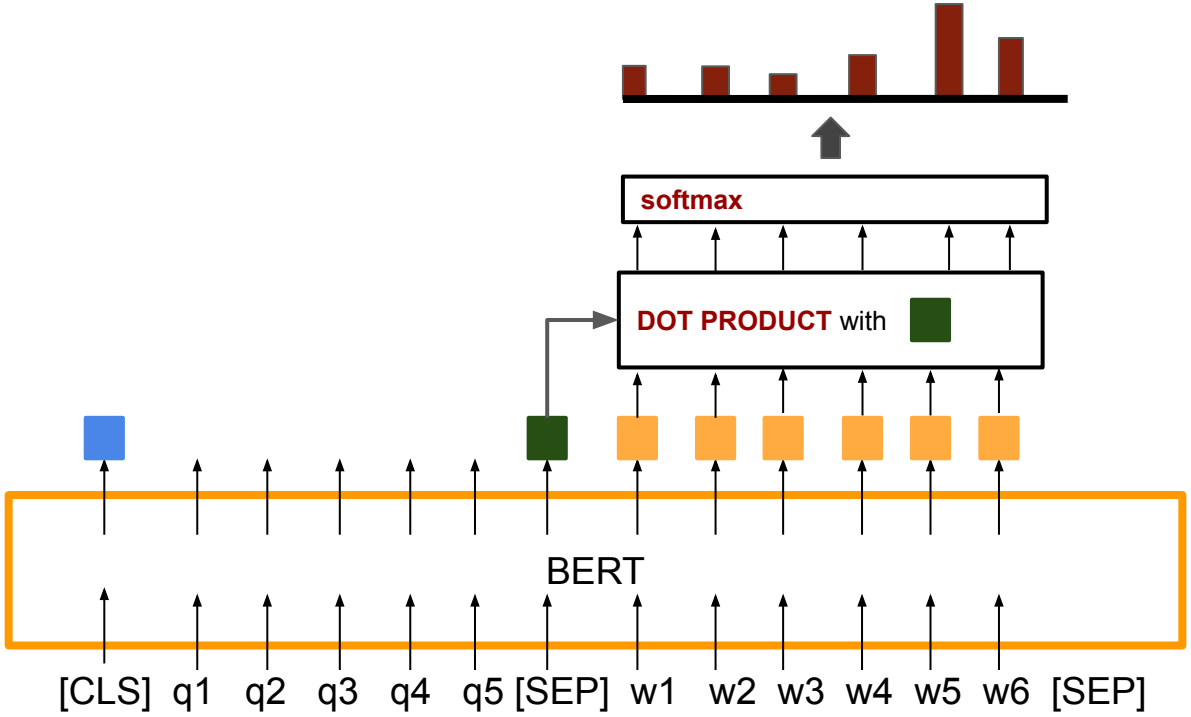
- Extraction-based Question Answering (SQuAD):
  - Question {q1,q2,q3,q4,q5} Reference Text{w1,w2,w3,w4,w5,w6}



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

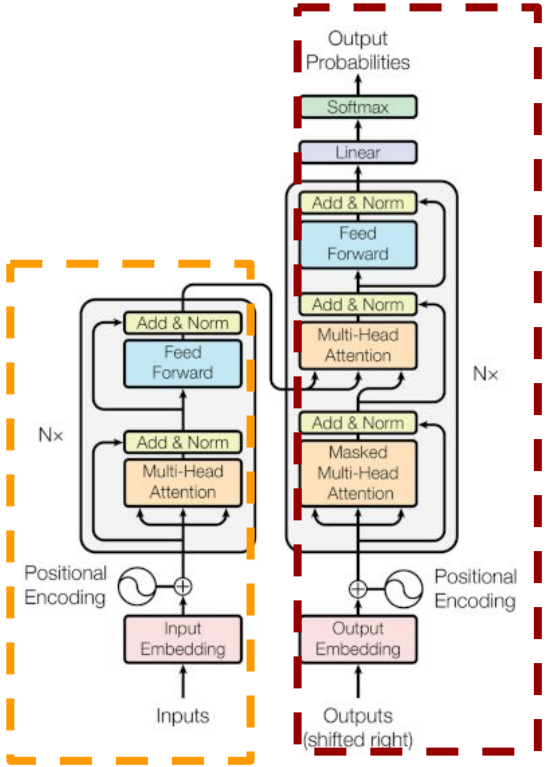
The answer is w4w5

BERT, GPT and Transformers

BERT

Encoder
Maksed “Language” Model

Learn representation



Transformer

GPT2
GPT3

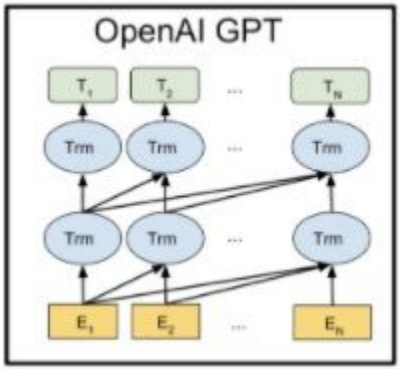
OpenAI

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

Decoder
Language Model

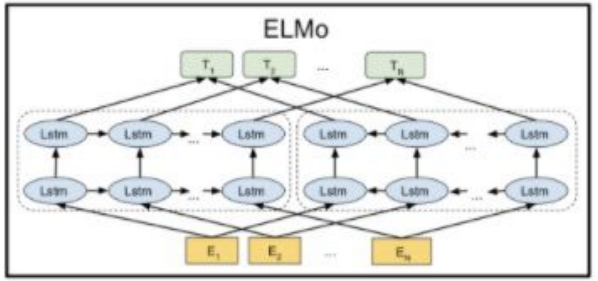
Generate probabilities

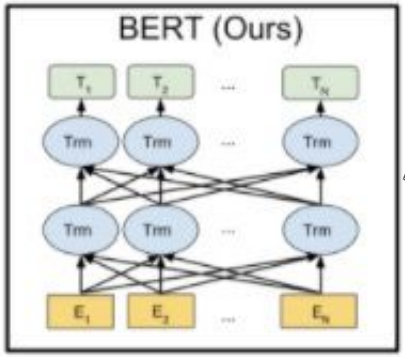
BERT and other Pre-trained Models



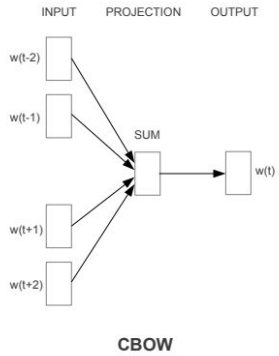
From
single-directional to
Bi-directional

From RNNs to
transformers





From FCNs to
transformers

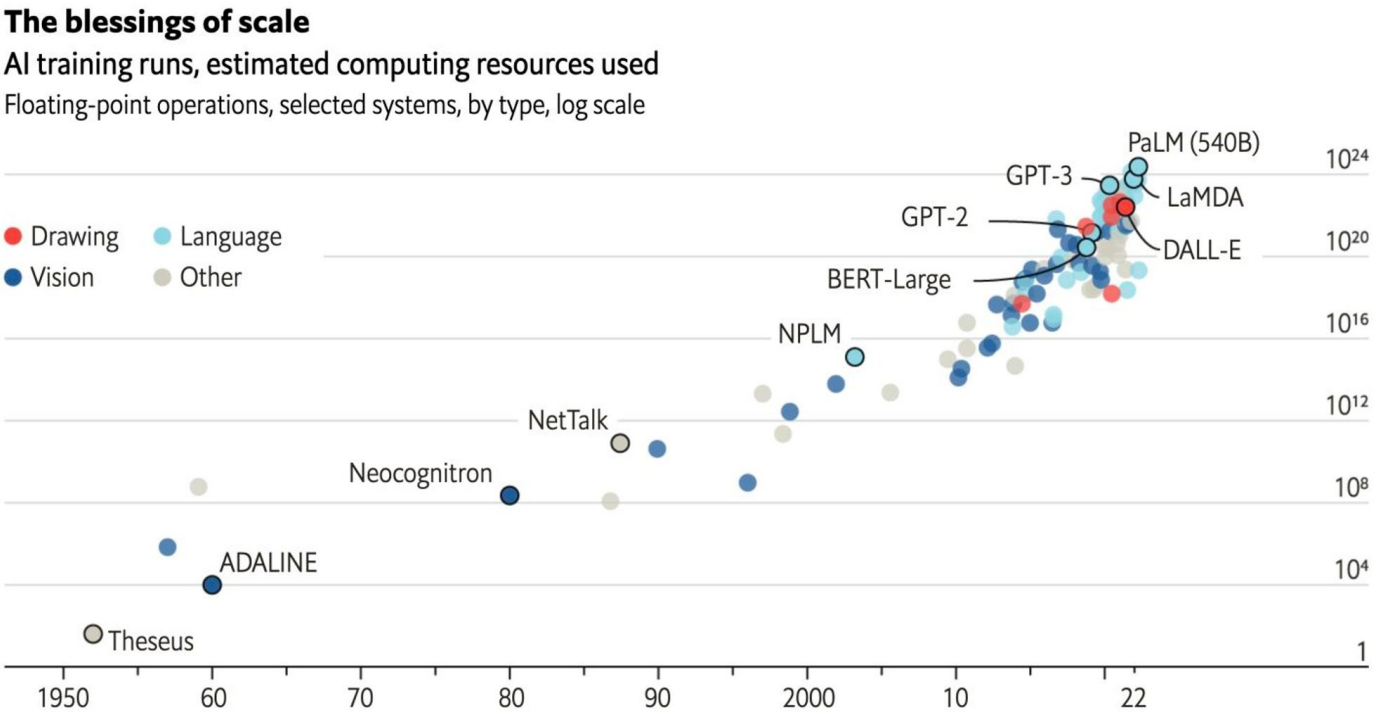


https://lilianweng.github.io/lil-log/2019/01/31/generalized-language-models.html

In the case of ChatGPT the generated numbers are probabilities. ChatGPT has a limited vocabulary, and the probabilities indicate how likely each vocabulary word is based on the input word sequence. ChatGPT has a limited reading range, and the input sequence has a maximum length of about 3000 words, broken into 4000 sub-word tokens. Once ChatGPT generates a word, it adds that word to the input sequence, and generates a new word. This process continues until it produces a special word called a “stop” token, or it hits a preset word limit.

Why the reading range is limited?

Pre-training



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:[8] 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

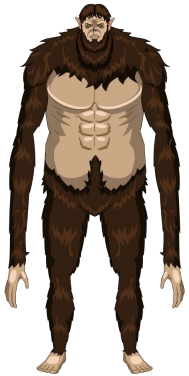
GPT4 was released this week. The estimated model size is around 175B to 280B.

Pre-training

GPT-1

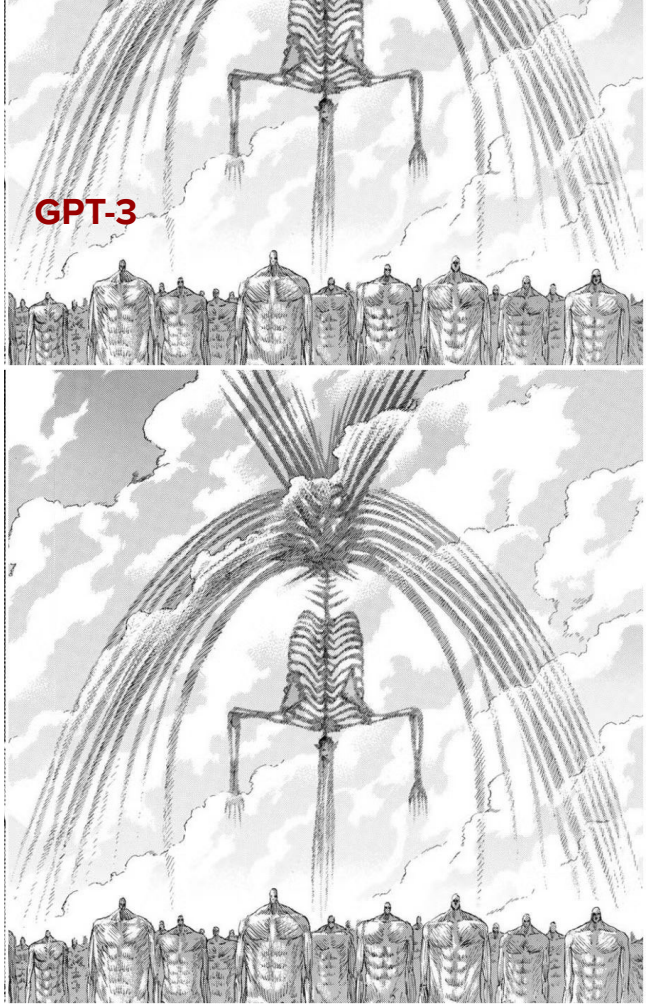
Levi 兵長 1.6m

GPT-2



Beast Titan 獣の巨人 17m

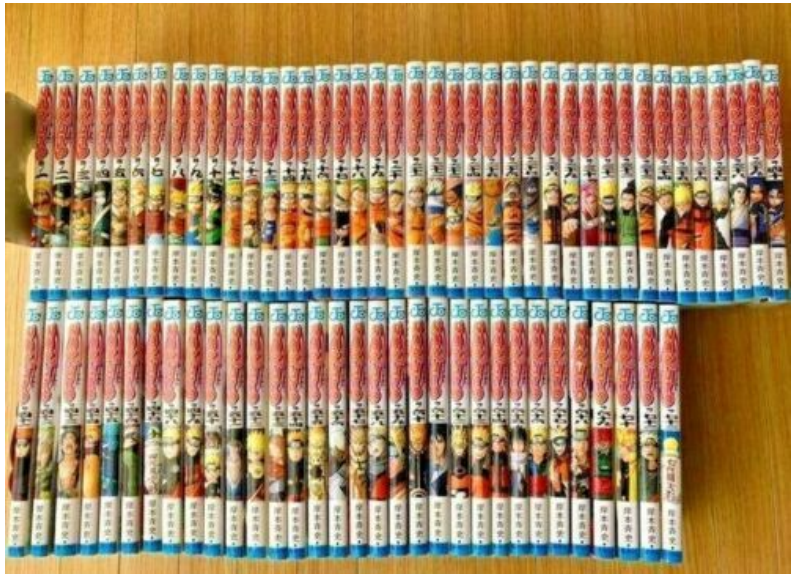
GPT-3



Two Stacked Eren Founding Titan 始祖の巨人 2*1000m

How large is 570GB

Read Naruto 270K times



Only lines/text are counted.

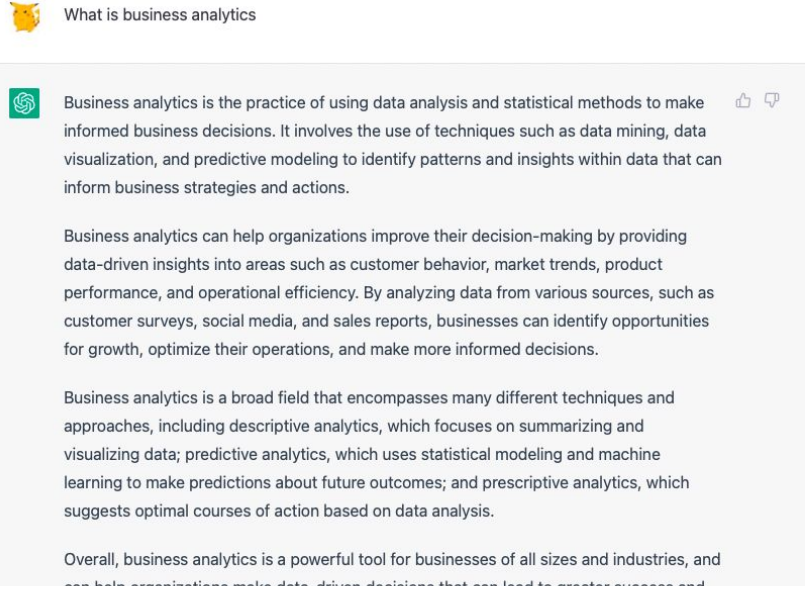
GPT3 is already powerful

- 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

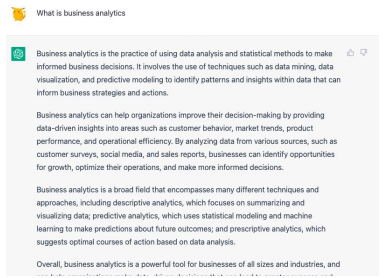
6. From GPT to ChatGPT

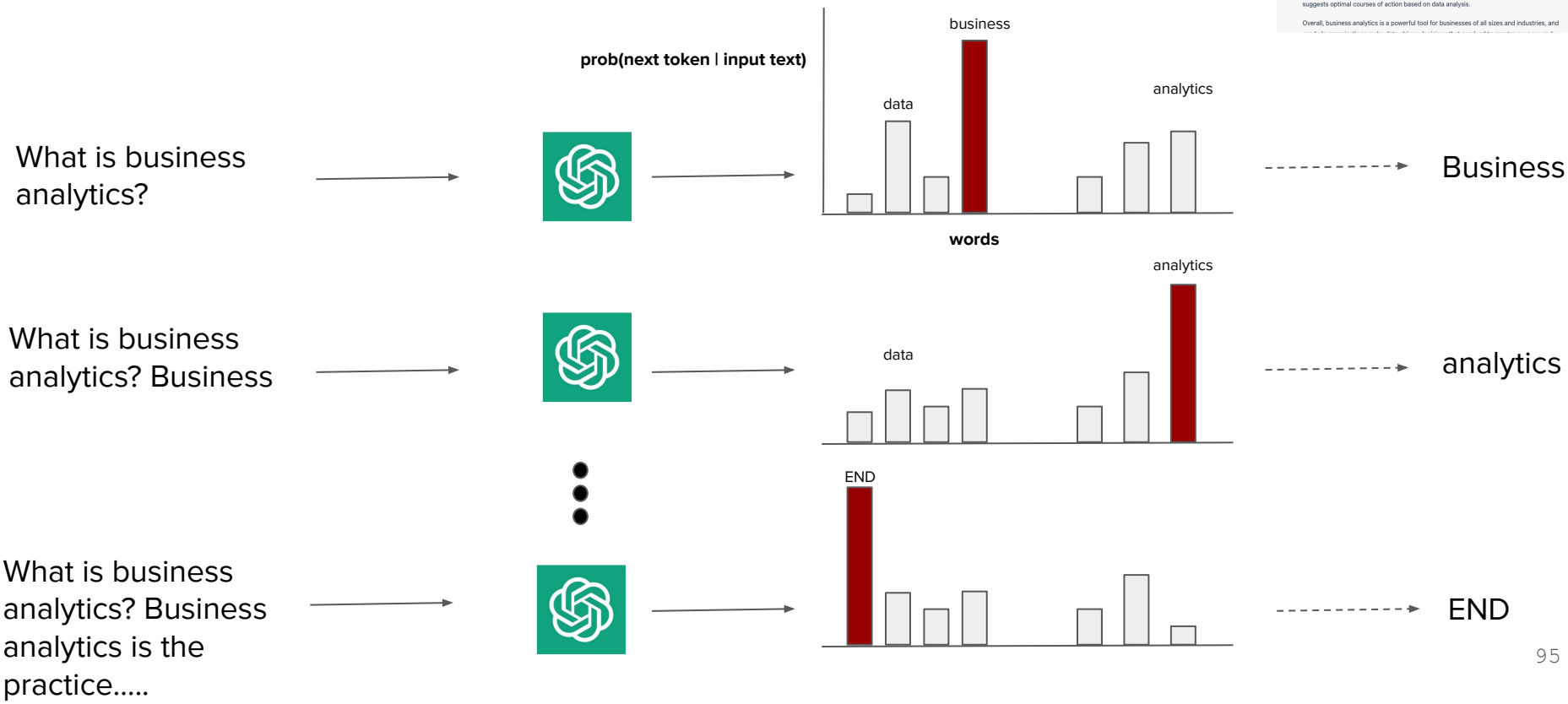
What is ChatGPT

- 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



Language model

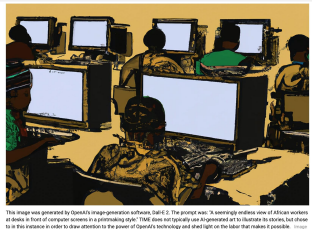




How to train ChatGPT

Pre-training
GPT GPT2 GPT3

Fine-tuning
InstructGPT ChatGPT



https://time.com/6247678/openai-chatgpt-kenya-workers/

Unannotated data

Human labelled data

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?

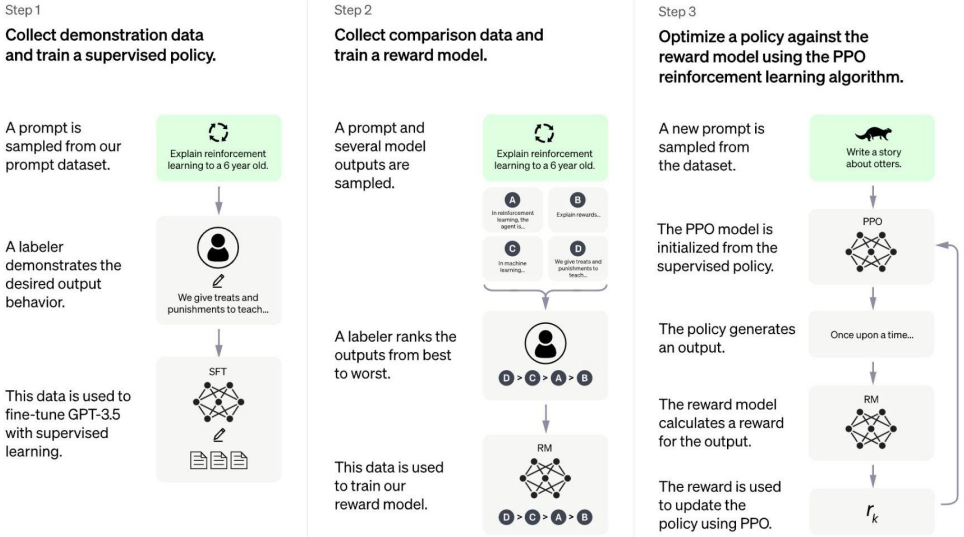
```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```

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† Peter Welinder Paul Christiano*†
Jan Leike* Ryan Lowe*
OpenAI

Abstract

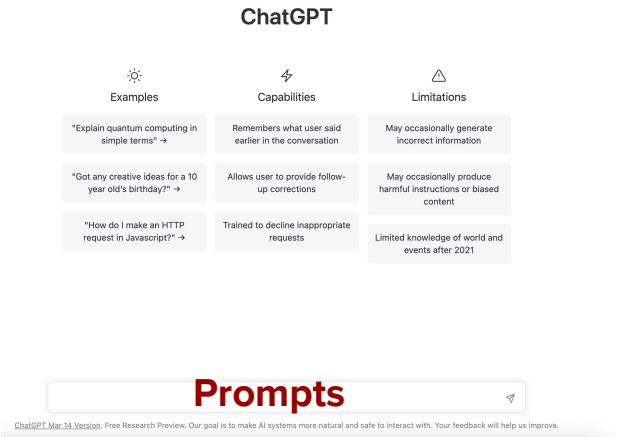
Methods

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

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:

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

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

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.

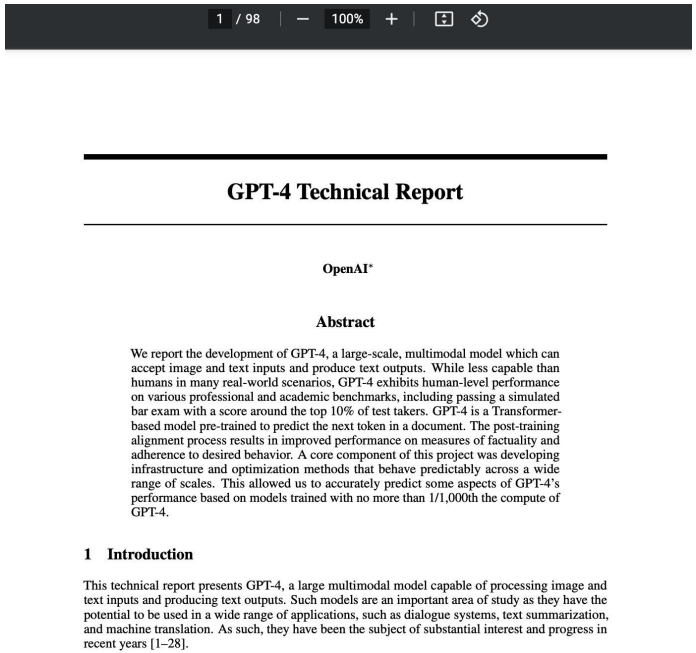
A. to store the value of C[0]
B. to store the value of C[1]
C. to store the value of C[i]
D. to store the value of C[i - 1]

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



https://cdn.openai.com/papers/gpt-4.pdf

GPT4

- Accept visual inputs now
  - https://www.youtube.com/watch?v=outcGtbnMuQ&t=971s
- Can handle up to 32 K words (3k for ChatGPT)
- Show the model performances

Exam	Contam	GPT-4 (no vision)	Non-contaminated GPT-4 (no vision)	GPT-4	Non-contaminated GPT-4
Uniform Bar Exam (MBE+MEE+MPT)	0 %	298 / 400 (~90th)	298 / 400 (~90th)	298 / 400 (~90th)	298 / 400 (~90th)
LSAT	39 %	161 (~83rd)	167 (~95th)	163 (~88th)	169 (~97th)
SAT Evidence-Based Reading & Writing	12 %	710 / 800 (~93rd)	710 / 800 (~93rd)	710 / 800 (~93rd)	710 / 800 (~93rd)
SAT Math	7 %	700 / 800 (~89th)	690 / 800 (~89th)	710 / 800 (~91st)	700 / 800 (~89th)
GRE Quantitative	35 %	157 / 170 (~62nd)	161 / 170 (~75th)	163 / 170 (~80th)	165 / 170 (~85th)
GRE Verbal	25 %	166 / 170 (~97th)	165 / 170 (~96th)	169 / 170 (~99th)	169 / 170 (~99th)
GRE Writing	100 %	4 / 6 (~54th)	N/A	4 / 6 (~54th)	N/A
USABO Semifinal Exam 2020	3 %	87 / 150 (99th - 100th)	87 / 150 (99th - 100th)	87 / 150 (99th - 100th)	87 / 150 (99th - 100th)
USNCO Local Section Exam 2022	5 %	38 / 60	38 / 60	36 / 60	36 / 60
Medical Knowledge Self-Assessment Program	19 %	75 %	75 %	75 %	75 %
Codeforces Rating	0 %	392 (below 5th)	392 (below 5th)	392 (below 5th)	392 (below 5th)
AP Art History	17 %	5 (86th - 100th)	5 (86th - 100th)	5 (86th - 100th)	5 (86th - 100th)
AP Biology	1 %	5 (85th - 100th)	5 (85th - 100th)	5 (85th - 100th)	5 (85th - 100th)
AP Calculus BC	3 %	4 (43rd - 59th)	4 (43rd - 59th)	4 (43rd - 59th)	4 (43rd - 59th)
AP Chemistry	16 %	4 (71st - 88th)	4 (71st - 88th)	4 (71st - 88th)	4 (71st - 88th)
AP Eng. Lang. and Comp.	79 %	2 (14th - 44th)	N/A	2 (14th - 44th)	N/A
AP Eng. Lit. and Comp.	92 %	2 (8th - 22nd)	N/A	2 (8th - 22nd)	N/A
AP Environmental Science	4 %	5 (91st - 100th)	5 (91st - 100th)	5 (91st - 100th)	5 (91st - 100th)
AP Macroeconomics	9 %	5 (84th - 100th)	5 (84th - 100th)	5 (84th - 100th)	5 (84th - 100th)
AP Microeconomics	2 %	4 (60th - 82nd)	5 (82nd - 100th)	5 (82nd - 100th)	5 (82nd - 100th)
AP Physics 2	12 %	4 (66th - 84th)	4 (66th - 84th)	4 (66th - 84th)	4 (66th - 84th)
AP Psychology	11 %	5 (83rd - 100th)	5 (83rd - 100th)	5 (83rd - 100th)	5 (83rd - 100th)
AP Statistics	13 %	5 (85th - 100th)	5 (85th - 100th)	5 (85th - 100th)	5 (85th - 100th)
AP US Government	24 %	5 (88th - 100th)	5 (88th - 100th)	5 (88th - 100th)	5 (88th - 100th)
AP US History	73 %	4 (74th - 89th)	4 (74th - 89th)	5 (89th - 100th)	5 (89th - 100th)
AP World History	47 %	5 (87th - 100th)	4 (65th - 87th)	4 (65th - 87th)	4 (65th - 87th)
AMC 10	4 %	36 / 150 (10th - 19th)	38 / 150 (14th - 21st)	30 / 150 (6th - 12th)	31 / 150 (7th - 12th)
AMC 12	4 %	48 / 150 (19th - 40th)	50 / 150 (26th - 44th)	60 / 150 (45th - 66th)	62 / 150 (52nd - 68th)
Introductory Sommelier (theory knowledge)	5 %	92 %	92 %	92 %	92 %
Certified Sommelier (theory knowledge)	9 %	86 %	86 %	86 %	86 %
Advanced Sommelier (theory knowledge)	4 %	77 %	77 %	77 %	77 %
Leetcode (easy)	0 %	31 / 41	31 / 41	31 / 41	31 / 41
Leetcode (medium)	0 %	21 / 80	21 / 80	21 / 80	21 / 80
Leetcode (hard)	0 %	3 / 45	3 / 45	3 / 45	3 / 45

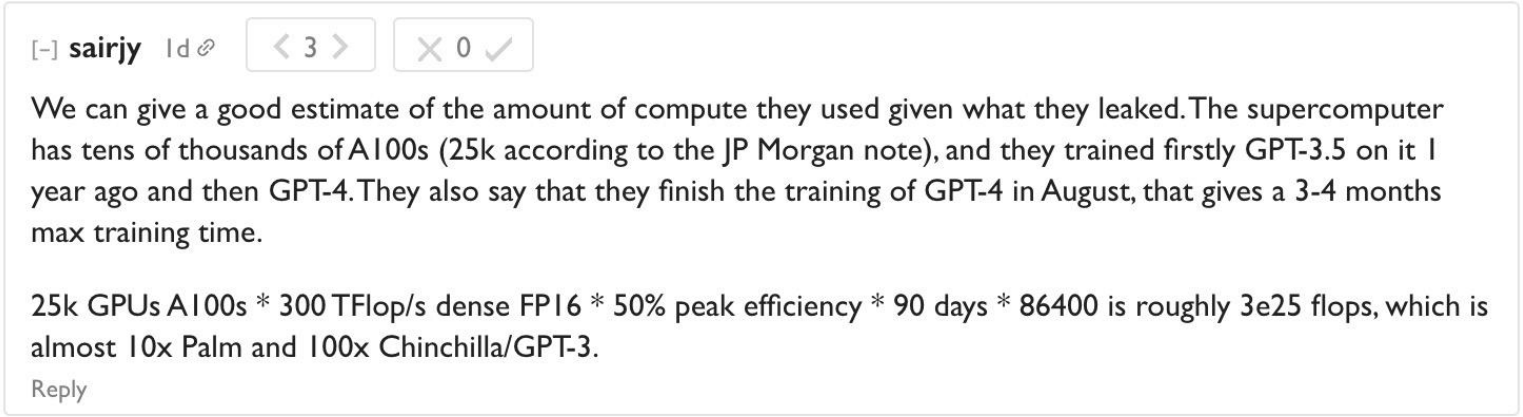
GPT4

- OpenAI will become close AI
  - The model was already developed last August

> When asked why OpenAI changed its approach to sharing its research, Sutskever replied simply, "We were wrong. Flat out, we were wrong. If you believe, as we do, that at some point, AI — AGI — is going to be extremely, unbelievably potent, then it just does not make sense to open-source. It is a bad idea... I fully expect that in a few years it's going to be completely obvious to everyone that open-sourcing AI is just not wise."

Source:
https://www.theverge.com/2023/3/15/23640180/openai-gpt-4-launch-closed-research-ilya-sutskever-interview?continueFlag=9a5809e532eaf8f52c6c050b322114d0

GPT4



[–] **sairjy** 1d 3 0

We can give a good estimate of the amount of compute they used given what they leaked. The supercomputer has tens of thousands of A100s (25k according to the JP Morgan note), and they trained firstly GPT-3.5 on it 1 year ago and then GPT-4. They also say that they finish the training of GPT-4 in August, that gives a 3-4 months max training time.

25k GPUs A100s * 300 TFlop/s dense FP16 * 50% peak efficiency * 90 days * 86400 is roughly 3e25 flops, which is almost 10x Palm and 100x Chinchilla/GPT-3.

Reply

Source:
https://www.lesswrong.com/posts/pckLdSgYWJ38NBFf8/gpt-4?commentId=2mKqGJnf2aTfQMZDq&continueFlag=9a5809e532eaf8f52c6c050b322114d0#2mKqGJnf2aTfQMZDq

GPT4

- Capability prediction
  - “Having a sense of capabilities of a model before training can improve decision around alignment, safety, and deployment.”

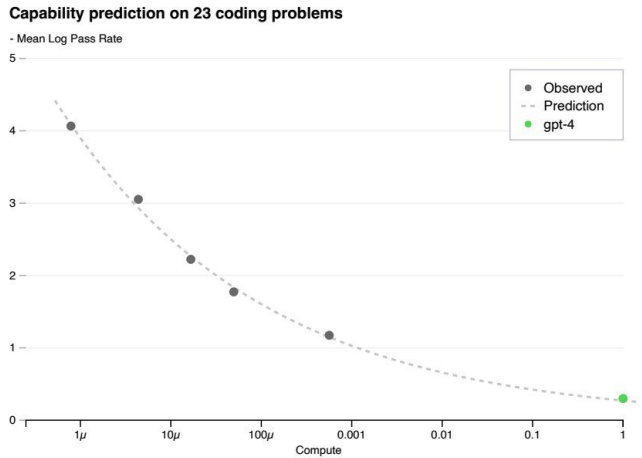
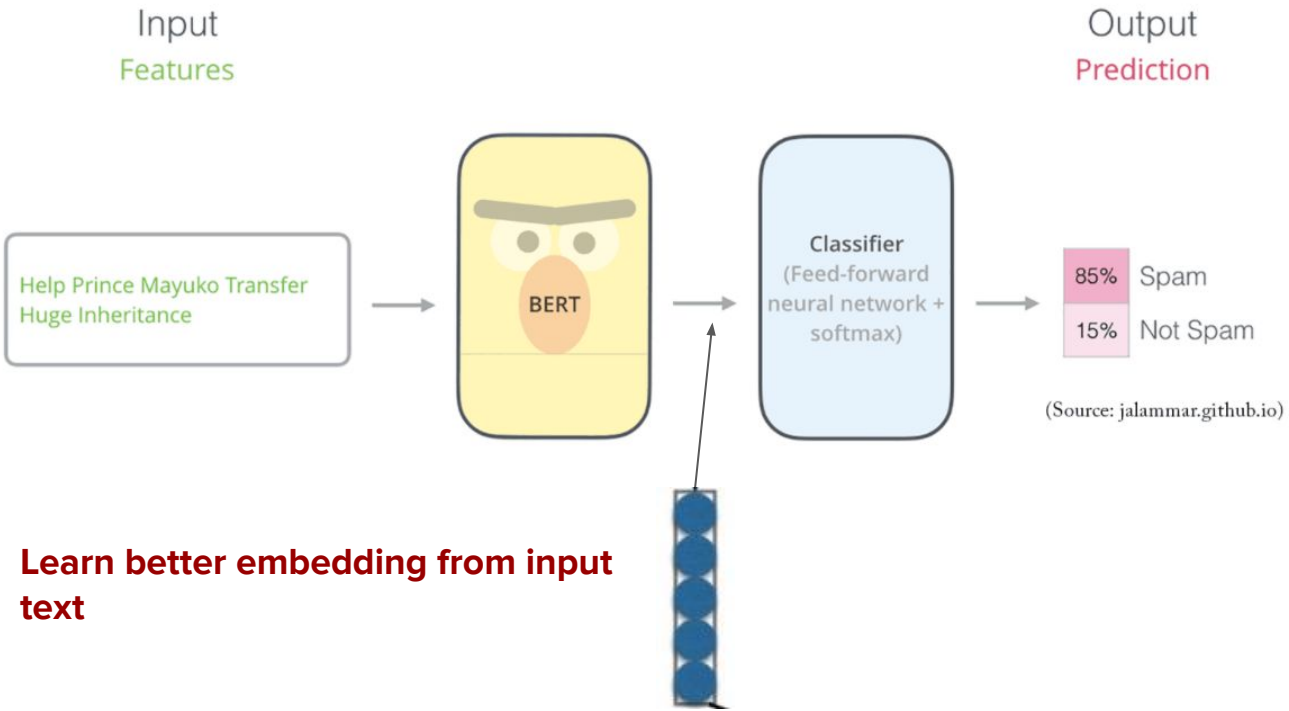


Figure 2. Performance of GPT-4 and smaller models. The metric is mean log pass rate on a subset of the HumanEval dataset. A power law fit to the smaller models (excluding GPT-4) is shown as the dotted line; this fit accurately predicts GPT-4’s performance. The x-axis is training compute normalized so that GPT-4 is 1.

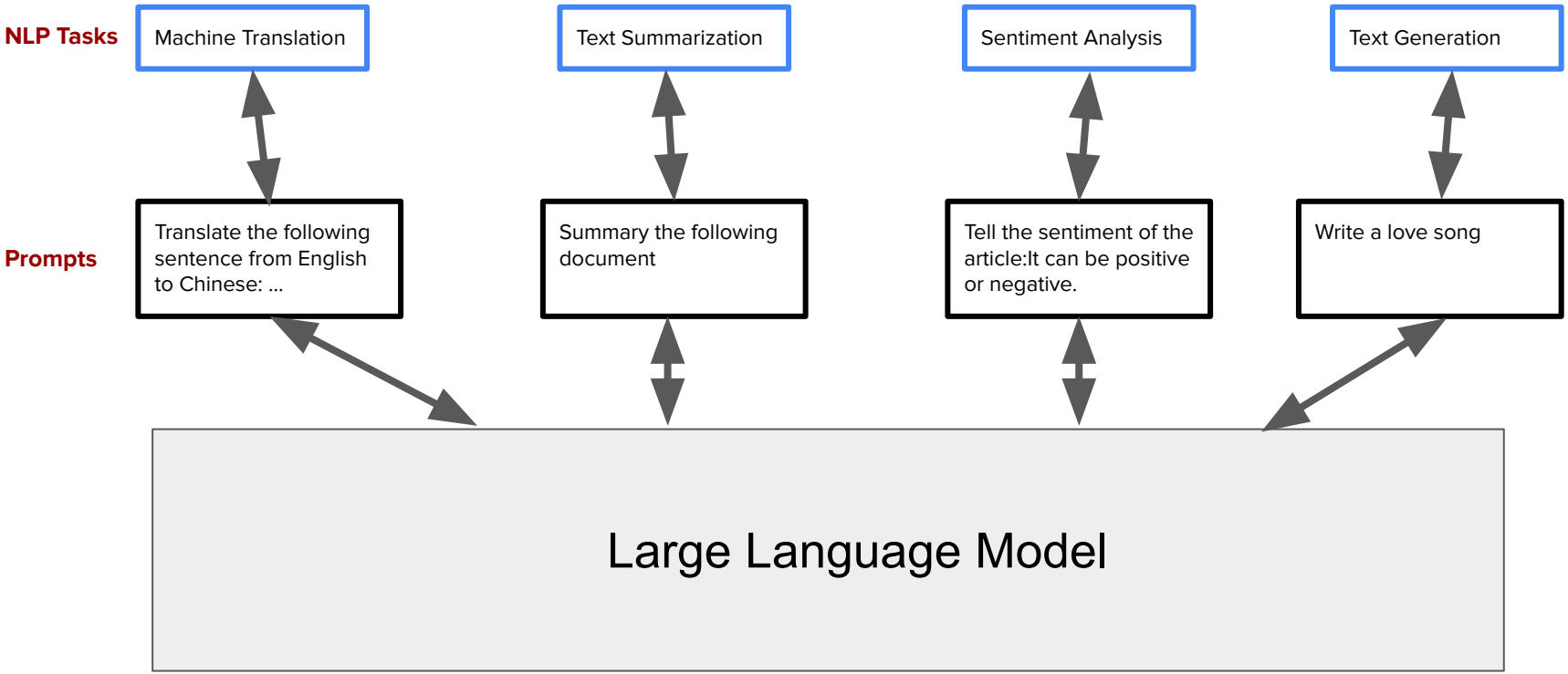
7. Fine-tuning vs Prompt Engineering

Bert + Fine-tuning



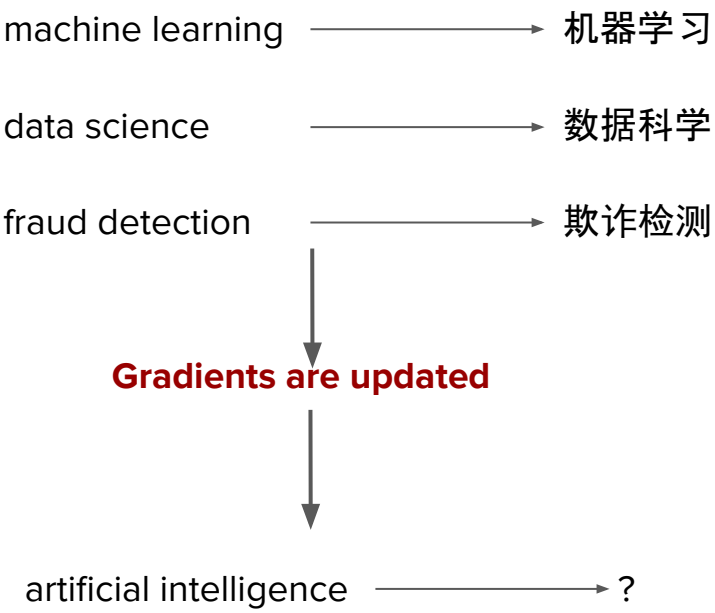
Learn better embedding from input text

GPT + Prompt Engineering



Fine-tuning vs Prompt

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



Prompt (few-shot learning here)

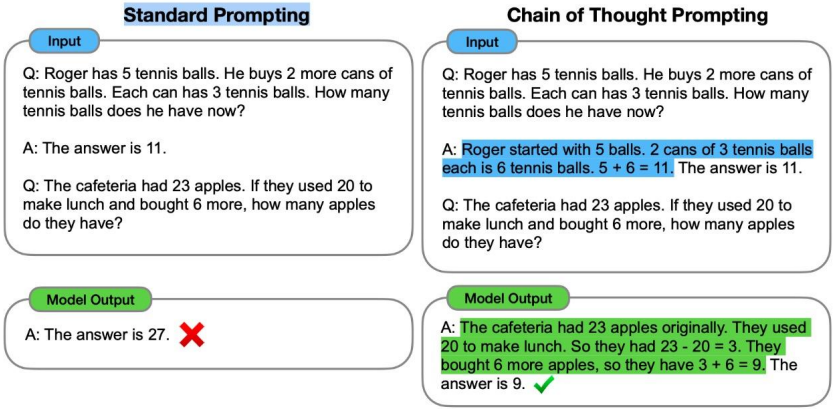
Translate English to Chinese
machine learning -> 机器学习
data science -> 数据科学
fraud detection -> 欺诈检测
artificial intelligence ->

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.



Source: https://arxiv.org/abs/2201.11903

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://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