tial Representation

Word	Number
a	1
able	2
about	3
...	...
hand	615
...	...
happy	621
...	...
zebra	1000

Tweet: This movie was almost good

[700 680 720 20 55]

Padding

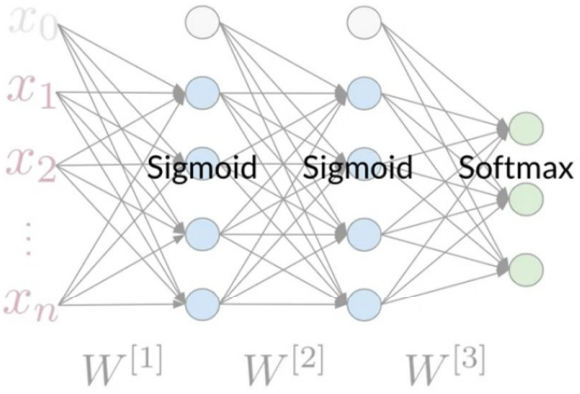
[700 680 720 20 55 0 0 0 0 0 0

To match size of longest tweet

3:22

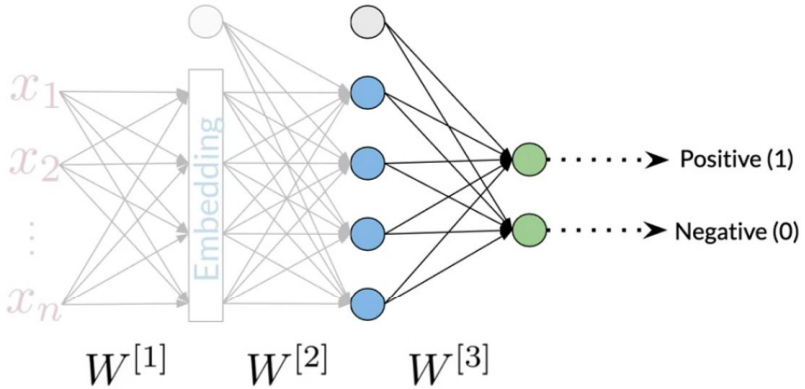
plearning.ai

ural Networks in Trax



plearning.ai

ural Networks for sentiment analysis

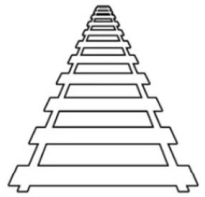


plearning.ai

tline

Define a basic neural network using Trax

Benefits of Trax



plearning.ai

vantages of using frameworks

Run fast on CPUs, GPUs and TPUs

Parallel computing

Record algebraic computations for gradient evaluation

Tensorflow Pytorch JAX

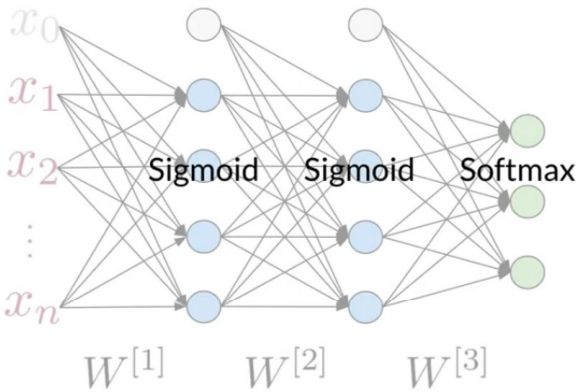
TensorFlow → Translate → Tensor2Tensor → Trax

TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,
reg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow
ndrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser,
Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray,
Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar,
Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals,
Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research*

around 2014-15 when we were making TensorFlow.

ural Networks in Trax




```
from trax import layers a
Model = tl.Serial(
        tl.Dense(4),
        tl.Sigmoid(),
        tl.Dense(4),
        tl.Sigmoid(),
        tl.Dense(3),
        tl.Softmax())
```


nmary

Order of computation ⟶ Model in Trax

Benefits from using frameworks

Trax

Deep Learning with Clear Code and Speed

Let me tell you why, so,

Trax makes programmers efficient

- Bottom-up clean re-design
- Easy to debug, you can read the code
- Full models with dataset bindings included
- Main models regression-tested daily

Is there a price to pay?

- Backwards compatibility

Trax code and understand what's going to come.

TensorFlow → Translate → Tensor2Tensor → Trax

Accelerating Deep Learning Research with the Tensor2Tensor Library

Monday, June 19, 2017

Posted by Łukasz Kaiser, Senior Research Scientist, Google Brain Team

Translation Model	Training time	BLEU (difference from baseline)
Transformer (T2T)	3 days on 8 GPU	**28.4 (+7.8)**
SliceNet (T2T)	6 days on 32 GPUs	26.1 (+5.5)
GNMT + Mixture of Experts	1 day on 64 GPUs	26.0 (+5.4)
ConvS2S	18 days on 1 GPU	25.1 (+4.5)
GNMT	1 day on 96 GPUs	24.6 (+4.0)
ByteNet	8 days on 32 GPUs	23.8 (+3.2)
MOSES (phrase-based baseline)	N/A	20.6 (+0.0)

BLEU scores (highe… …on task.

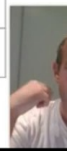
To do that, we created the Tensor2Tensor Library.

13:16

What do you want from a deep learning library?

- Makes programmers efficient
- Runs code fast

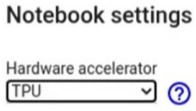
GPU (8x V100)	on-demand: $19.84 / hour	preemptible: $5.92 / hour
TPU (8x v3)	on-demand: $8 / hour	preemptible: $1.40 / hour



there are really two things that matters.

Trax runs code fast

- Designed to use a JIT compiler with JAX and XLA
- JAX: fastest Transformer in MLPerf 2020 (JAX: 0.26, TF: 0.35, pyTorch: 0.62)
- No code changes at all between CPU, GPU and TPU, preemptible training
- Tested with TPUs on colab too:



GPU (8x V100)	on-demand: $19.84 / hour	preemptible: $5.92 / hour
TPU (8x v3)	on-demand: $8 / hour	preemptible: $1.40 / hour

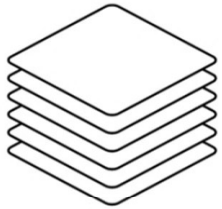
beautiful things where you

9:33

/ 13:16

tline

How classes work and their implementation



their methods.

plearning.ai

Trax Adam

```
class Adam(Optimizer):
 """Adam optimizer."""

 def init(self, param):
   m = np.zeros_like(param)
   v = np.zeros_like(param)
   return m, v

 def update(self, step, grads, param, slots, opt_params):
   m, v = slots
   learning_rate, b1, b2, eps = opt_params
   m = (1 - b1) * grads + b1 * m  # First  moment estimate.
   v = (1 - b2) * (grads ** 2) + b2 * v  # Second moment estimate.
   mhat = m / (1 - b1 ** (step + 1))  # Bias correction.
   vhat = v / (1 - b2 ** (step + 1))
   param = param - (
       learning_rate * mhat / (np.sqrt(vhat) + eps)).astype(param.dtype)
   return param, (m, v)
```

$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment esti
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estima
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while

is that you're using a new thing.

9:33

/ 13:16

rax is fun to learn and use!

```
f TransformerLM(vocab_size, d_model=512, d_ff=2048,
                n_layers=6, n_heads=8, dropout=0.1,
                max_len=2048, mode='train'):
""Returns a Transformer language model. """
nbedder = [
   tl.Embedding(d_model, vocab_size),
   tl.Dropout(rate=dropout, mode=mode),
   tl.PositionalEncoding(max_len=max_len),

eturn tl.Serial(
   tl.ShiftRight(),
   embedder,
   [DecoderBlock(d_model, d_ff, n_heads,
                 dropout, mode)
    for _ in range(n_layers)],
   tl.LayerNorm(),
   tl.Dense(vocab_size),
   tl.LogSoftmax(),
```

github.com/google/trax

```
# Create a Transformer model.
# Pre-trained model config in gs://trax-ml/models/translation/ende_wmt32k.gi
model = trax.models.Transformer(
    input_vocab_size=33300,
    d_model=512, d_ff=2048,
    n_heads=8, n_encoder_layers=6, n_decoder_layers=6,
    max_len=2048, mode='predict')

# Initialize using pre-trained weights.
model.init_from_file('gs://trax-ml/models/translation/ende_wmt32k.pkl.gz',
                     weights_only=True)

# Tokenize a sentence.
sentence = 'It is nice to learn new things today!'
tokenized = list(trax.data.tokenize(iter([sentence]),  # Operates on streams
                                    vocab_dir='gs://trax-ml/vocabs/',
                                    vocab_file='ende_32k.subword'))[0]

# Decode from the Transformer.
tokenized = tokenized[None, :]  # Add batch dimension.
tokenized_translation = trax.supervised.decoding.autoregressive_sample(
    model, tokenized, temperature=0.0)  # Higher temperature: more diverse r

# De-tokenize,
tokenized_translation = tokenized_translation[0][:-1]  # Remove ba
translation = trax.data.detokenize(tokenized_translation,
                                   vocab_dir='gs://trax-ml/vocabs/
                                   vocab_file='ende_32k.subword')
```

because we had the liberty

to do things from scratch

/ 13:16

ısses in Python

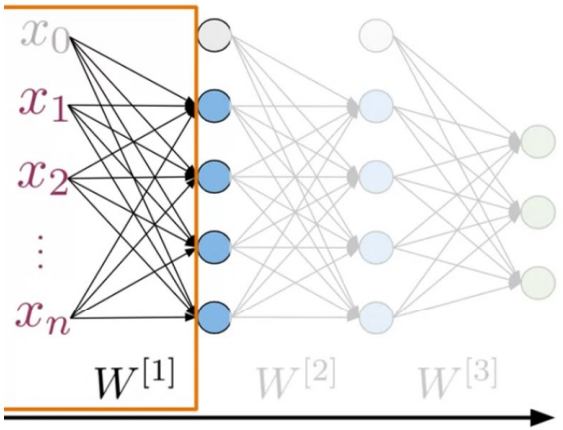
```
lass MyClass(Object):
    def __init__(self, y):
        self.y = y
    def my_method(self,x):
        return x + self.y
    def __call__(self, x):
        return self.my_method(x)
```

```
f = MyClass(7)
```

the program will return
the results from my_method.

plearning.ai

nse Layer



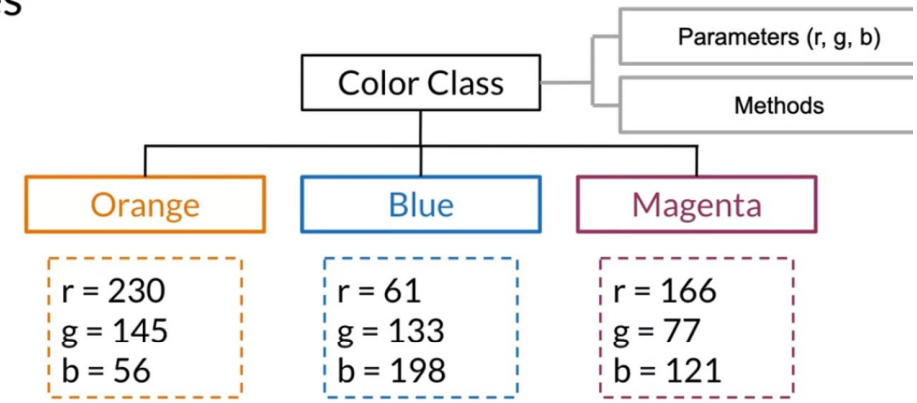
Fully connected laye

$$z^{[i]} = W^{[i]}a^{[$$

You will have to compute
the inner products between

plearning.ai

ısses



magenta, would be defined with
a specific set of r, g, b values.

plearning.ai

ɔclasses

```
ıss MyClass(Object):
    def __init__(self,y):
        self.y = y
    def my_method(self,x):
        return x + self.y
    def __call__(self,x):
        return self.my_method(x)
```

```
class SubClass(MyClass):
    def my_method(self,x):
        return x + self.y**
```

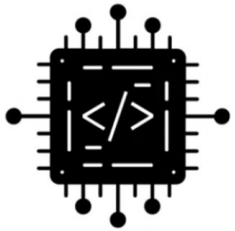
will be inherited from the parent class,
but my_method will be replaced.

plearning.ai

utline

Embedding layer

Mean layer



the mean layer in serial models

eplearning.ai

bedding Layer

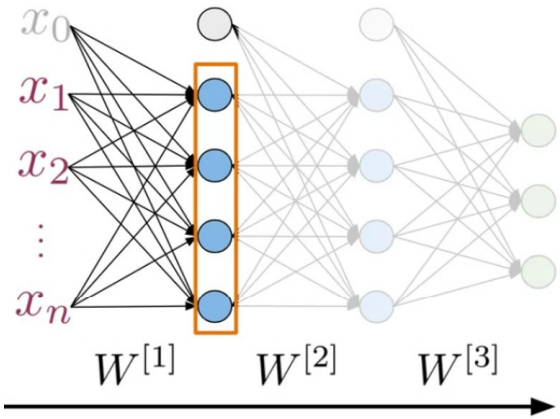
Vocabulary	Index		
I	1	0.020	0.006
am	2	-0.003	0.010
happy	3	0.009	0.010
because	4	-0.011	-0.018
learning	5	-0.040	-0.047
NLP	6	-0.009	0.050
sad	7	-0.044	0.001
not	8	0.011	-0.022

Traina
weigh

When using an embedding
layer in your model,

plearning.ai

LU Layer



ReLU = Rectified linear

$$a^{[i]} = g(z^{[i]})$$

that typically follows a
dense fully connected layer,

plearning.ai

bedding Layer

Vocabulary	Index		
I	1	0.020	0.006
am	2	-0.003	0.010
happy	3	0.009	0.010
because	4	-0.011	-0.018
learning	5	-0.040	-0.047
NLP	6	-0.009	0.050
sad	7	-0.044	0.001
not	8	0.011	-0.022

we'll return a vector
equal to 0.020,

plearning.ai

ean Layer

et: I am happy

Vocabulary	Index		
I	1	0.020	0.006
am	2	-0.003	0.010
happy	3	0.009	0.010

0.020	0.006
-0.003	0.010
0.009	0.010

Mean of the word embeddings

0.009
0.009

the same number of features as you're embedding size.

plearning.ai

mputing gradients in Trax

$$f(x) = 3x^2 + x$$

$$\frac{\delta f(x)}{\delta x} = 6x + 1$$

radient

```
def f(x):
    return 3*x**2 + x

grad_f = trax.math.grad(f)
```

Returns a function

The function grad will return a function

0:54

2:34

plearning.ai

bedding Layer

Vocabulary	Index		
I	1	0.020	0.006
am	2	-0.003	0.010
happy	3	0.009	0.010
because	4	-0.011	-0.018
learning	5	-0.040	-0.047
NLP	6	-0.009	0.050
sad	7	-0.044	0.001
not	8	0.011	-0.022

Traina weigh

Vocabu x Embedd

treated as a hyperparameter in your model.

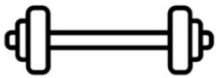
plearning.ai

Press Esc to exit full screen

tline

Computing gradients in trax

Training using grad()



allows you to easily compute gradients,

2:34

plearning.ai

ditional Language Models

J'ai vu le match de foot

Sequence	$P(\text{Seque}$
I saw the game of soccer	4.5 e-
I saw the soccer game	6.0 e-
I saw the soccer match	4.6 e-
Saw I the game of soccer	2.6 e-

For an accurate translation, you could compute the probabilities of each sentence

grams

$$P(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$ → Bigrams

$$P(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$ → Trigrams

$$P(w_1, w_2, w_3) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_2)$$

- Large N-grams to capture dependencies between distant words
- Need a lot of space and RAM

Even in the case of large corpora your model would need a lots of space and

Training - DeepLearn...

```
y = model(x)
grads = grad(y.forward)(y.weights,x)
```

In a loop

```
weights -= alpha*grads
```

Forward and Back-propagation

performed in a single line.

grams

$$P(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$ → Bigrams

$$P(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$ → Trigrams

$$P(w_1, w_2, w_3) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_2)$$

- Large N-grams to capture dependencies between distant words

One of them is that in order to capture dependencies between words

vantages of RNNs

r was supposed to study with me. I called her but she did not _____ answ

want
respond
choose
want
have
ask
attempt
answer
know

RNNs look at every previous word

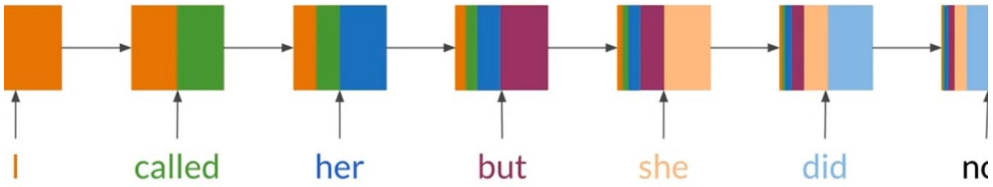
Similar probabilities with trigram

To see how recurrent
neural networks work,

plearning.ai

lNs Basic Structure

led her but she did not _____



represents the
computations made at

plearning.ai

lNs Basic Structure

led her but she did not _____

To see how recurrent
neural networks work,

plearning.ai

vantages of RNNs

r was supposed to study with me. I called her but she did not _____ answ

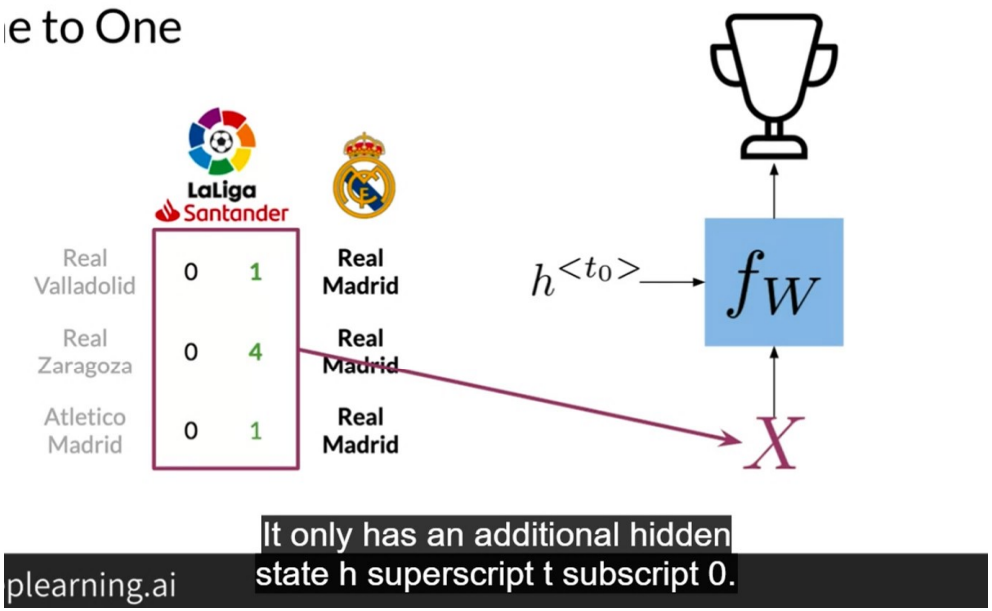
want
respond
choose
want
have
ask
attempt
answer
know

RNNs look at every previous word

Similar probabilities with trigram

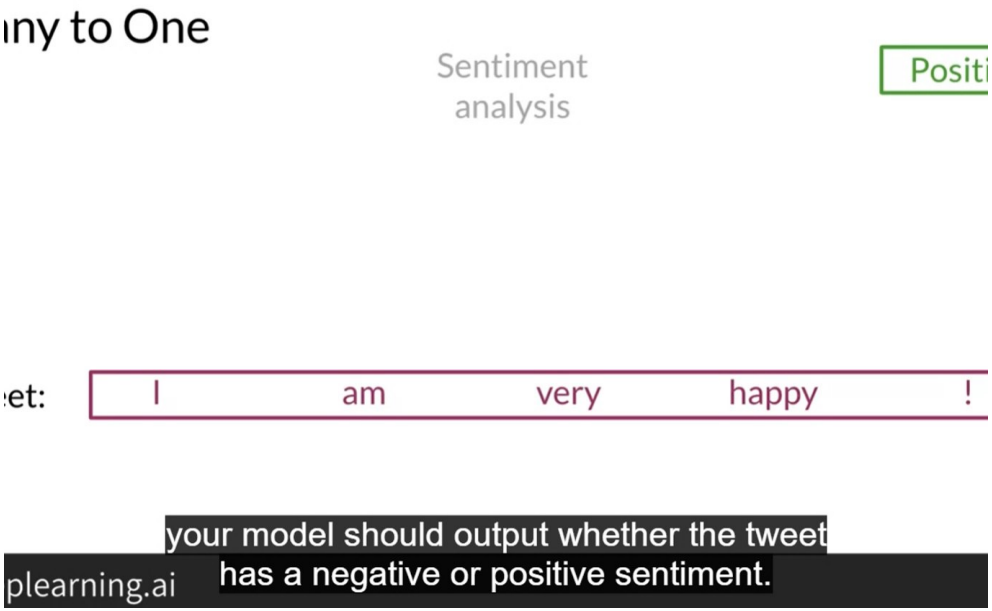
you would have to account
for six-word-long sequences,

plearning.ai



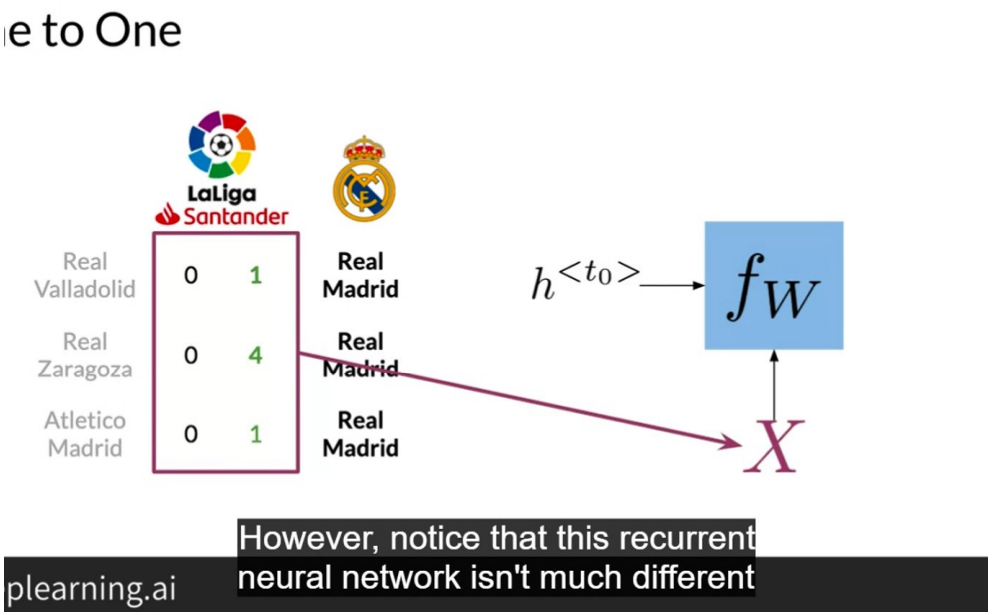
It only has an additional hidden state h superscript t subscript 0.

plearning.ai



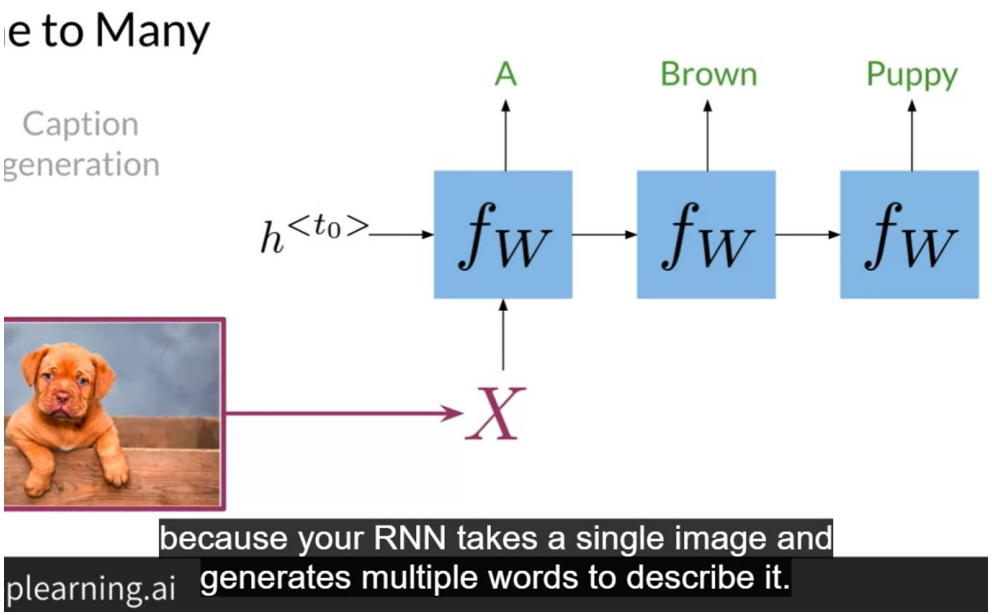
your model should output whether the tweet has a negative or positive sentiment.

plearning.ai



However, notice that this recurrent neural network isn't much different

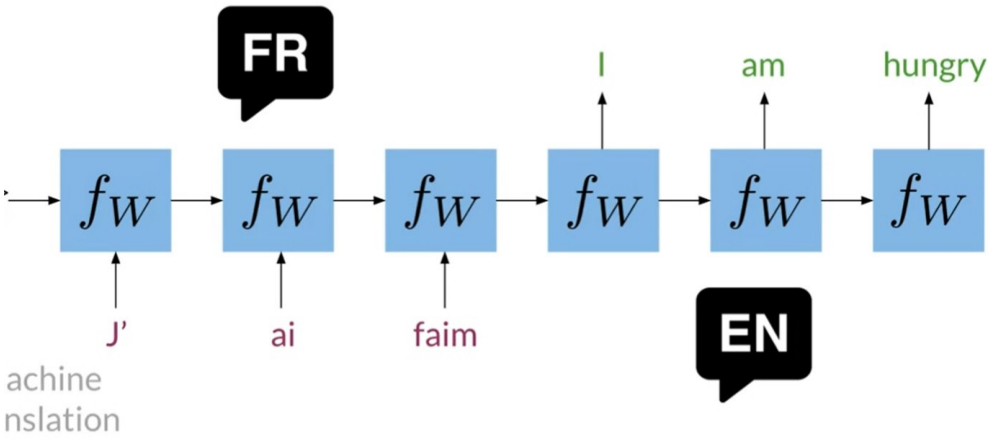
plearning.ai



because your RNN takes a single image and generates multiple words to describe it.

plearning.ai

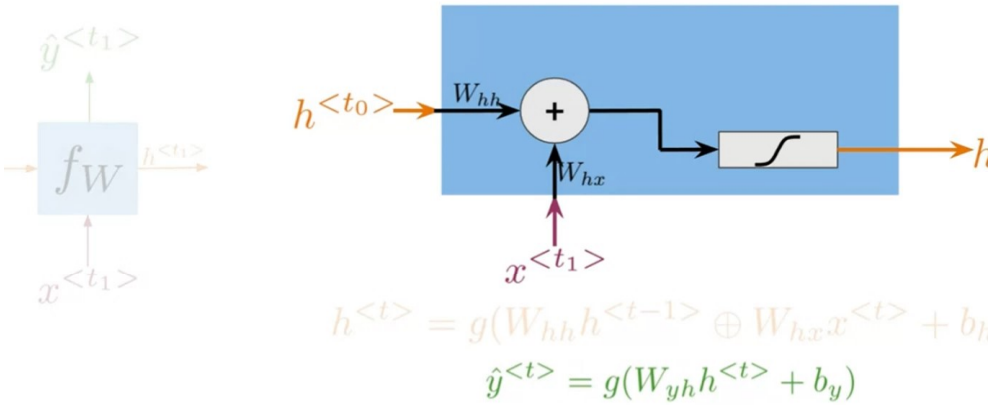
ıny to Many



this task, because they propagate
information from the beginning to end.

plearning.ai

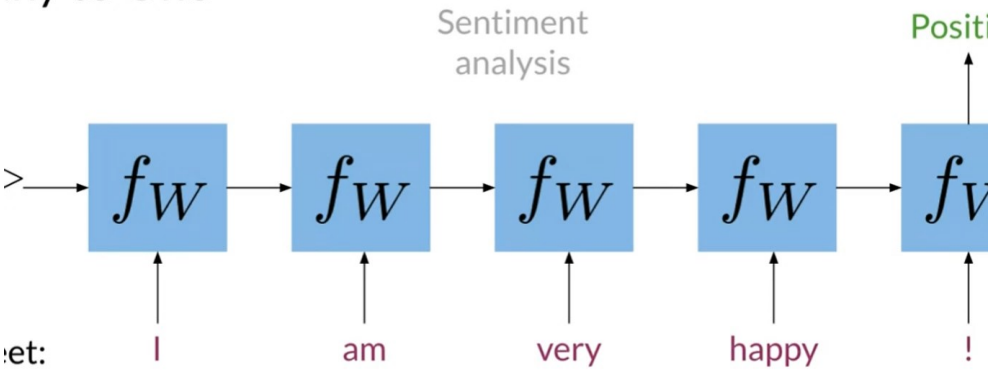
Vanilla RNN



you can compute the current
y-hat by multiplying

plearning.ai

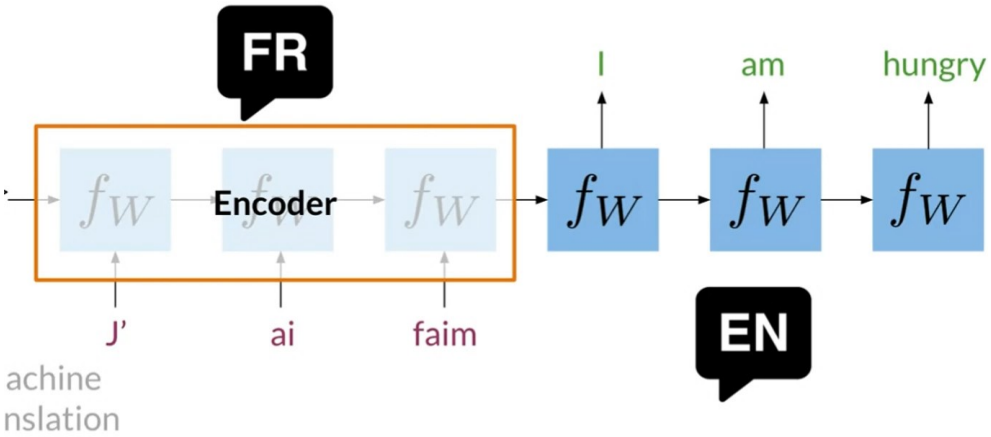
ıny to One



Your RNN would take every word from
the sequence as inputs in different steps,

plearning.ai

ıny to Many



machine translation.

plearning.ai

nmary

Hidden states propagate information through time

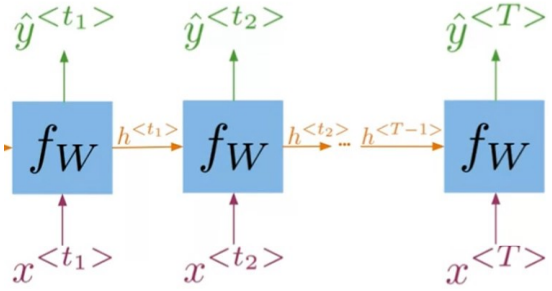
Basic recurrent units have two inputs at each time: $h^{<t-1>}$, $x^{<t}$

different positions within a sequence.

oss Entropy Loss

$$h^{<t>} = g(W_h[h^{<t-1>}, x^{<t>}$$

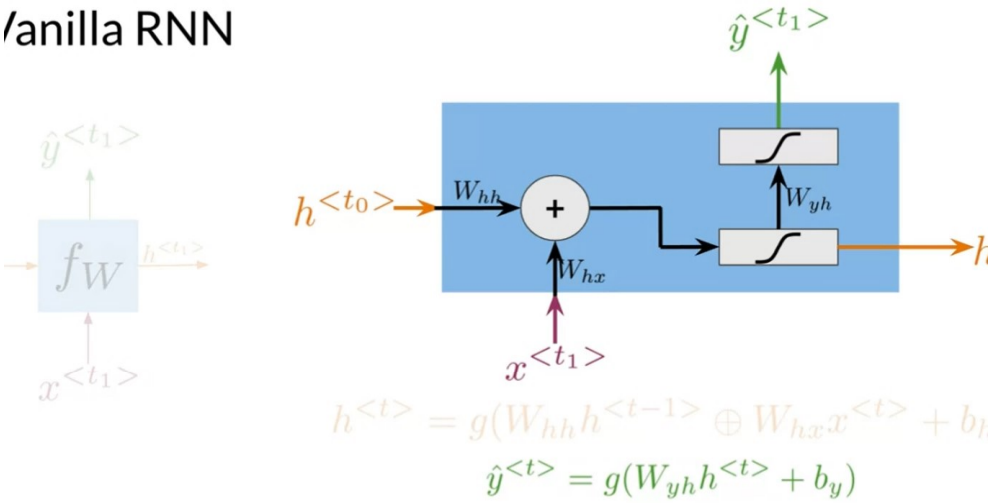
$$\hat{y}^{<t>} = g(W_{yh}h^{<t>} + b$$



$$J = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{K} y_j^{<t>} \log$$

the summation over time t and the division by the total number of steps, capital T.

Vanilla RNN

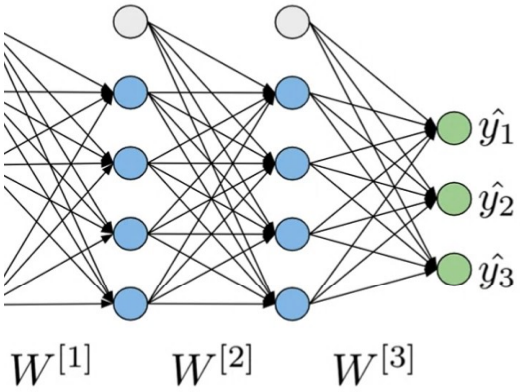


$$h^{<t>} = g(W_{hh}h^{<t-1>} \oplus W_{hx}x^{<t>} + b_h$$

$$\hat{y}^{<t>} = g(W_{yh}h^{<t>} + b_y)$$

w_y times h,

oss Entropy Loss



K - classes or possibilities

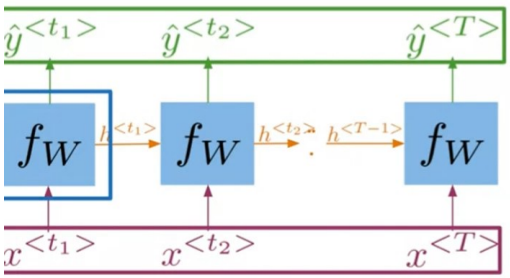
Either 0 o

$$J = -\sum_{j=1}^{K} y_j \log \hat{y}_j$$

Looking at a single example $(x$

output probability in y-hat and compare them with the true y values, 0 or 1.

can() function




```
def scan(fn, elems, initializer=None
    cur_value = initializer
    ys = []
    for x in elems:
        y, cur_value = fn(x, cur_v
        ys.append(y)
    return ys, cur_value
```

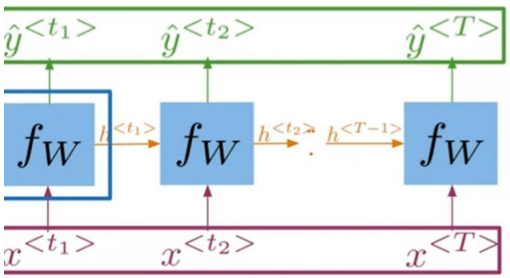
Finally, the function returns the list of predictions, and the last hidden state.

1:20

plearning.ai

2:02

can() function




```
def scan(fn, elems, initializer=None
    cur_value = initializer
    ys = []
    for x in elems:
        y, cur_value = fn(x, cur_v
        ys.append(y)
    return ys, cur_value
```

Finally, the function returns the list of predictions, and the last hidden state.

1:16

plearning.ai

2:02

tline

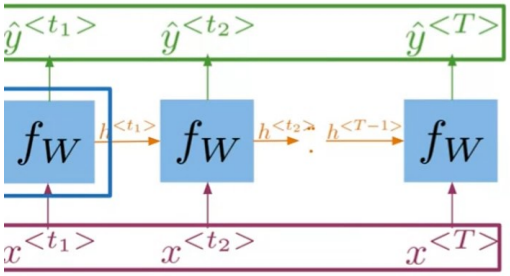
scan() function in tensorflow

Computation of forward propagation using abstractions

for computing the forward pass in RNNs.

plearning.ai

can() function




```
def scan(fn, elems, initializer=None
    cur_value = initializer
    ys = []
    for x in elems:
        y, cur_value = fn(x, cur_v
        ys.append(y)
```

stores the values of the prediction and hidden states.

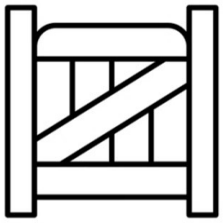
1:16

plearning.ai

2:02

)utline

- Gated recurrent unit (GRU) structure
- Comparison between GRUs and vanilla RNNs



GRUs for shorts, with a comparison to vanilla RNNs.

eeplearning.ai

;ated Recurrent Units

"Ants are really interesting. They are everywhere."

Plural

Relevance and update gates to remember important prior information

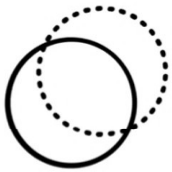
computing irrelevance and update gates,

eeplearning.ai

mmary

Frameworks require abstractions

tf.scan() mimics RNNs



It is important to know that these types of abstractions are needed for deep

plearning.ai

;ated Recurrent Units

"Ants are really interesting. They are everywhere."

Plural

keep the information about the subject.

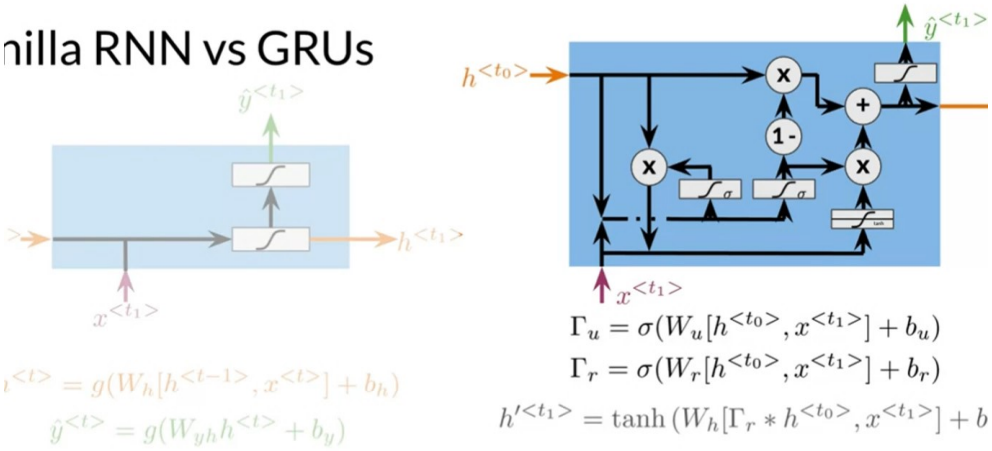
eeplearning.ai

nilla RNN vs GRUs

Let's compare GRUs with vanilla RNNs.

plearning.ai

nilla RNN vs GRUs

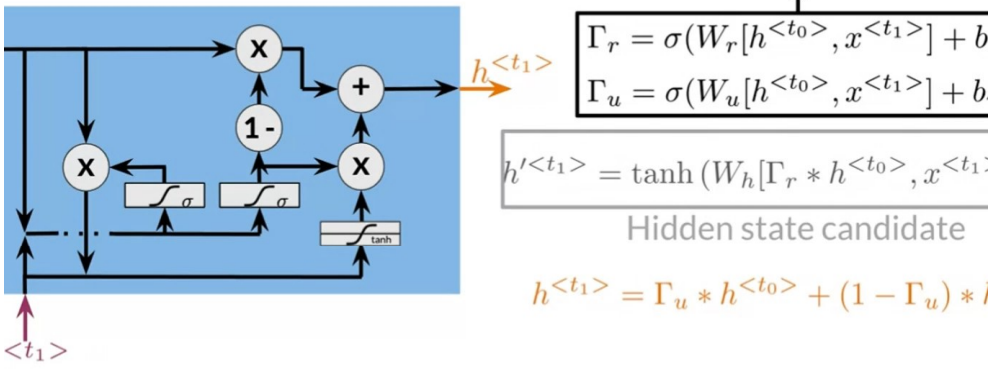


The current hidden state is computed and

plearning.ai

ted Recurrent Unit

Gates to keep/update releva information in the hidden sta

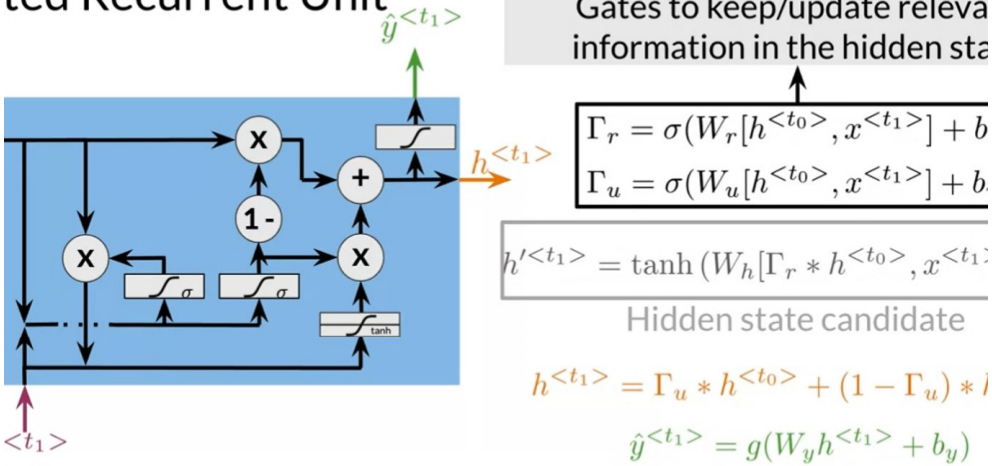


After that, a new value for the hidden state is

plearning.ai

ted Recurrent Unit

Gates to keep/update releva information in the hidden sta



Finally, a prediction y hat

2:38

4:26

plearning.ai

tline

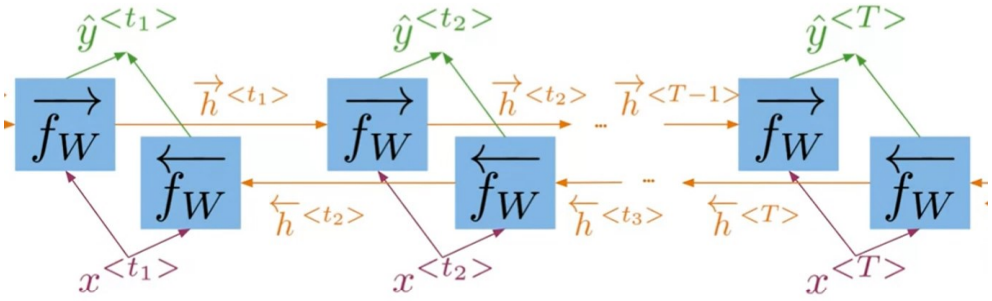
How bidirectional RNNs propagate information

Forward propagation in deep RNNs

how stacking RNNs together can produce a deep neural network.

plearning.ai

directional RNNs

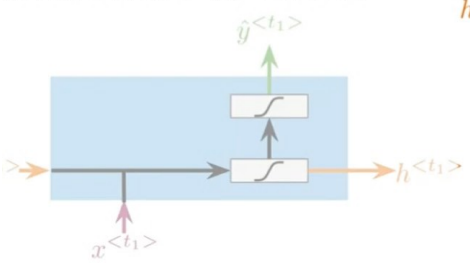


Information flows from the past and from the future **independently**

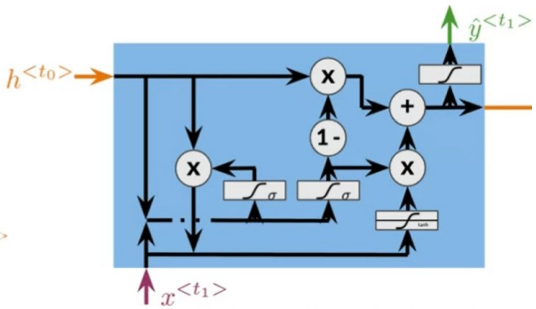
So the computations from left to right

plearning.ai

nilla RNN vs GRUs



$$\hat{y}^{<t>} = g(W_{yh}h^{<t>} + b_y)$$



$$\Gamma_u = \sigma(W_u[h^{<t_0>}, x^{<t_1>}] + b_u)$$
$$\Gamma_r = \sigma(W_r[h^{<t_0>}, x^{<t_1>}] + b_r)$$
$$h'^{<t_1>} = \tanh(W_h[\Gamma_r * h^{<t_0>}, x^{<t_1>}] + b$$
$$h^{<t_1>} = \Gamma_u * h^{<t_0>} + (1 - \Gamma_u) * h'^{<t_1>}$$
$$\hat{y}^{<t_1>} = g(W_y h^{<t_1>} + b_y)$$

All of these computations allow the network to learn

plearning.ai

directional RNNs

Press Esc to exit full screen

Bidirectional RNNs work in much the same way that simple RNNs do.

plearning.ai

ep RNNs

$$h^{[l]<t>} = f^{[l]}(W_h^{[l]}[h^{[l]<t-1>}, a^{[l-1]<t>}] -$$

$$a^{[l]<t>} = f^{[l]}(W_a^{[l]}h^{[l]<t>} +$$

But let's see how information flows in this case.

plearning.ai

lNs: Advantages

Captures dependencies within a short range

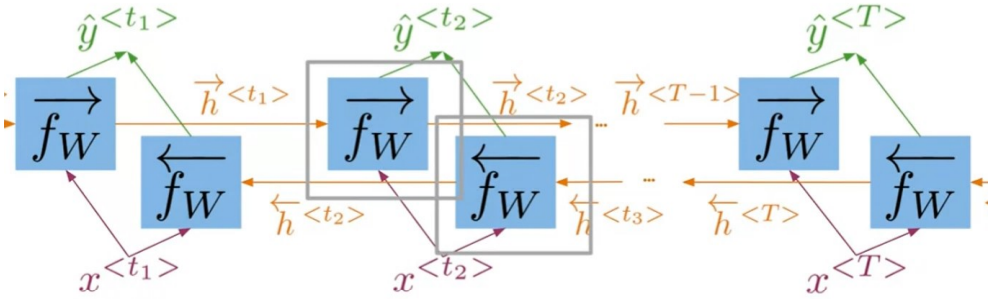
Takes up less RAM than other n-gram models

They're also relatively lightweight compared to other n-gram models,

0:41

plearning.ai

directional RNNs



Information flows from the past and from the future **independently**

When you have computed both of the hidden states for a time step,

plearning.ai

tline

RNNs and vanishing/exploding gradients

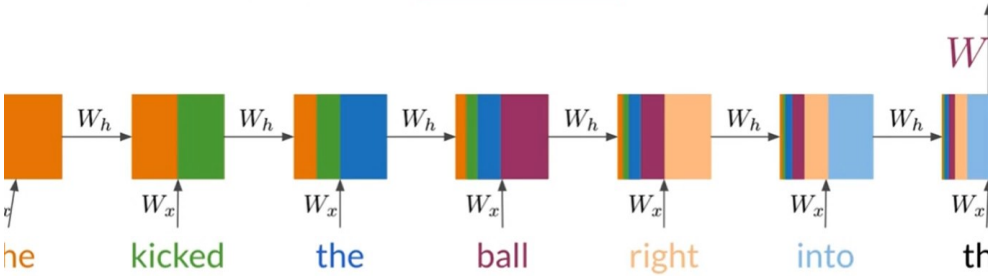
Solutions

a problem common to RNNs, and then demonstrate a few ways to handle them.

plearning.ai

IN Basic Structure

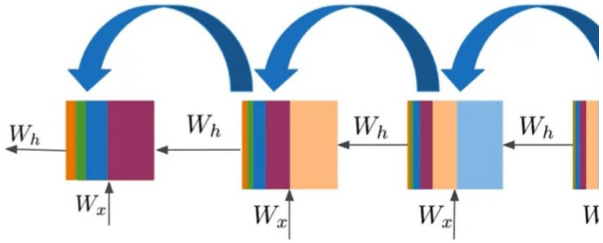
kicked the ball right into the _____



propagates information from the beginning of the sequence through to the end.

plearning.ai

e vanishing gradient problem



at the early layers as the product of all the terms from the later layers makes for

plearning.ai

INs: Disadvantages

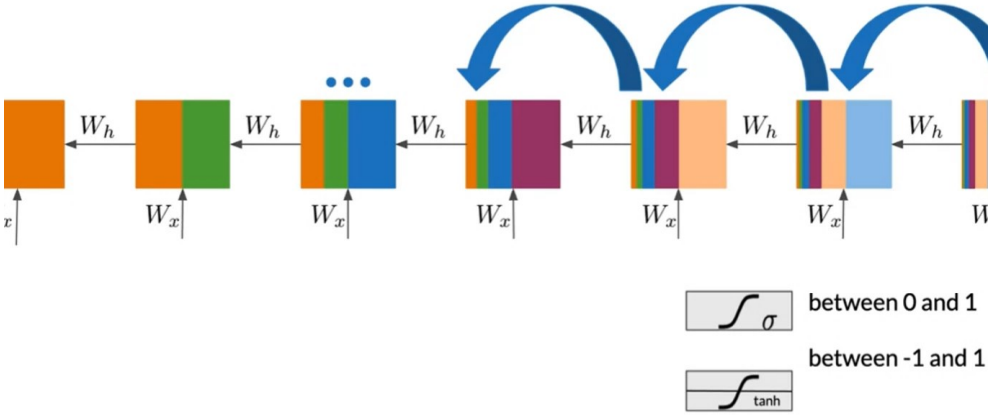
Struggles with longer sequences

Prone to vanishing or exploding gradients

is part of how vanishing/exploding gradients are created,

plearning.ai

ckpropagation through time



You can think of gradients as a measure

plearning.ai

tline

Meet the Long short-term memory unit!
LSTM architecture
Applications

along with a few examples of

ΓMs: a memorable solution

Learns when to remember and when to forget

Basic anatomy:

- A cell state
- A hidden state with three gates
- Loops back again at the end of each time step

Gates allow gradients to flow unchanged

these three gates that track inputs as it arrives.

ving for vanishing or exploding gradients

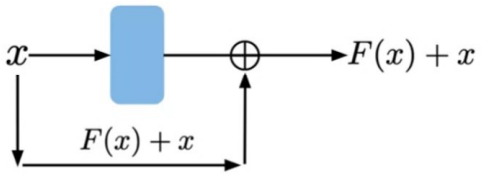
Identity RNN with ReLU activation

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

-1 ⟶ 0

Gradient clipping 32 ⟶ 25

Skip connections



This effectively skips over the activation functions and

ΓMs: a memorable solution

Learns when to remember and when to forget

Basic anatomy:

- A cell state
- A hidden state with three gates
- Loops back again at the end of each time step

training to decide on what changes to make.

ΓMs: Based on previous understanding

state = before conversation
get gate = beginning of conversation

ıt gate = thinking of a response
put gate = responding

ated cell state = after conversation

since you began
your conversation.

plearning.ai

plications of LSTMs

t-character
liction

Chatbots

Music
composition

Image
captioning

Speech
recognition

In overcoming the problems
of a traditional RNN,

plearning.ai

ΓMs: Based on previous understanding

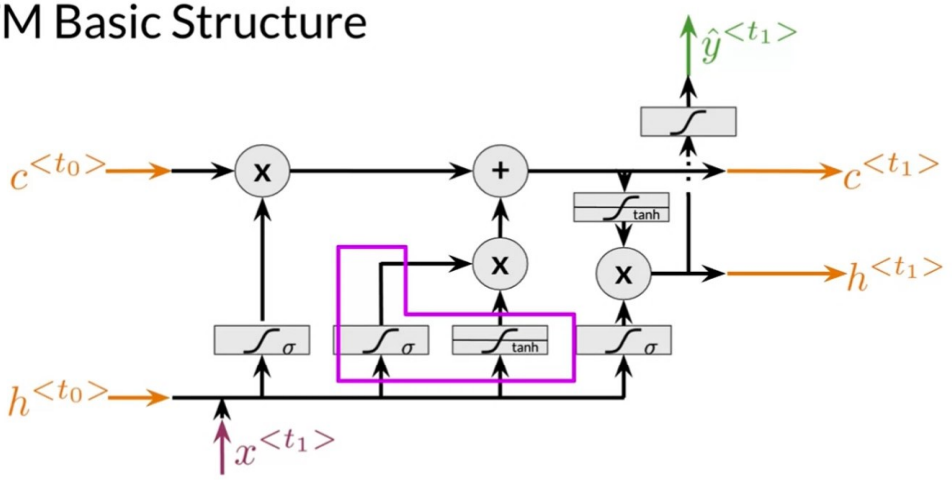
state = before conversation
get gate = beginning of conversation

ıt gate = thinking of a response
put gate = responding

When you decide what to say next,

plearning.ai

ΓM Basic Structure



a sigmoid layer to choose
which values to update,

plearning.ai

mmary

LSTMs use a series of gates to decide which information to keep:

- Forget gate decides what to keep
- Input gate decides what to add
- Output gate decides what the next hidden state will be

One time step is completed after updating the states

you will have completed one timestep.

plearning.ai

pes of Entities

Thailand: Geographical

Google: Organization

Indian: Geopolitical

or geopolitical entities like Indian.

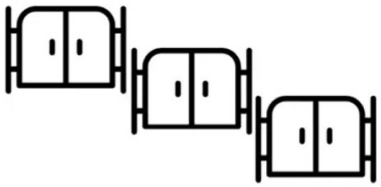
plearning.ai

mmary

LSTMs offer a solution to vanishing gradients

Typical LSTMs have a cell and three gates:

- Forget gate
- Input gate
- Output gate



That typical LSTMs have a cell or memory and three gates,

plearning.ai

hat is Named Entity Recognition?

Locates and extracts predefined entities from text

Places, organizations, names, time and dates

They can even be times and dates.

plearning.ai

ample of a labeled sentence

Sharon flew to Miami last Friday.

B-per B-geo B-tim

You'll notice that here

plearning.ai

Press Esc to exit full screen

Training NER Data Processing

deeplearning.ai

Let's go over how you can apply both to your named entity recognition model.

re Types of Entities

December: Time Indicator

Egyptian statue: Artifact

Barack Obama: Person

or a person such as Barack Obama.

plearning.ai

plications of NER systems

- Search engine efficiency
- Recommendation engines
- Customer service
- Automatic trading

building a web scraper to get

plearning.ai

ocessing data for NERs

Assign each class a number

Assign each word a number

Sharon flew to Miami last Friday.

[4282, 853, 187, 5388, 2894, 7]

B-per O O B-geo O B-tim

853 corresponds to flew,
187 corresponds with to, and so forth.

ining the NER

Create a tensor for each input and its corresponding number

Put them in a batch ⟶ 64, 128, 256, 512 ...

Feed it into an LSTM unit

Run the output through a dense layer

Predict using a log softmax over K classes

a tensor for each input and its
corresponding label, as you saw before.

tline

Convert words and entity classes into arrays

Token padding

Create a data generator

label data into arrays of numbers
that correspond to one another.

ken padding

LSTMs, all sequences need to be the same size.

Set sequence length to a certain number

Use the <PAD> token to fill empty spaces

You'll be implementing
this in the assignments.

nmary

Convert words and entities into same-length numerical arrays

Train in batches for faster processing

Run the output through a final layer and activation

like numerical array
conversion with token padding.

aluating the model

Pass test set through the model

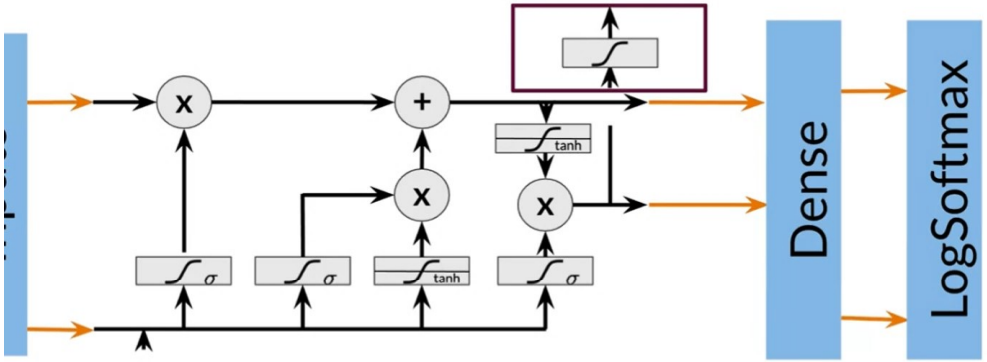
Get arg max across the prediction array

Mask padded tokens

Compare outputs against test labels

compare the outputs against your test
labels to see how accurate your model is.

ining the NER



And then you compute a log softmax
to get the corresponding outputs.

Press Esc to exit full screen

deeplearning.ai

Computing
Accuracy

You're almost ready to use your new model.

nmary

If padding tokens, remember to mask them when computing accurac
Coding assignment!

plearning.ai

at do Siamese Networks learn?

I am happy because I am learning 😃 🙁

Classification: categorize things

Siamese Networks: Identify similarity between things

What is your age?

How old are you?

When you build a Siamese model, you're trying to identify the difference or

plearning.ai

aluating the model in Python

```
def evaluate_model(test_sentences, test_labels, model):
    pred = model(test_sentences)
    outputs = np.argmax(pred, axis=2)
    mask = ...
    accuracy = np.sum(outputs==test_labels)/float(np.sum(mask))

    return accuracy
```

and dividing it by the sum of all your mask tokens.

plearning.ai

estion Duplicates

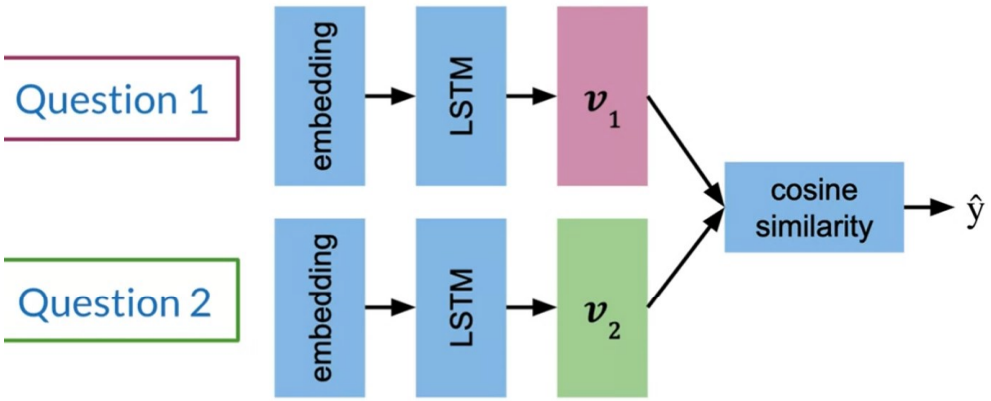
How old are you? = What is your age?

Where are you from? ≠ Where are you going?

This example shows that comparing meaning is not as simple as just comparing words.

plearning.ai

odel Architecture



represents Question 1 and Question 2.

ss Function

$$\cos(v_1, v_2) = \frac{v_1 \cdot}{||v_1||}$$

$$s(v_1, v_2)$$

How old are you?	Anchor
What is your age?	Positive
Where are you from?	Negative

$$s(A, P)$$

where A refers to the anchor question,

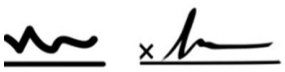
mese Networks in NLP

ndwritten checks

What is your age?
How old are you?

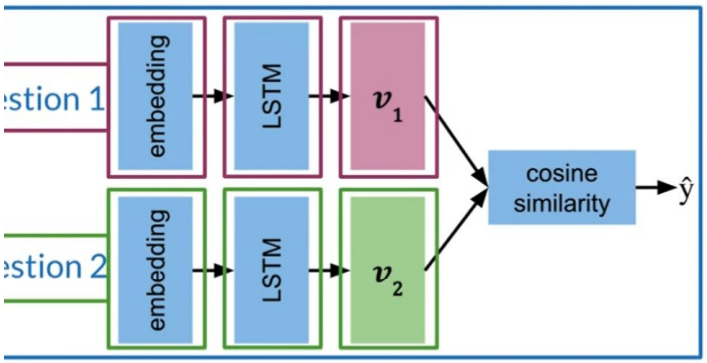
Question duplicates

Queries



to predict whether a new query is similar to the one that was already executed.

odel Architecture



1) Inputs
2) Embedding
3) LSTM
4) Vectors

Then you take the outputs of each of the sub-networks and

plets

How old are you? A
What is your age? P
Where are you from? N

} Triplets !!!

what gives rise to the name triplets,

plearning.ai

plet Selection

Hard triplets are better for training !

Triplet A, P, N: { duplicate set: A, P; non-duplicate set: A, N

Random:

$$\mathcal{L} = \max(diff + \alpha, 0)$$

$$\text{diff} = s(A, N) - s(A, P)$$

Easy to satisfy. Little to learn

Instead of selecting random triplets,

plearning.ai

ss Function

$$\cos(v_1, v_2) = \frac{v_1 \cdot}{||v_1||}$$

$$s(v_1, v_2)$$

How old are you? Anchor

What is your age? Positive $s(A, P) \approx 1$

Where are you from? Negative $s(A, N) \approx -1$

$$\text{diff} = s(A, N) - s(A, P)$$

and P to calculate the difference.

plearning.ai

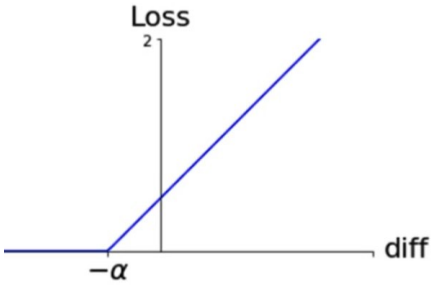
plet Loss

Simple loss:

$$\text{diff} = s(A, N) - s(A, P$$

Non linearity:

$$\mathcal{L} = \begin{cases} 0; & \text{if } diff \leq 0 \\ diff; & \text{if } diff > 0 \end{cases}$$



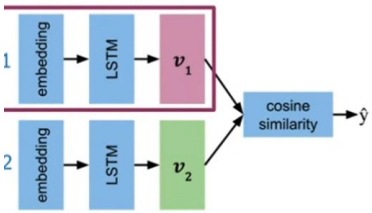
Alpha margin:

$$\mathcal{L} = \begin{cases} 0; & \text{if } diff + \alpha \\ diff + \alpha; & \text{if } diff + \alpha \end{cases}$$

improve and learn from this training example.

plearning.ai

mputing The Cost



Batch 1
What is your age?
Can you see me?
Where are thou?
When is the game?

v_1 = (1, d_mod

v_{1_1}				
v_{1_2}				
v_{1_3}				
v_{1_4}				

the vector outputs for each question in the batch.

plearning.ai

mputing The Cost

$s(v_1, v_2)$

v_1

_1	_2	_3	_4
0.9	-0.8	0.3	-0.5
-0.8	0.5	0.1	-0.2
0.3	0.1	0.7	-0.8
-0.5	-0.2	-0.8	1.0

ranges from negative 1 to positive 1,

4:21

learning.ai

5:24

mputing The Cost

repare the batches as follows:

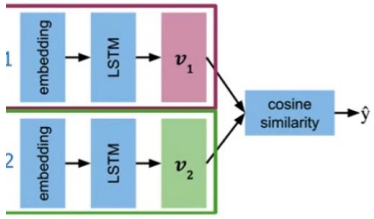
What is your age?	How old are you?
Can you see me?	Are you seeing me?
Where are thou?	Where are you?
When is the game?	What time is the game?

b = 4

none of the rows in those column contain a sentence that

plearning.ai

mputing The Cost



Batch 1
What is your age?
Can you see me?
Where are thou?
When is the game?

Batch 2
How old are you?
Are you seeing me?
Where are you?
What time is the game?

v_1 = (1, d_mod

v_{1_1}				
v_{1_2}				
v_{1_3}				
v_{1_4}				

v_2

v_{2_1}				
v_{2_2}				
v_{2_3}				
v_{2_4}				

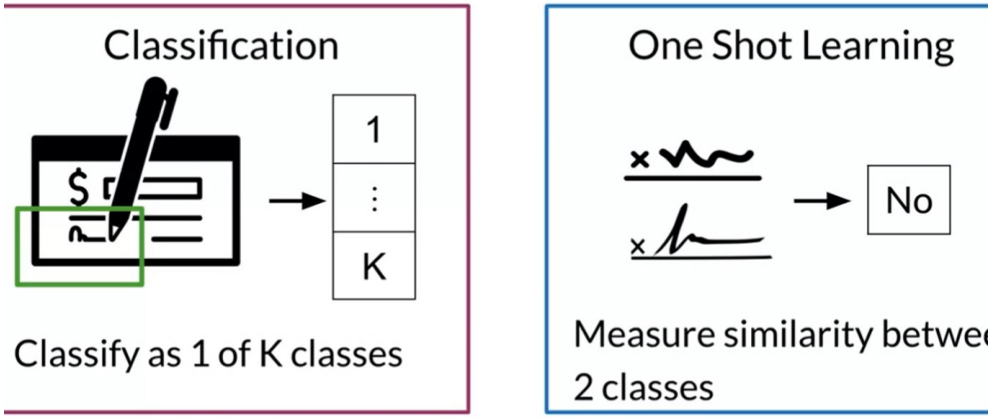
Here, for example, v_1_1 is a duplicate of v_2_1,

3:18

learning.ai

5:24

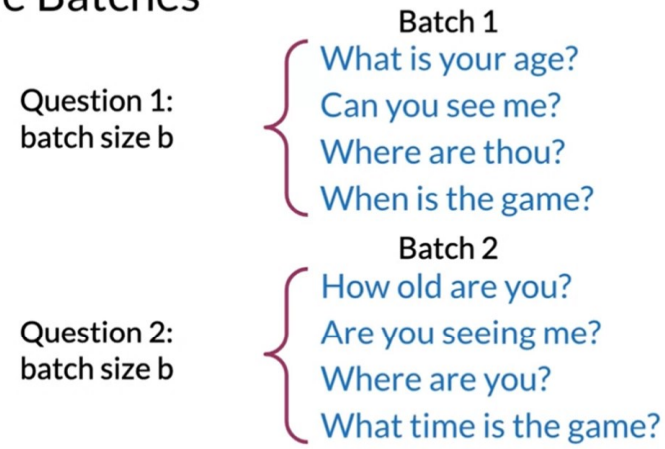
ssification vs One Shot Learning



Then you can test a similarity score against

plearning.ai

epare Batches



The second question in Batch 1 is

plearning.ai

mputing The Cost

$s(v_1, v_2)$

v_1

_1	_2	_3	_4
0.9	-0.8	0.3	-0.5
-0.8	0.5	0.1	-0.2
0.3	0.1	0.7	-0.8
-0.5	-0.2	-0.8	1.0

$$\mathcal{L}(A, P, N) = \max(diff + \alpha, 0)$$

$$\text{diff} = s(A, N) - s(A, P)$$

$$\mathcal{J} = \sum_{i=1}^{m} \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)})$$

but there are more techniques available that's can

plearning.ai

5:24

e Shot Learning

No need for retraining !

Learn a similarity score

a similarity function that can be

plearning.ai

tline

Introduction to Neural Machine Translation
Seq2Seq model and its shortcomings
Solution for the information bottleneck

To get started on this weeks material,
I'll introduce you to neural machine

plearning.ai

2Seq model

How are you today?

Wie geht es Ihnen heute

It takes in a hidden states and a string
of words, such as a single sentence.

epare Batches

Batch 1

Question 1:
batch size b

- What is your age?
- Can you see me?
- Where are thou?
- When is the game?

Batch 2

Question 2:
batch size b

- How old are you?
- Are you seeing me?
- Where are you?
- What time is the game?

v_1 = (1, d_mod

v_{1_1}
v_{1_2}
v_{1_3}
v_{1_4}

v_2

v_{2_1}
v_{2_2}
v_{2_3}
v_{2_4}

to get outputs vectors
for each batch.

plearning.ai

q2Seq model

Introduced by Google in 2014

Maps variable-length sequences to fixed-length memory

LSTMs and GRUs are typically used to overcome the vanishing gradient problem

GRUs are typically used to
avoid vanishing gradients.

plearning.ai

q2Seq shortcomings

Variable-length sentences + fixed-length memory =

As sequence size increases, model performance decreases

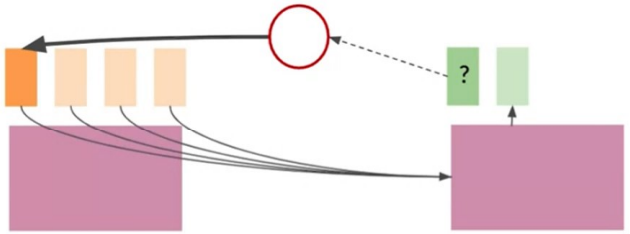
This results in lower model performance as sequence size increases,

plearning.ai

ution: focus attention in the right place

Prevent sequence overload by giving the model a way to focus on the **likeliest** words at each step

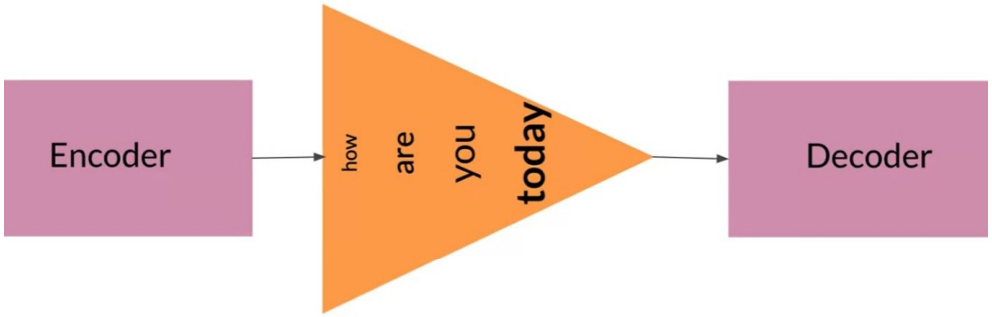
Do this by providing the information specific to each input word



Up next you'll get a conceptual idea of what this new layer is doing and why.

plearning.ai

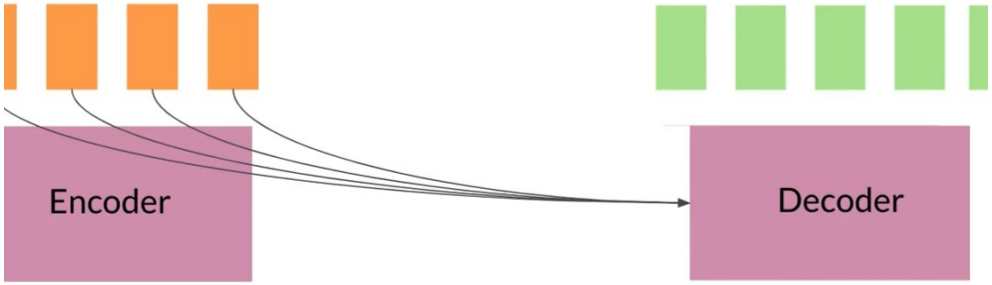
e information bottleneck



are given more importance.

plearning.ai

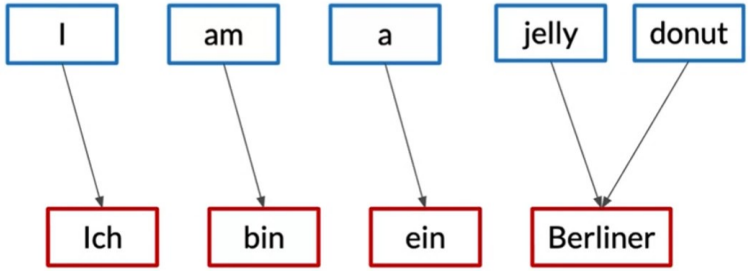
e vector per word



memory efficient model that predicts accurately from a long sequence?

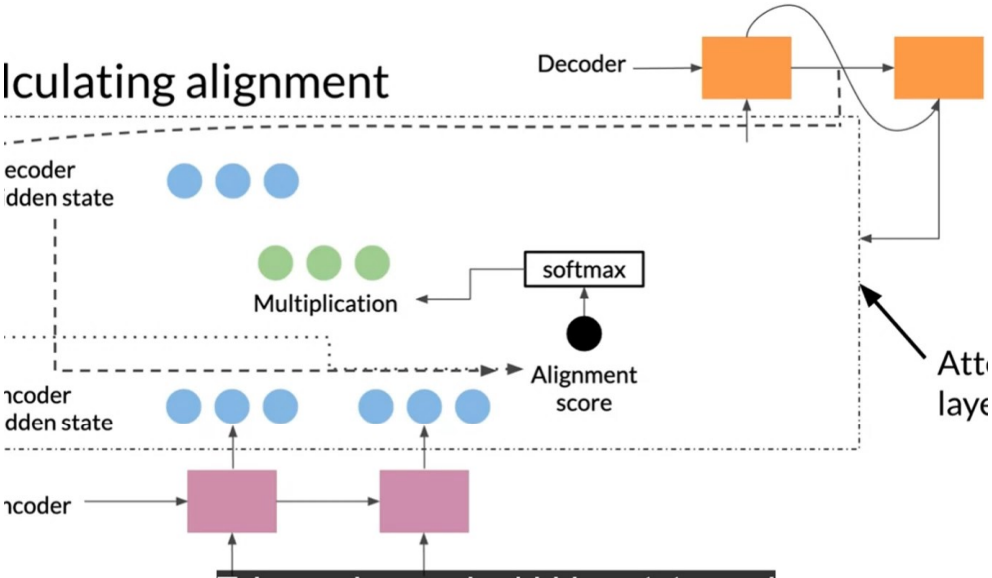
plearning.ai

ord alignment



an optional reading about this historic misunderstanding In the course material.

lculating alignment



Take each encoder hidden state, and multiply it by its softmax score,

otivation for alignment

rectly aligned words are the goal:

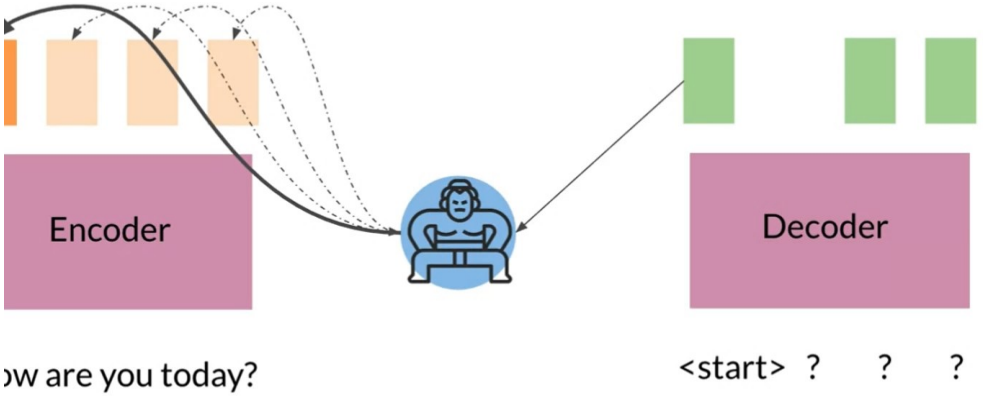
Translating from one language to another

Word sense discovery and disambiguation

Achieve alignment with a system for retrieving information step by step and scoring it

For this week's assignments on neural machine translation with attention,

ve some inputs more weight!



You can see here that the hidden state shown in dark orange has a heavier line,

ta in machine translation

English	German
I am hungry!	Ich habe Hunger.
...	...
I watched the soccer game.	Ich habe das Fußballspiel gesehen.

You're going to have a great many of these inputs.

plearning.ai

ıchine translation setup

e-of-the-art uses pre-trained vectors

erwise, represent words with a one-hot vector to create the input

p track of index mappings with word2ind and ind2word dictionaries

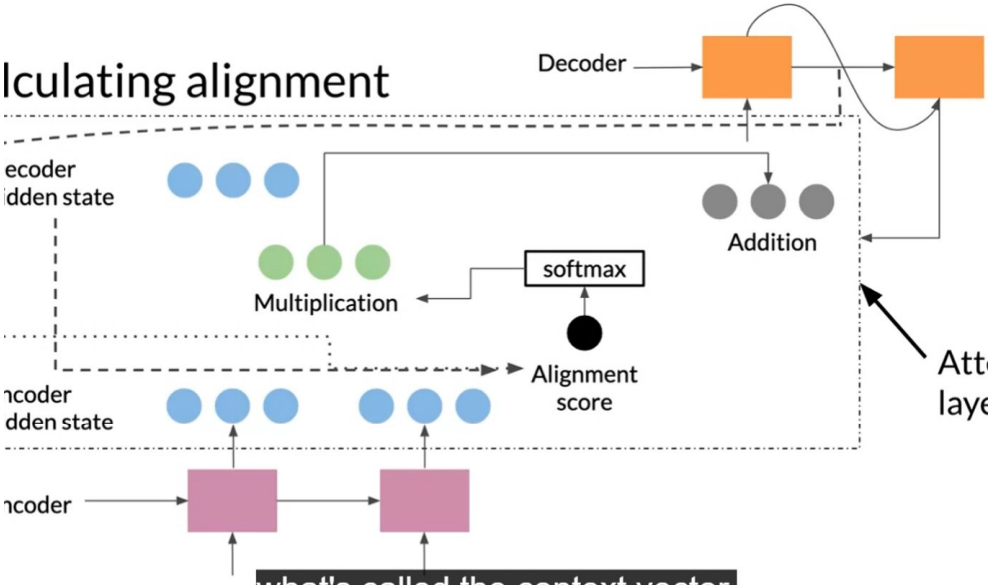
start-of and end-of sequence tokens:

<SOS> <EOS>

a starts of sequence token like this,

plearning.ai

lculating alignment



what's called the context vector.

plearning.ai

ıchine translation setup

e-of-the-art uses pre-trained vectors

erwise, represent words with a one-hot vector to create the input

p track of index mappings with word2ind and ind2word dictionaries

with a word to index and index toward dictionary.

plearning.ai

eparing to Translate to German

GLISH SENTENCE:

n the ballpoint and the mechanical pencil in the series are equipped
a special mechanism: when the twist mechanism is activated, the
is pushed forward.

KENIZED VERSION OF THE ENGLISH SENTENCE:

```
46   4   11358  362   8   4   23326   20104  1745  8210  9641   5   6
103  31  2767   30    13  914 4797    64   196   4   22474   5  4797  16
64   86   2    4  1060  16  6413 1138   3   1   0   0   0   0   0   0   0
 0   0    0    0   0    0    0    0    0    0    0   0   0   0   0   0   0]
```

Then a series of zeros
that's if you were to count,

tline

Teacher forcing
Model for NMT with attention

along with being introduced to

eparing to Translate to German

GLISH SENTENCE:

n the ballpoint and the mechanical pencil in the series are equipped
a special mechanism: when the twist mechanism is activated, the
is pushed forward.

KENIZED VERSION OF THE ENGLISH SENTENCE:

```
46   4   11358  362   8   4   23326   20104  1745  8210  9641   5   6
103  31  2767   30    13  914 4797    64   196   4   22474   5  4797  16
64   86   2    4  1060  16  6413 1138   3   1   0   0   0   0   0   0   0
 0   0    0    0   0    0    0    0    0    0    0   0   0   0   0   0   0]
```

the tokenized version of
the English sentence,

glish to German

MAN TRANSLATION:

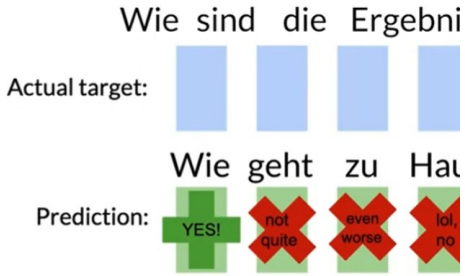
Kugelschreiber und der Drehbleistift der Serie sind mit einem
onderen Mechanismus ausgestattet: Bei Betätigung der Drehmecha
die Schreibmine nach vorne geschoben.

KENIZED VERSION OF THE GERMAN TRANSLATION:

```
9  3892  5280 14774 2418   12   11  9883  6959  7298 15157   5   11 845
114 5324 10565  2520   64  752 12954 26538   147   11  9883 23326 201
 78   10 21150 10166  126 14566   5  23850  1171    3   1   0   0   0   0
 0   0    0    0    0    0    0    0    0    0    0   0   0   0   0]
```

of unpaired tokens
in the sequence.

acher forcing: motivation



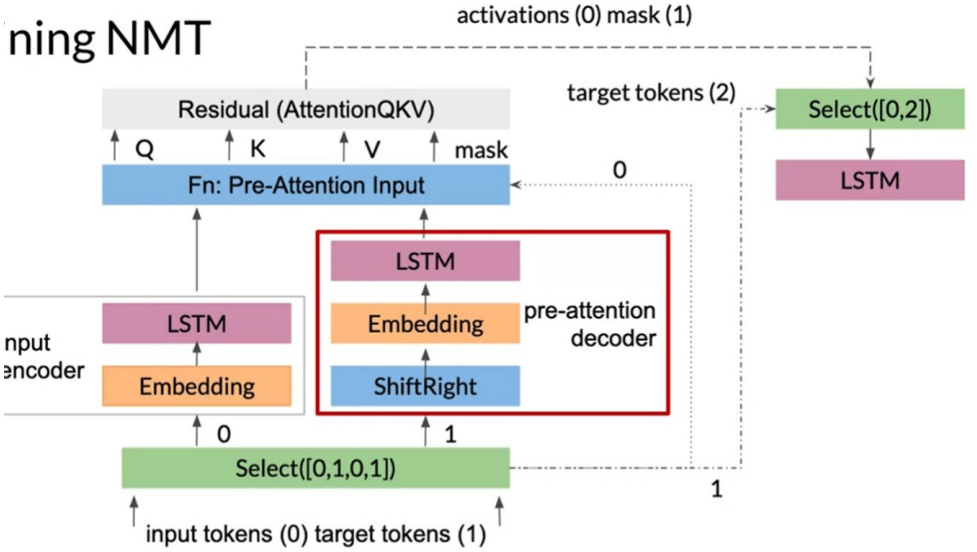
low are the results?

Encoder

Decoder

the prediction made at each step.

ning NMT



which is what the second Select is doing.

w to know predictions are correct?

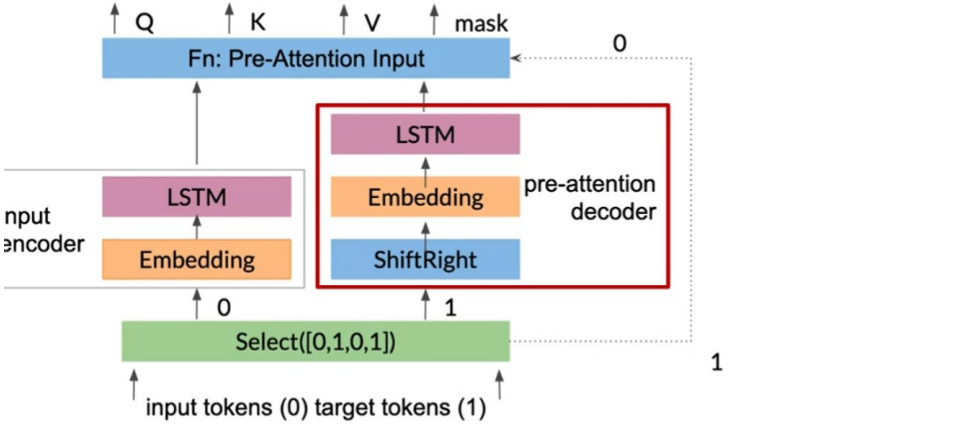
her forcing allows the model to "check its k" at each step

compare its prediction against the real ut during training

lt: Faster, more accurate training

This yields faster training with

ning NMT



you can prepare them for the attention layer.

EU Score

ids for Bilingual Evaluation Understudy

uates the quality of machine-translated text by comparing "candidat
to one or more "reference" translations.

es: the closer to 1, the better, and vice versa:

The closer to zero,
the worse it is.

plearning.ai

EU score is great, but...

sider the following:

BLEU doesn't consider semantic meaning

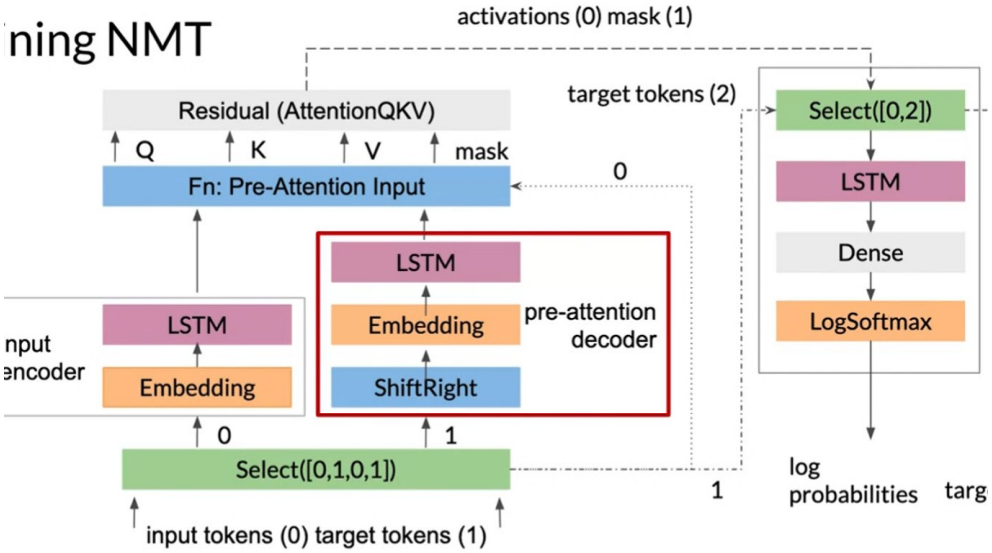
BLEU doesn't consider sentence structure:

"Ate I was hungry because!"

It also doesn't consider the
structure of the sentence.

plearning.ai

ning NMT



Take a break and just
let all that sink in.

plearning.ai

EU Score

Candidate	I	I	am	I	I
Reference 1	Younes	said	I	am	hungry
Reference 2	He	said	I	am	hungry

"I" appears at most once in both, so clip to one: $m_w = 1$

$$\frac{\text{(Sum over unique n-gram counts in the candidate)}}{\text{(total \# of words in candidate)}}$$

the unique uni-gram
counts in the candidates.

plearning.ai

)UGE evaluation

lodel	The	cat	had	striped	orange	fur
eference	The	cat	had	orange	fur	

ecall = How much of the reference text is the system text capturing?

ecision = How much of the model text was relevant?

which you can think of as answering this question.

ecision in ROUGE

lodel	The	cat	had	striped	orange	fur
eference	The	cat	had	orange	fur	

um of overlapping unigrams in model and reference)

(total # of words in model)

$\frac{5}{6}$ Precis = 0.83

or roughly 83 percent of the words were relevant.

)UGE

all-Oriented Understudy for Gisting Evaluation

uates quality of machine text

asures precision and recall between generated text and an-created text

and then bi-gram can be the two words a pie.

call in ROUGE

lodel	The	cat	had	striped	orange	fur
eference	The	cat	had	orange	fur	

(Sum of overlapping unigrams in model and reference)

(total # of words in reference)

$\frac{5}{5}$

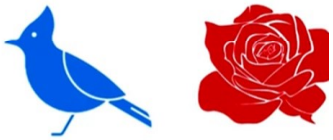
the cat had many orange fur and the reference texts,

nmary

BLEU score compares “candidate” against “references” using an n-gr
average

BLEU doesn’t consider meaning or structure

ROUGE measures machine-generated text against an “ideal” referer



which was created to evaluate

plearning.ai

eedy decoding

cts the most probable word at each step

the best word at each step may not be the best for longer sequences

Ich habe Hunger.

I am _hungry_ .
I am, am, am, am...

fine but if you have many other words to consider,

plearning.ai

oblems in ROUGE

Doesn’t take themes or concepts into consideration (i.e., a low ROU
score doesn’t necessarily mean the translation is bad)

Model	I	am	a	fruit-filled	pastry
Reference	I	am	a	jelly	donut

you start taking your own ROUGE scores.

plearning.ai

tline

Random sampling
Temperature in sampling
Greedy decoding
Beam search
Minimum Bayes’ risk (MBR)

of this week's lectures. That's awesome.

plearning.ai

mperature

ampling, temperature is a parameter allowing for more or less
domness in predictions

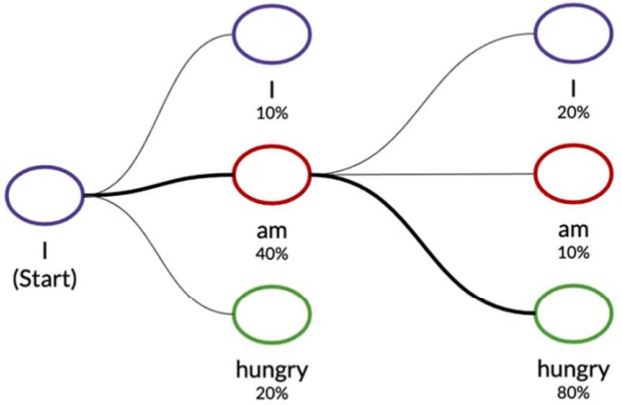
er temperature setting = More confident, conservative network

ner temperature setting = More excited, random network (and more
akes)

and you may get some
pretty fun predictions.

plearning.ai

am search example



B = 3

This is a very simple
example but you can

5:38

plearning.ai

ndom sampling

am	full	hungry	I	the
0.05	0.3	0.15	0.25	0.25

en a little too random for accurate translation!

ition: Assign more weight to more probable words, and less weight t
probable words.

and less weight to the others.

plearning.ai

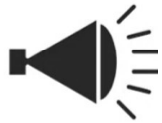
am search decoding

oader, more exploratory decoding alternative

cts multiple options for the best input based on conditional
bability

nber of options depends on a predetermined beam width parameter

cts B number of best alternatives at each time step



to select several
possible options.

4:31

plearning.ai

oblems with beam search

e the model learns a distribution, that tends to carry more weight tha
e tokens

cause translation problems, i.e. in a speech corpus that hasn't been
ned

It can also use single
tokens in a problematic way,

oblems with beam search

mag die Vereinigten Staaten, weil die Vereinigten Staaten groß sind."

with 11 good English translations of "Vereinigten Staaten," but a ~1
ability of the non-word "Uhm" occurring, you might get this as a
slation:

e the United States, because the Uhm is big. "

n with 11^2 good translations, the most probable one will still be "Uh

and said because one over
11 squared is less than

oblems with beam search

e the model learns a distribution, that tends to carry more weight tha
e tokens

However, beam search decoding
runs into issues where

oblems with beam search

mag die Vereinigten Staaten, weil die Vereinigten Staaten groß sind."

with 11 good English translations of "Vereinigten Staaten," but a ~1
ability of the non-word "Uhm" occurring, you might get this as a
slation:

e the United States, because the Uhm is big. "

These could be USA, US,

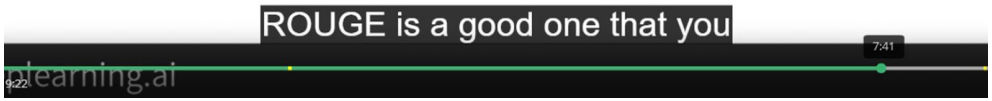
nimum Bayes Risk (MBR)

npares many samples against one another. To implement MBR:

Generate several random samples

Compare each sample against all the others and assign a similarity score (such as ROUGE!)

Select the sample with the highest similarity: the golden one ✨



ample: MBR Sampling

enerate the scores for 4 samples:

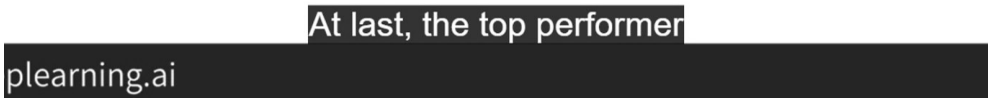
Calculate similarity score between sample 1 and sample 2

Calculate similarity score between sample 1 and sample 3

Calculate similarity score between sample 1 and sample 4

Average the score of the first 3 steps (Usually a weighted average)

Repeat until all samples have overall scores



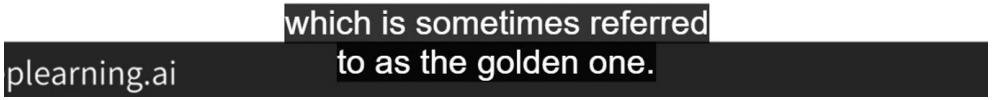
nimum Bayes Risk (MBR)

npares many samples against one another. To implement MBR:

Generate several random samples

Compare each sample against all the others and assign a similarity score (such as ROUGE!)

Select the sample with the highest similarity: the golden one ✨



ample: MBR Sampling

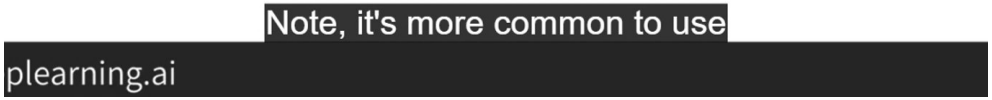
enerate the scores for 4 samples:

Calculate similarity score between sample 1 and sample 2

Calculate similarity score between sample 1 and sample 3

Calculate similarity score between sample 1 and sample 4

Average the score of the first 3 steps (Usually a weighted average)



tline

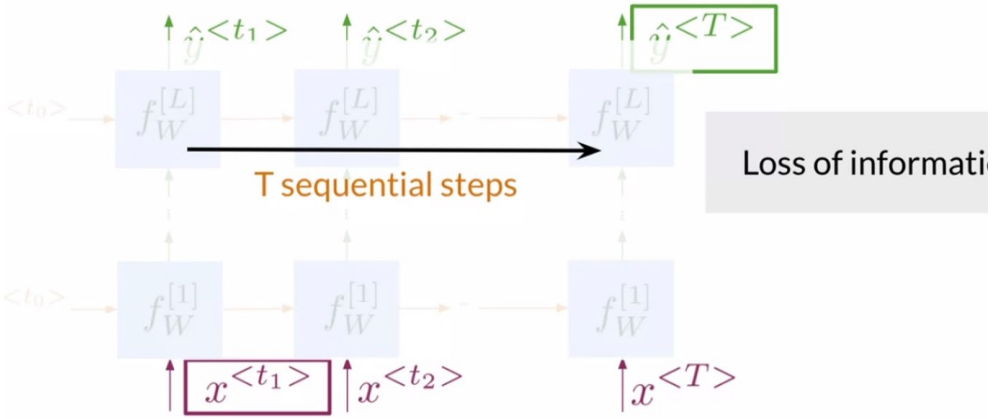
Issues with RNNs

Comparison with Transformers

recurrent neural networks using

plearning.ai

q2Seq Architectures



arise related to the length of your input sequences.

plearning.ai

mmary

Beam search uses conditional probabilities and the beam width parameter

MBR (Minimum Bayes Risk)takes several samples and compares them against each other to find the golden one ✨

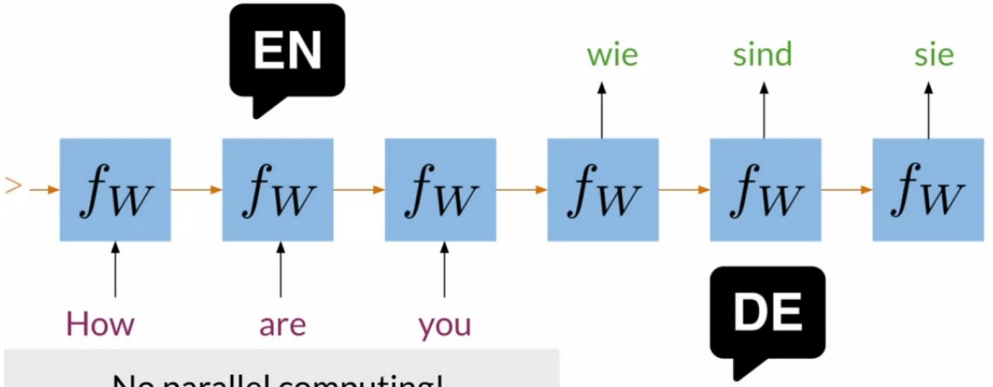
Go forth to the coding assignment!

then chooses the best performer.

plearning.ai

9:22

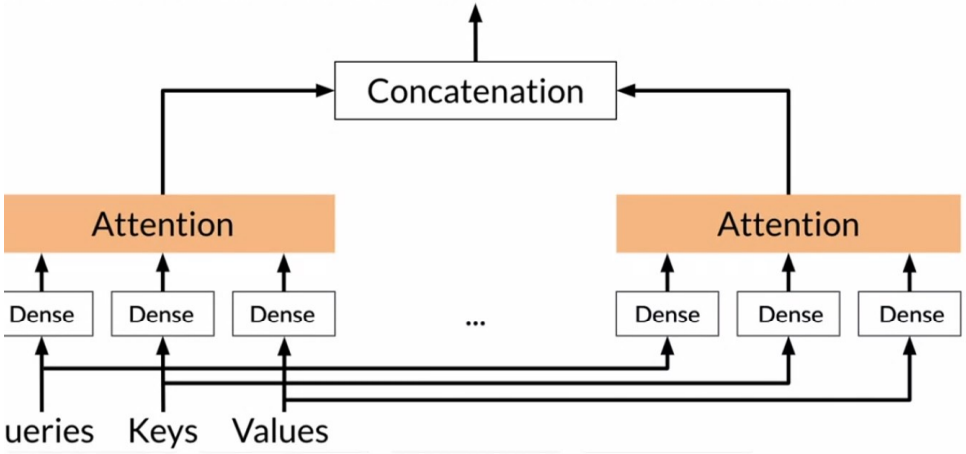
ural Machine Translation



In sequential matcher in order to start the translation,

plearning.ai

NNs vs Transformer: Multi-headed attention



These heads are further

plearning.ai

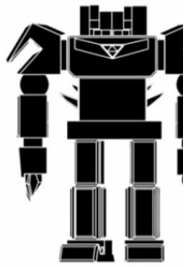
mmary

In RNNs parallel computing is difficult to implement

For long sequences in RNNs there is loss of information

In RNNs there is the problem of vanishing gradient

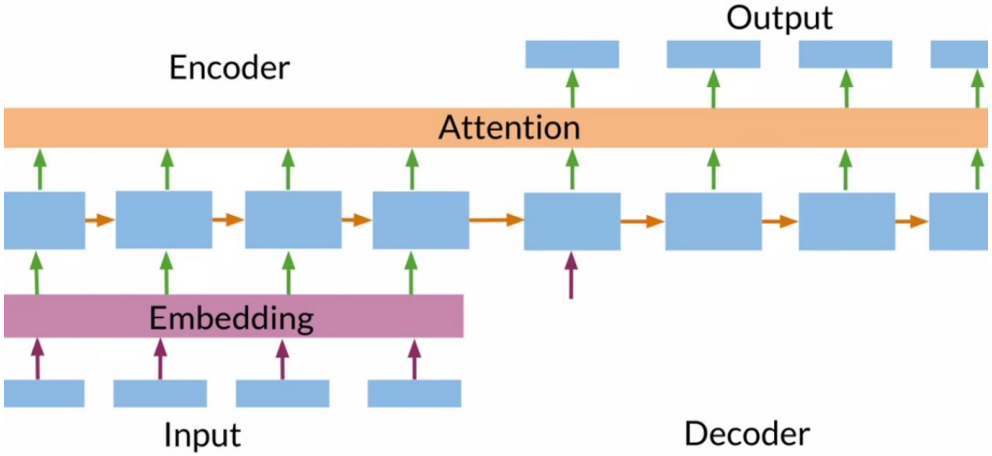
Transformers help with all of the above



With RNNs, it is hard to fully

plearning.ai

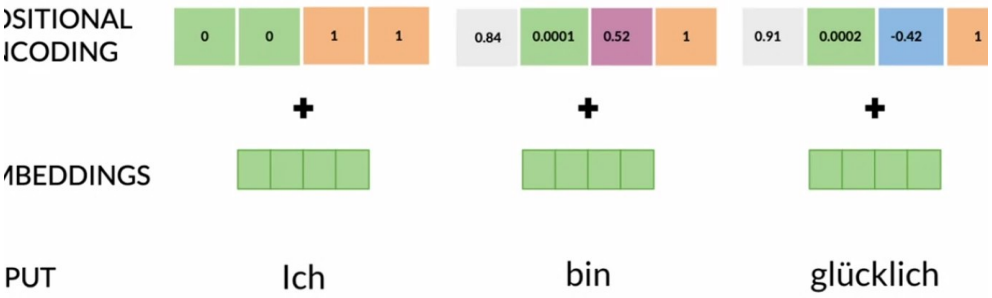
NNs vs Transformer: Encoder-Decoder



For RNNs, the number of steps is equal to T. Finally,

plearning.ai

NNs vs Transformer: Positional Encoding



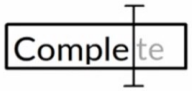
the multi-head attention layer computes

plearning.ai

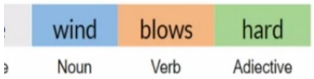
ransformer NLP applications

Text summarization

Translation



Auto-Complete



Named entity recognition (NER)

Question answering (Q&A)

automatic question answering, machine translation.

tate of the Art Transformers

lford, A., et al. (2018)
Open AI

GPT-2: Generative Pre-training for Transformer

vlin, J., et al. (2018)
oogle AI Language

BERT:Bidirectional Encoder Representations from Transformers

olin, R., et al. (2019)
Google

T5: Text-to-text transfer transformer

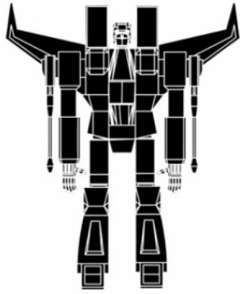
was also created by Google.

Outline

Transformers applications in NLP

Some Transformers

Introduction to T5

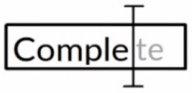


First, I'll mention the most popular applications of Transformers in NLP.

ransformer NLP applications

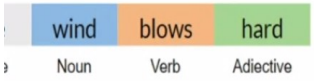
Text summarization

Translation



Auto-Complete

Chat-bots



Named entity recognition (NER)

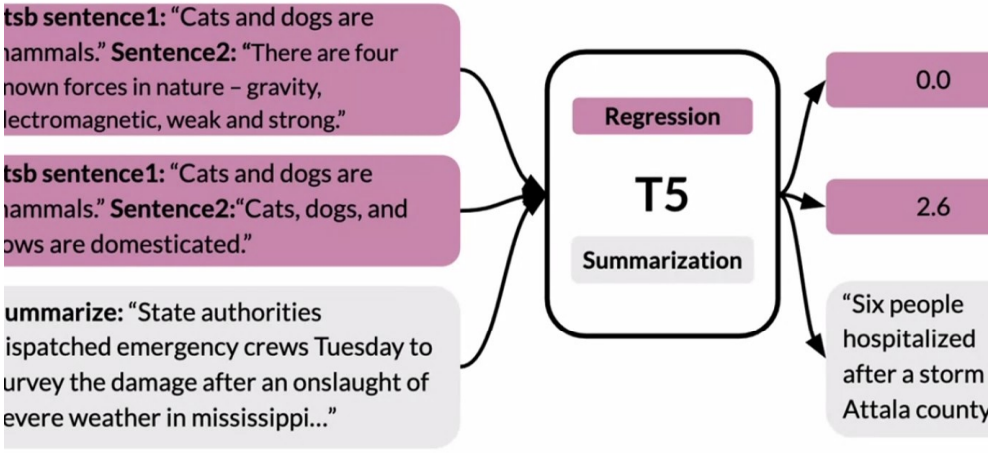
Other NLP tasks

Sentiment Analysi
Market Intelligenc
Text Classification
Character Recogn
Spell Checking

Question answering (Q&A)

Another application is Chat-bots, and many other NLP tasks like Sentiment Analysis,

5: Text-To-Text Transfer Transformer



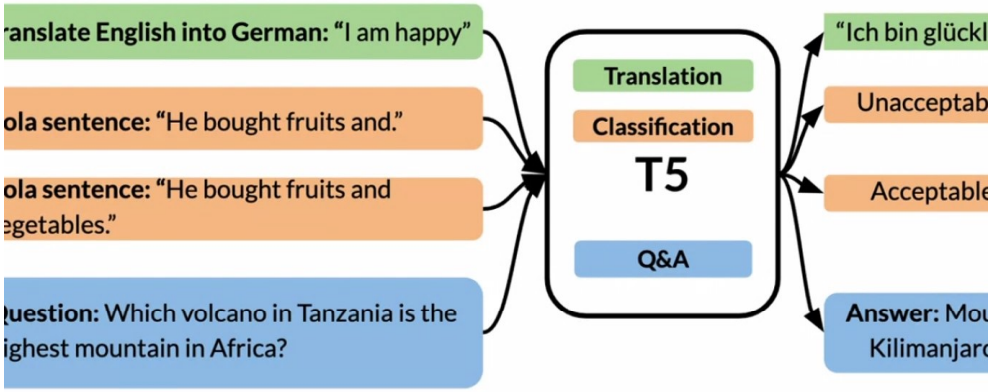
six people hospitalized after
a storm in Attala County.

Outline

- Introducing attention (Translation example)
- Mathematics behind Attention

First I'll introduce the concept
of attention using an example of

5: Text-To-Text Transfer Transformer



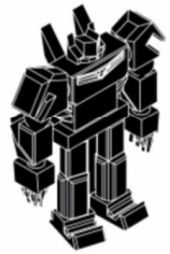
And remember, that's all of these tasks
are done by the same model with no

Summary

Transformers are suitable for a wide range of NLP applications

GPT-2, BERT and T5 are the cutting-edge Transformers

T5 is a powerful multi-task transformer



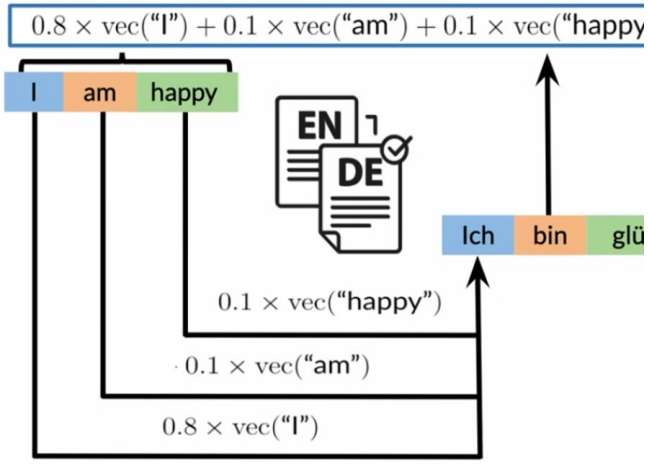
In this video you saw what
are the Transformers applications in NLP,

ntroducing attention - Translation example

query (German ord) looks for similar ys (English words).

ch key has a obability of being a atch for the query.

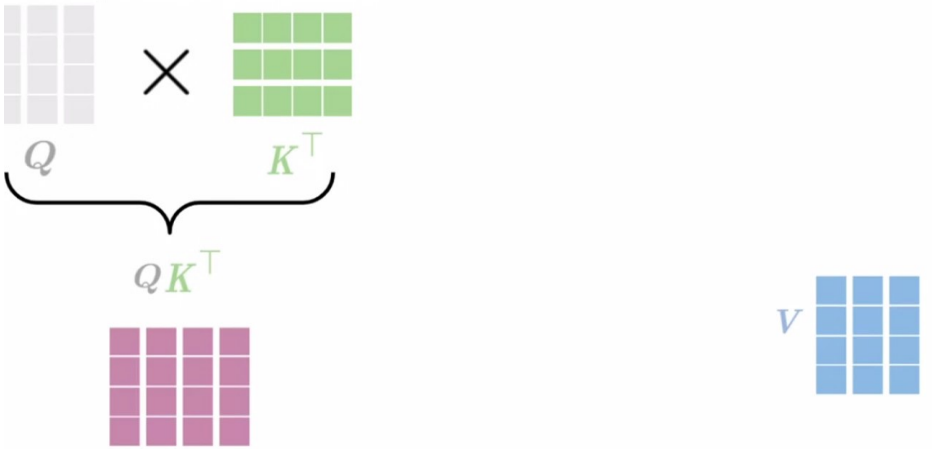
eturn sum of the ys weighted by their obabilities.



In the end, the query gets the sum of the key vectors waited by the probabilities.

plearning.ai

oncept of attention



attention model calculates weights or scores representing

2:55

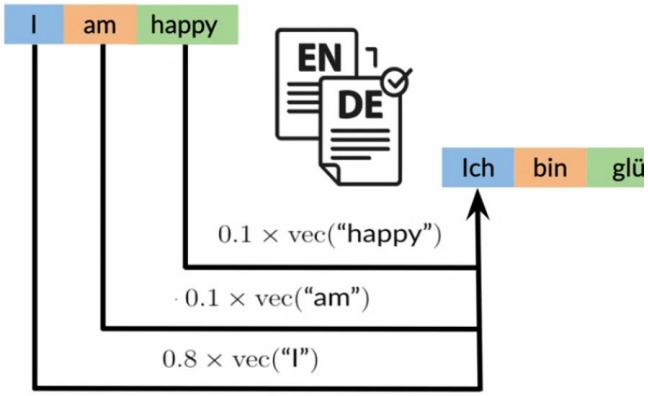
plearning.ai

ntroducing attention - Translation example

query (German ord) looks for similar ys (English words).

ch key has a obability of being a atch for the query.

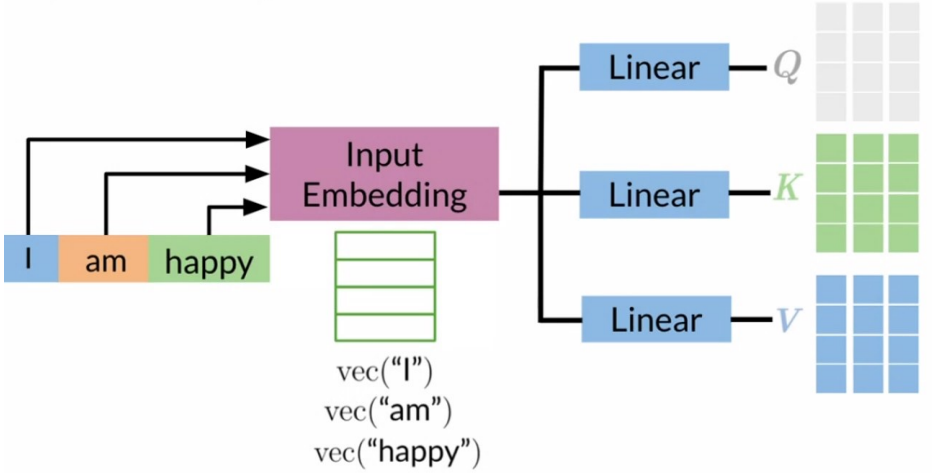
eturn sum of the ys weighted by their obabilities.



get zero points its probability, while it's not very similar to am or

plearning.ai

Queries, Keys and Values

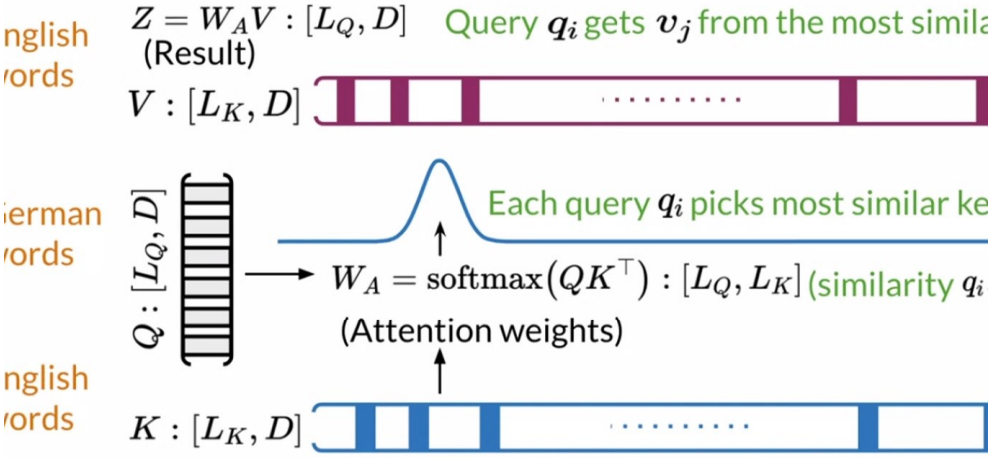


Then you can do the same for the word am to output a second vector.

2:52

plearning.ai

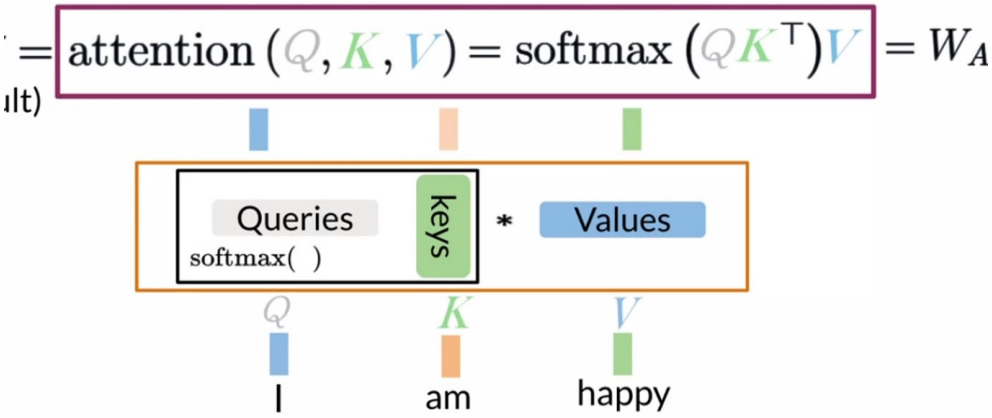
Attention math



which is another matrix of the same shape as keys.

plearning.ai

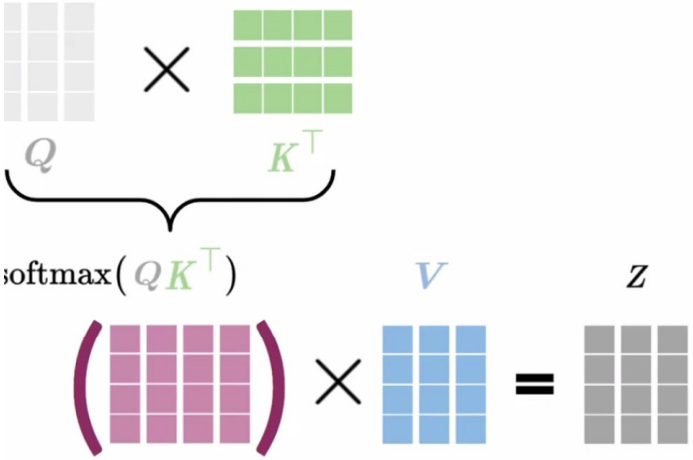
Attention formula



Q times K transpose and then by V and once after softmax operation.

plearning.ai

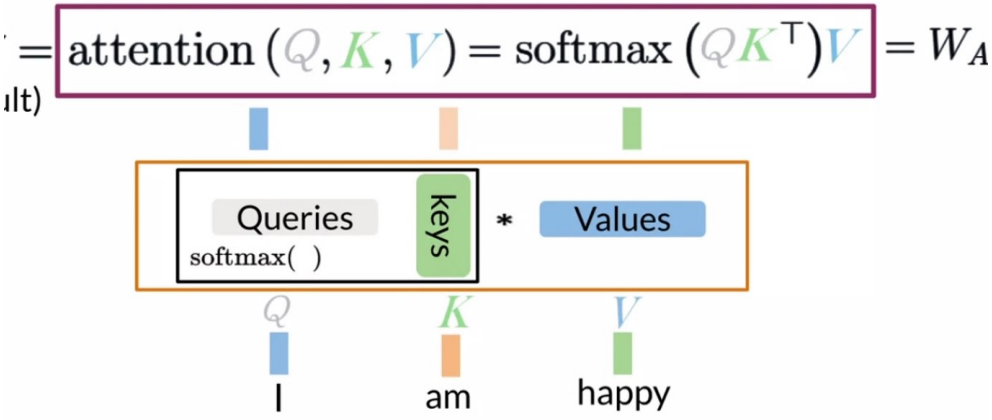
oncept of attention



you will then get a weighted sequence, which is the attention results itself.

plearning.ai

Attention formula



which is the softmax of the dot product between QK transpose times V.

plearning.ai

tline

- Ways of Attention
- Overview of Causal Attention
- Math behind causal attention

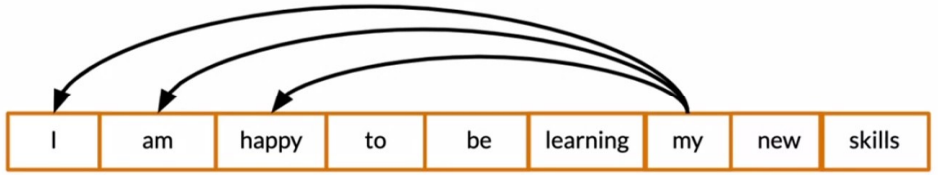
First, you'll see what are three main ways of attention.

plearning.ai

usal attention

Queries and keys are words from the same sentence

Queries should only be allowed to look at words before



these were not generated yet, they can attend to any word in the past though.

plearning.ai

Summary

Dot-product Attention is essential for Transformer

The input to Attention are queries, keys, and values

A softmax function makes attention more focused on best keys

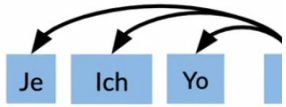
GPUs and TPUs is advisable for matrix multiplications

Dot-product Attention is the heart and soul of transformers.

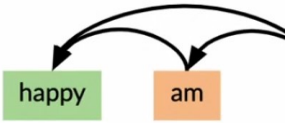
plearning.ai

ree ways of attention

Encoder/decoder attention: One sentence (decoder) looks at another one (encoder)



Causal (self) attention: In one sentence, words look at previous words (used for generation)

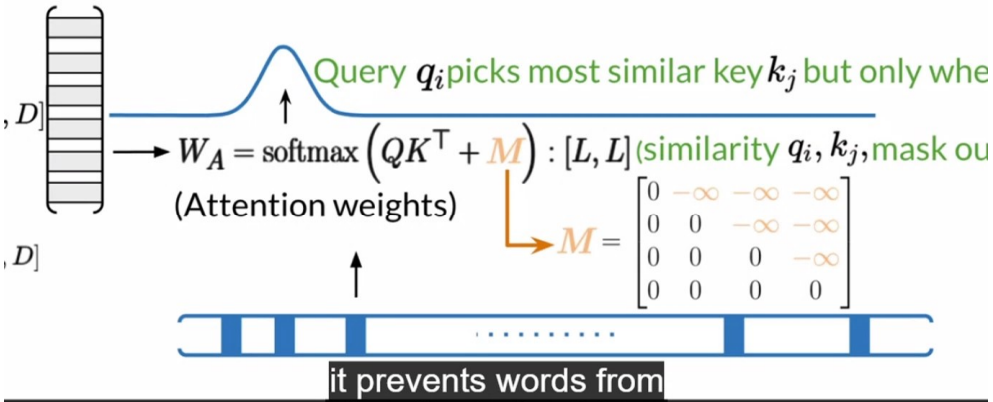


Bi-directional self attention: In one sentence, words look at both previous and future words

where words in the same sentence look both at previous and future words.

plearning.ai

usal attention math



it prevents words from attending to the future.

plearning.ai

mmary

There are three main ways of Attention: Encoder/Decoder, Causa

Bi-directional type

In causal attention, queries and keys come from the same sentenc

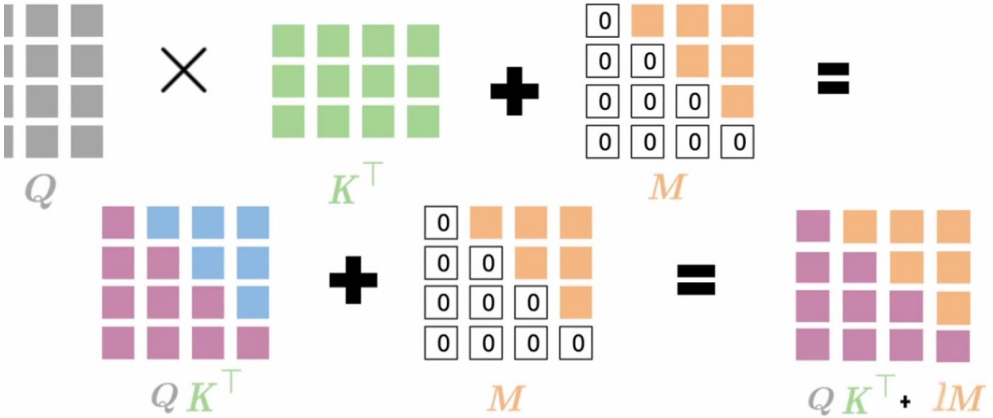
queries search among words before only

Encoder decoder attention, causal self-attention, and

plearning.ai

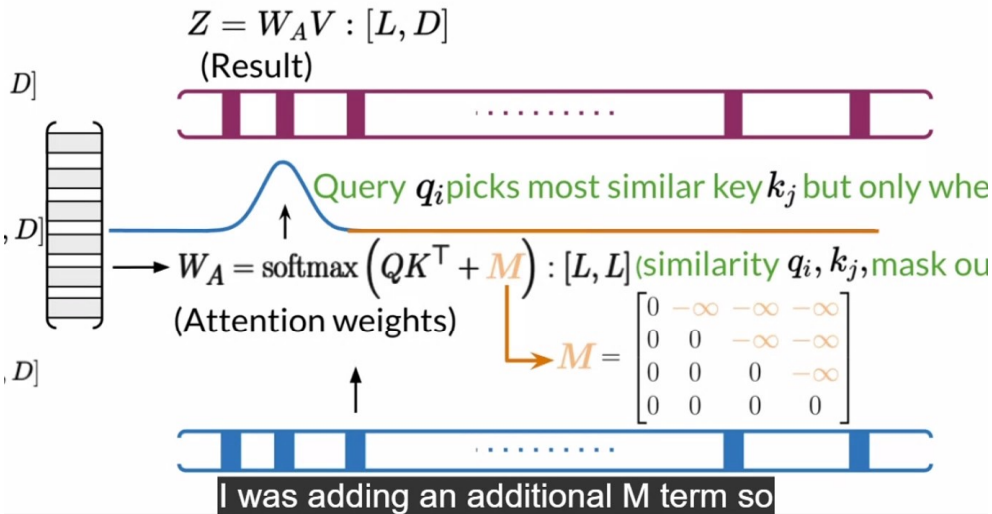
usal attention math

Minus infinity -in practice, a huge negative number



plearning.ai

usal attention math

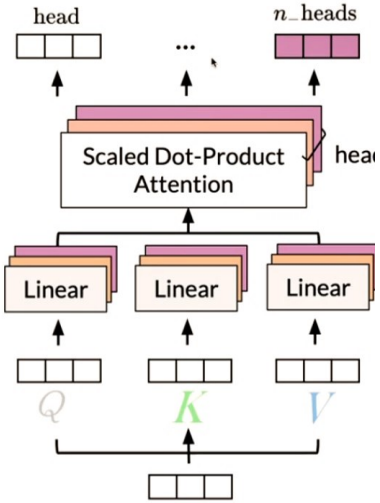


I was adding an additional M term so

plearning.ai

lti-Head Attention

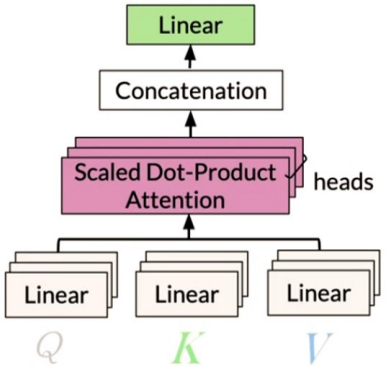
- Each head uses different linear transformations to represent words
- Different heads can learn different relationships between words



So over here you have
N different heads,

plearning.ai

lti-Head Attention - Overview



except by the scale factor,

2:06

learning.ai

tline

Intuition of Multi-Head Attention

Scaled dot-product and concatenation

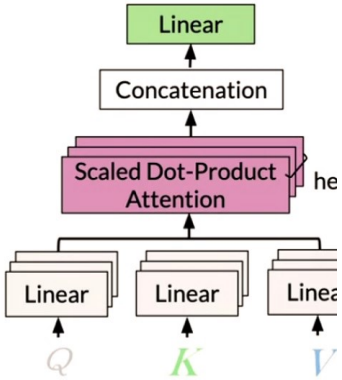
Multi-Head Attention formula

some intuition on
multi-head attention.

plearning.ai

lti-Head Attention - Scaled dot product

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$



DK is the key inquiry dimension.

2:11

learning.ai

ılti-Head Attention math

To understand the math
behind multi-head attention,

ılti-Head Attention math

bedding

W_i^Q W_i^K W_i^V Q_i K_i V_i Z_i W^0

W_h^Q W_h^K W_h^V Q_h K_h V_h Z_h

matrix capital W, superscript 0,

ılti-Head Attention - Scaled dot product

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Query and key dimension

Linear
Concatenation
Scaled Dot-Product Attention
he
Linear Linear Linea
Q K V

from the function to be

ılti-Head Attention - Concatenation

Input(Q, K, V): `[batch, length, d_model]`

Linear layer: `[batch, length, n_heads * d_head]`

Transpose: `[batch, n_heads, length, d_head]`

Apply attention treating `n_heads` like batch

Result shape: `[batch, n_heads, length, d_head]`

Transpose: `[batch, length, n_heads * d_head]`

Linear layer into: `[batch size, length, d_model]`

Linear
Concatenation
Scaled Dot-Product Attention
he
Linear Linear Linea
Q K V

different heads into a
joint representation.

mmary

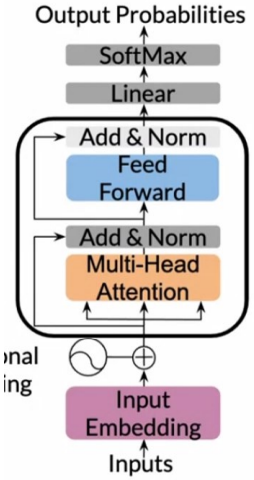
Different heads can learn different relationship between words

Scaled dot-product is adequate for Multi-Head Attention

Multi-Headed models attend to information from different representations at different positions

and values more than once,

ansformer decoder



Overview

- input: sentence or paragraph
  - we predict the next word
- sentence gets embedded, add positional encodi
  - (vectors representing $\{0, 1, 2, \ldots, K\}$)
- multi-head attention looks at previous words
- feed-forward layer with ReLU
  - that's where most parameters are!
- residual connection with layer normalization
- repeat N times

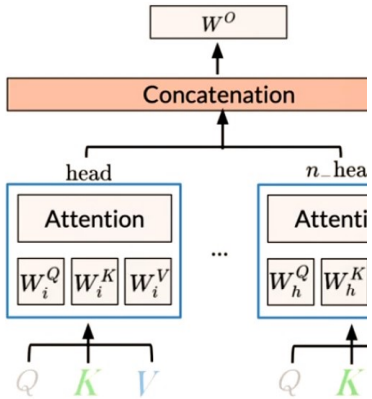
The attention and feed-forward layers are repeated N times.

ılti-Head Attention Formula

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \ldots, h_h)W^0$$

where $h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

Each head h_i is the attention function of **Query, Key and Value** with trainable parameters (W_i^Q, W_i^K, W_i^V)



of queries, keys, and values.

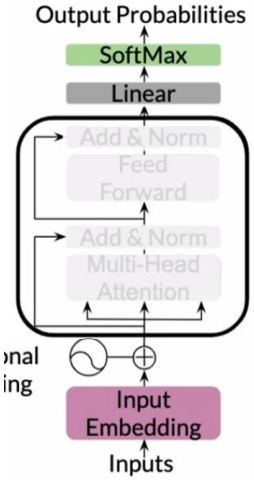
tline

Overview of Transformer decoder

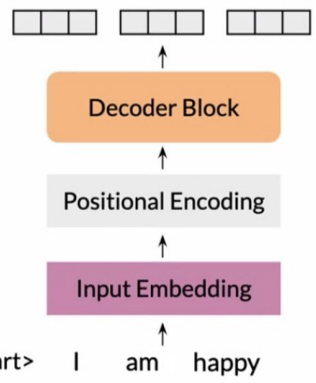
Implementation (decoder and feed-forward block)

>> In this video, you'll see the basic structure of a transformer decoder.

ansformer decoder



Explanation



by vocab size.

mmary

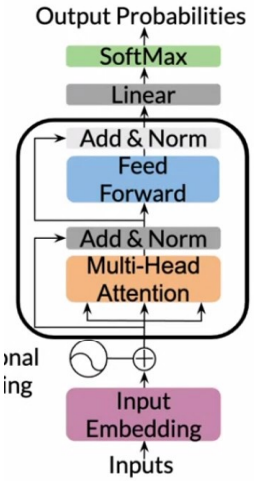
Transformer decoder mainly consists of three layers

Decoder and feed-forward blocks are the core of this model code

It also includes a module to calculate the cross-entropy loss

You saw the building blocks used to implement a transformer decoder.

ansformer decoder

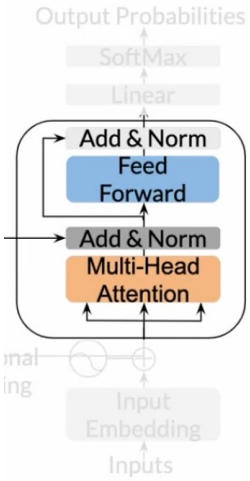


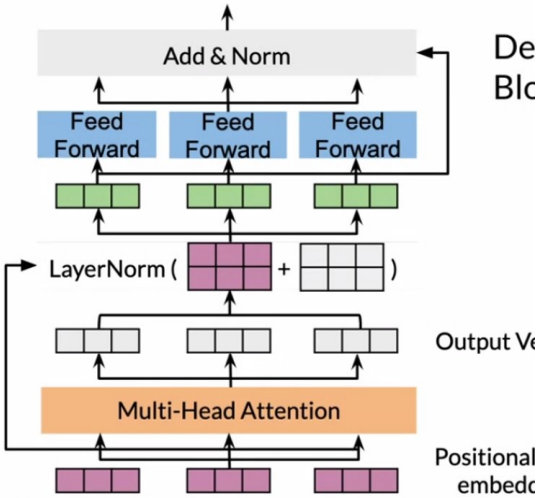
Overview

- input: sentence or paragraph
  - we predict the next word
- sentence gets embedded, add positional encodi
  - (vectors representing $\{0, 1, 2, \ldots, K\}$)
- multi-head attention looks at previous words
- feed-forward layer with ReLU
  - that's where most parameters are!
- residual connection with layer normalization
- repeat N times
- dense layer and softmax for output

And then you have a final dense layer for outputs and a softmax layer, and

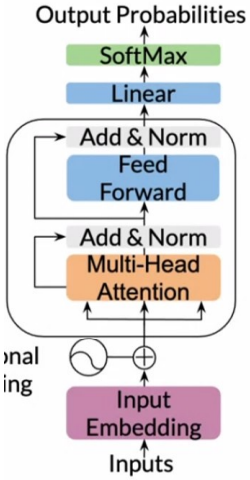
e Transformer decoder





Finally, the encoder layer output is obtained.

chnical details for data processing



Model Input:

```
ARTICLE TEXT <EOS> SUMMARY <EOS> <pad> ...
```

Tokenized version:

```
[2,3,5,2,1,3,4,7,8,2,5,1,2,3,6,2,1,0,0]
```

Loss weights: 0s until the first <EOS> and th
1 on the start of the summary.

you weight the loss for
the words within the article with 0s, and

plearning.ai

erence with a Language Model

lodel input:

```
rticle] <EOS>  [Summary] <EOS>
```

nference:

Provide: [Article] <EOS>

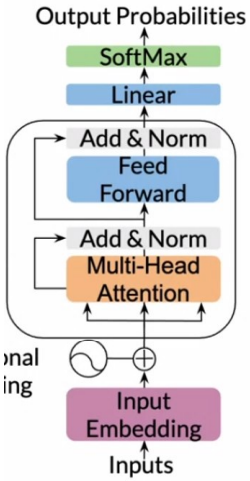
Generate summary word-by-word
- until the final <EOS>

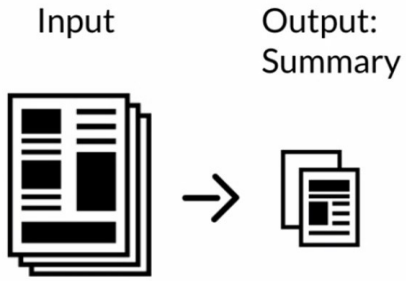
Pick the next word by random sampling
- each time you get a different summary!

it generates a probability
distribution over all possible words.

plearning.ai

ansformer for summarization

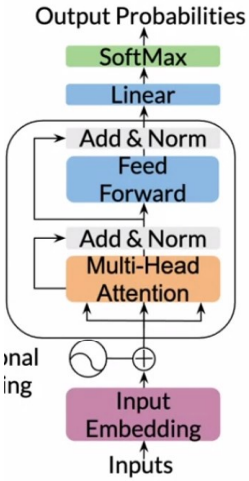




But the one thing may
immediately stand out to you.

plearning.ai

st function



Cross entropy loss

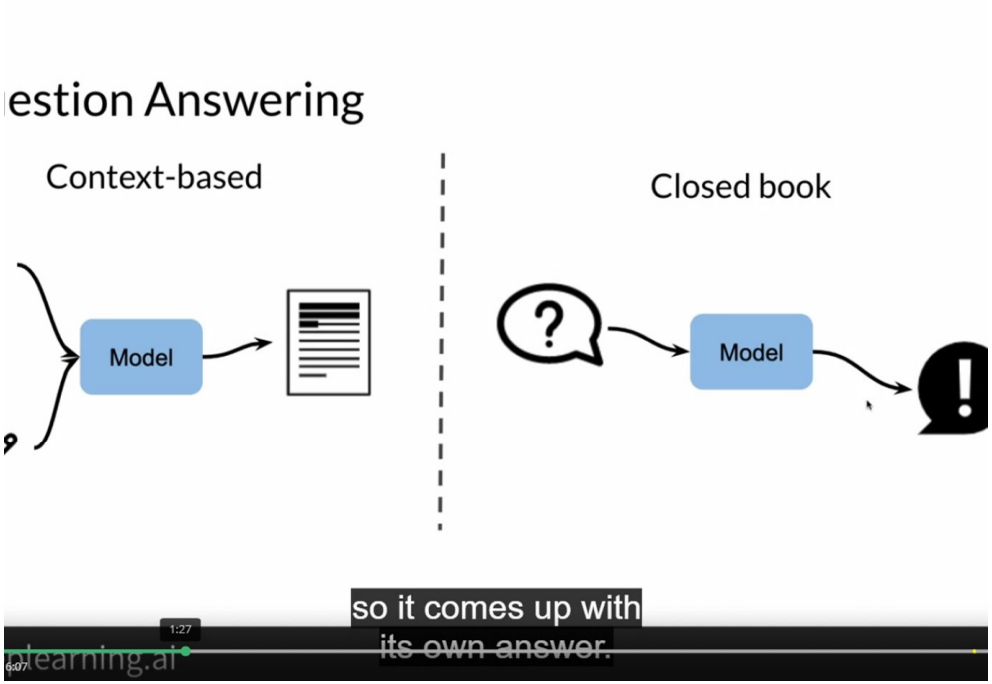
$$J = -\frac{1}{m}\sum_{j}^{m}\sum_{i}^{K} y_j^i \log \hat{y}_j^i$$

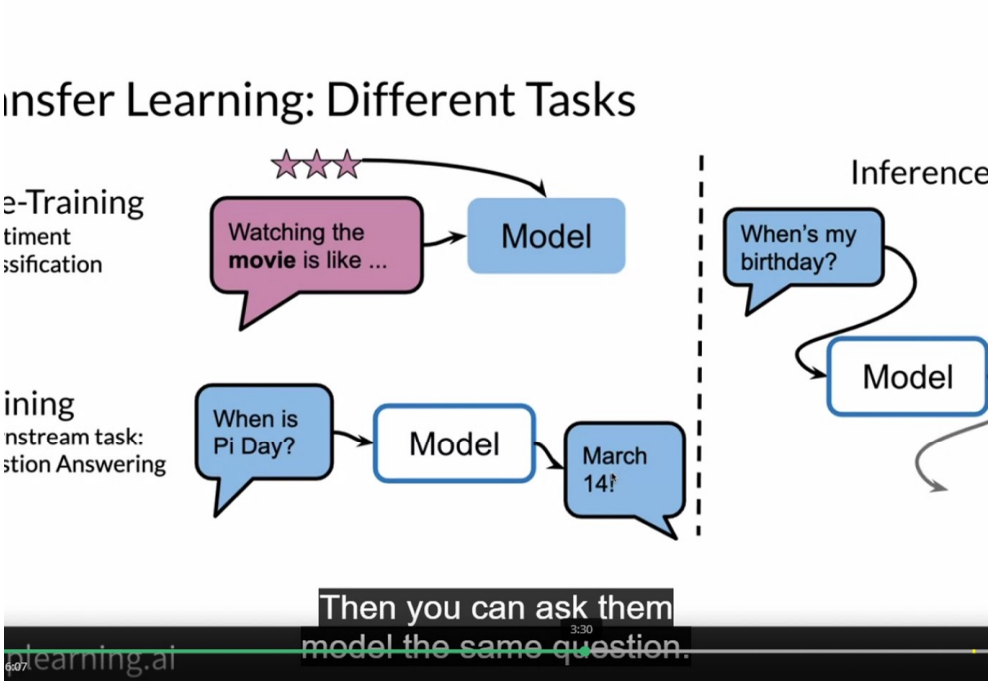
j : over summary

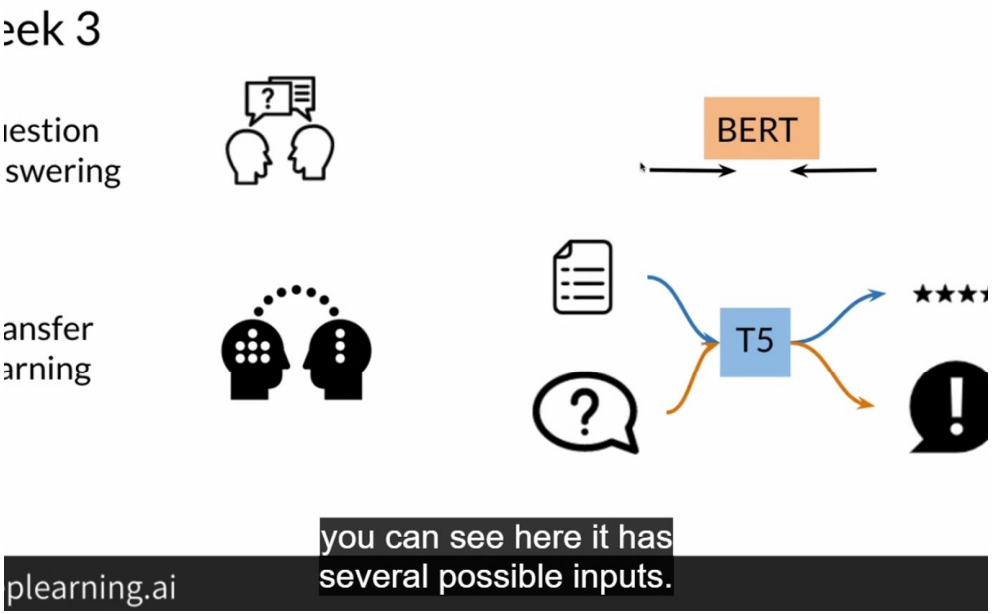
i : bach elements

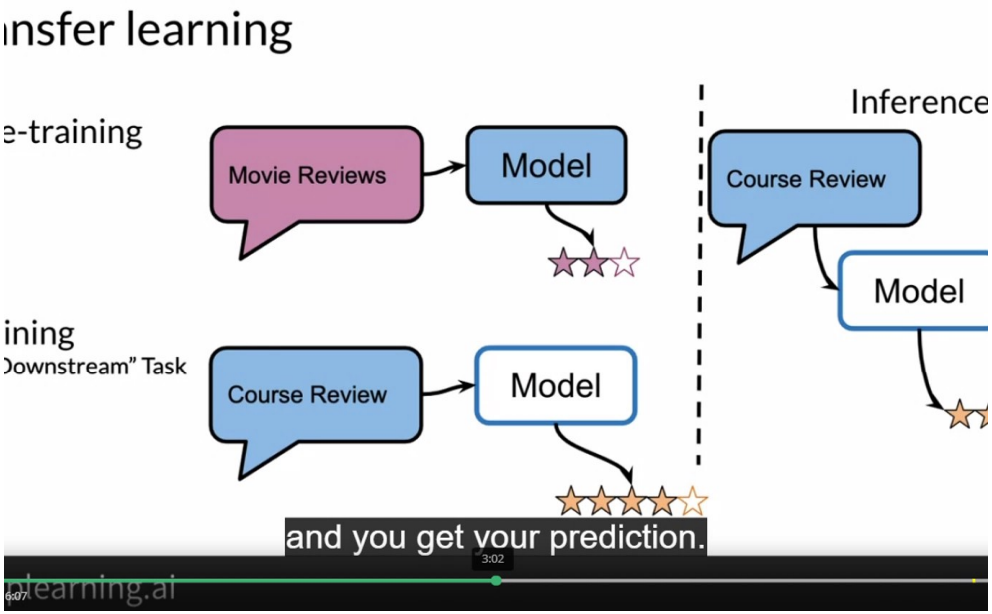
the article to be summarized.

plearning.ai


estion Answering
Context-based
Closed book
Model
Model
so it comes up with
its own answer.
1:27


nsfer Learning: Different Tasks
e-Training
timent
ssification
Watching the movie is like ...
Model
ining
nstream task:
stion Answering
When is Pi Day?
Model
March 14!
Inference
When's my birthday?
Model
Then you can ask them
model the same question.
3:30


eek 3
uestion
swering
ansfer
arning
BERT
T5
you can see here it has
several possible inputs.


nsfer learning
e-training
Movie Reviews
Model
ining
Downstream" Task
Course Review
Model
Inference
Course Review
Model
and you get your prediction.
3:02

: more data, better performance

English wikipedia
~13 GB

C4
Colossal **C**lean **C**rawled
Corpus
~800 GB

is that the more data you have,

sirable Goals

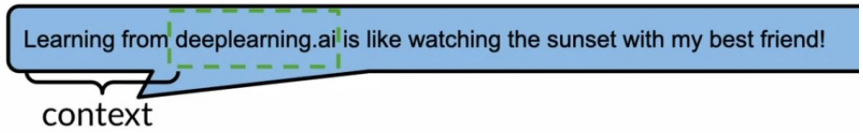
- Reduce training time
- Improve predictions
- Small datasets

Transfer Learning!

a lot from other tasks.

RT: Bi-directional Context

-directional



directional

This is one of the main
takeaways for bidirectionality.

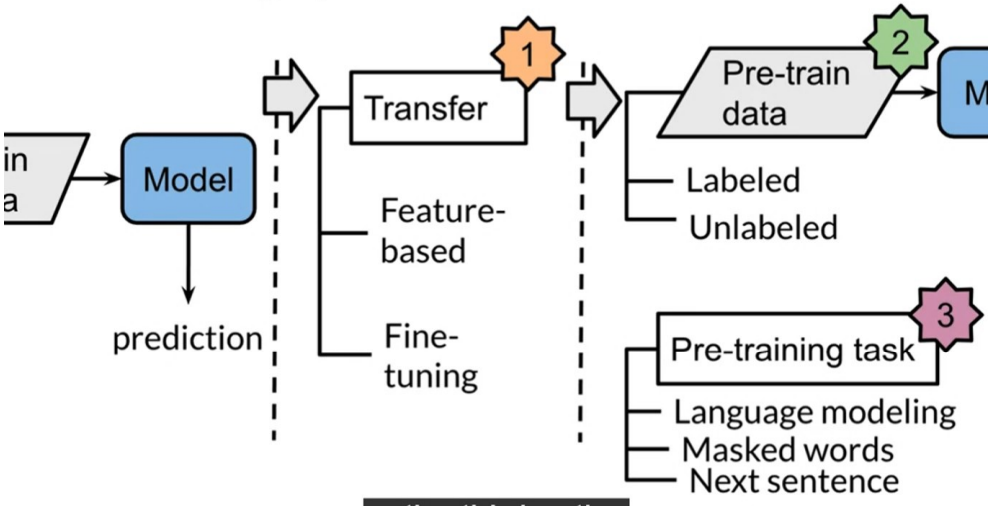
: more data, better performance

English wikipedia
~13 GB

C4
Colossal **C**lean **C**rawled
Corpus
~800 GB

when compared to the
English Wikipedia.

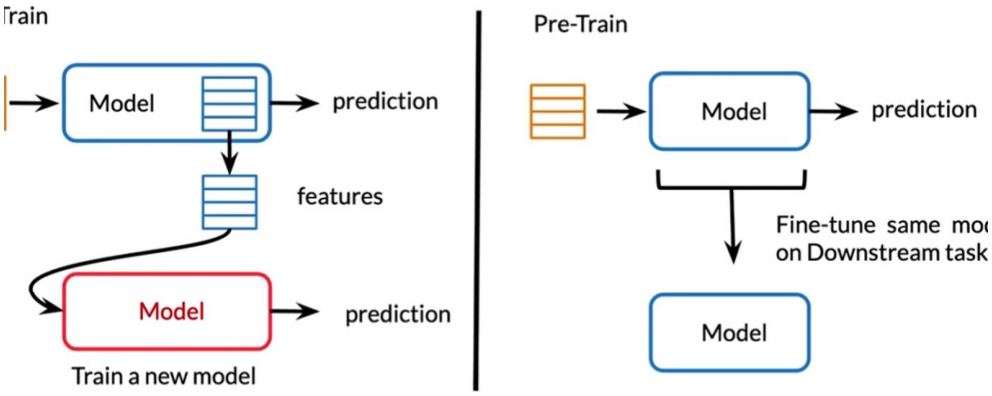
nsfer learning options



or the third option
is pre-training task,

Transfer 1

ture-based vs. Fine-Tuning



fine tune the same model
on a downstream tasks.

sirable Goals

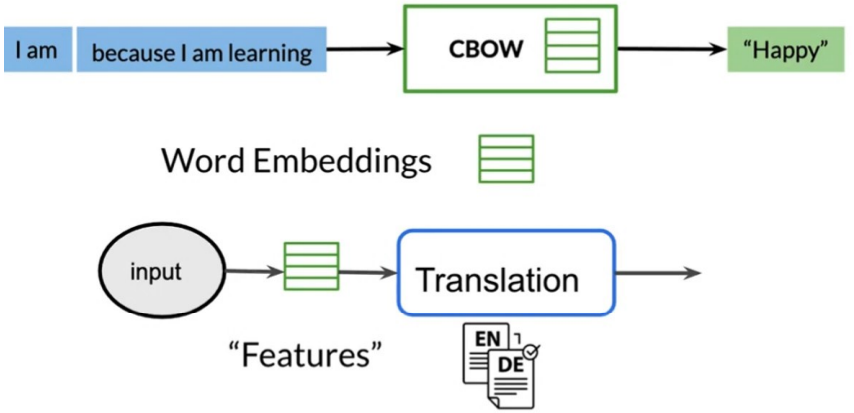
- Reduce training time
- Improve predictions
- Small datasets

Transfer Learning!

We can improve on
these three goals with

Transfer 1

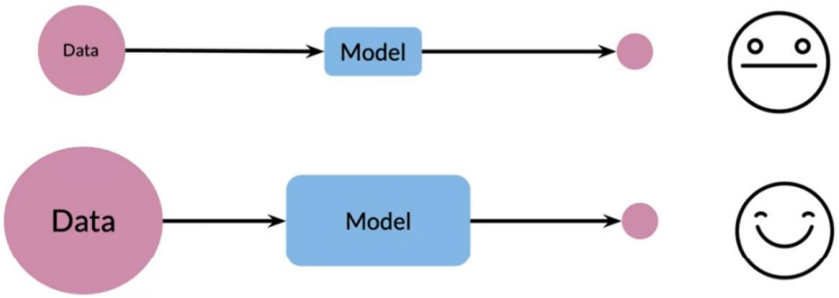
neral purpose learning



it can be used to provide more
meaning to the raw inputs.

:a and performance

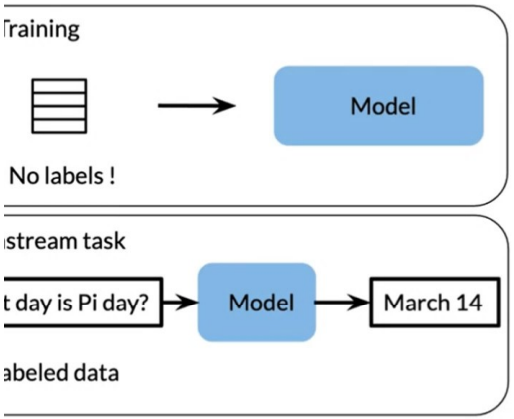
Pre-train data



the bigger models you can build,

nsfer learning with unlabeled data

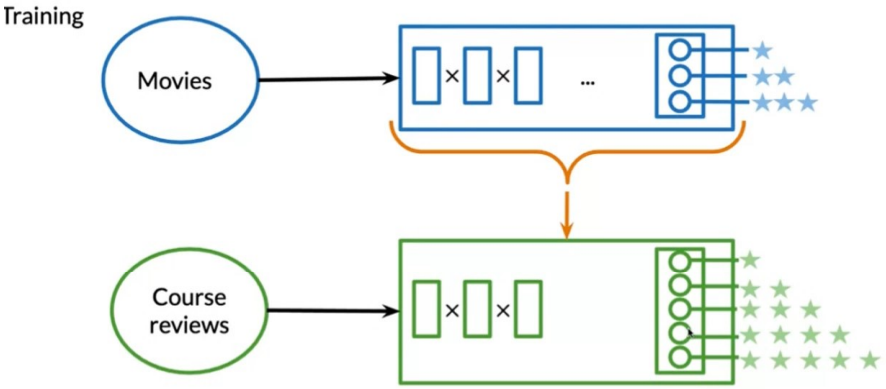
Pre-train data



for example, over here,

e-tune: adding a layer

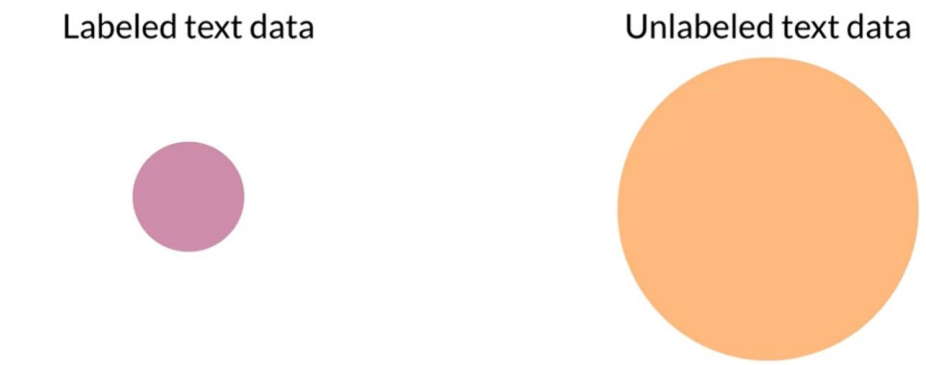
Transfer



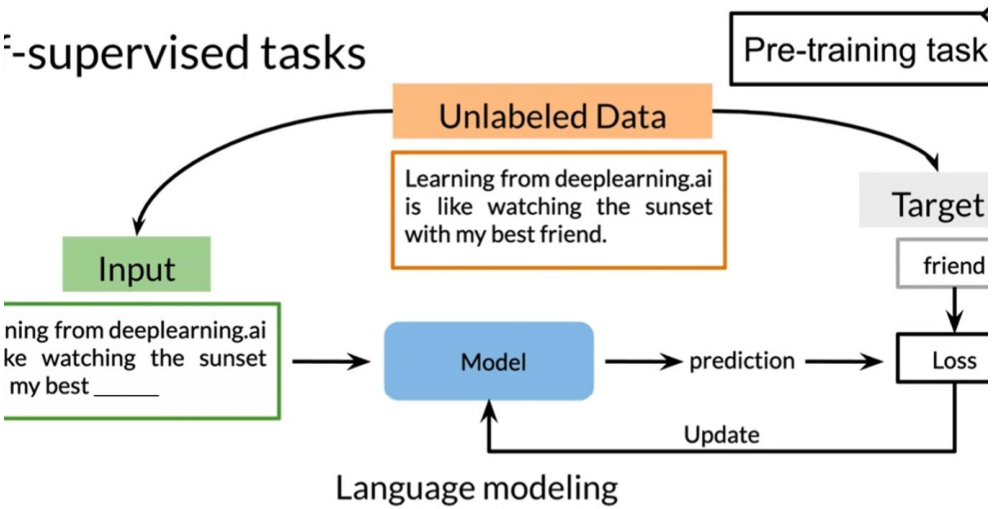
Then you create a new output layer,

eled vs Unlabeled Data

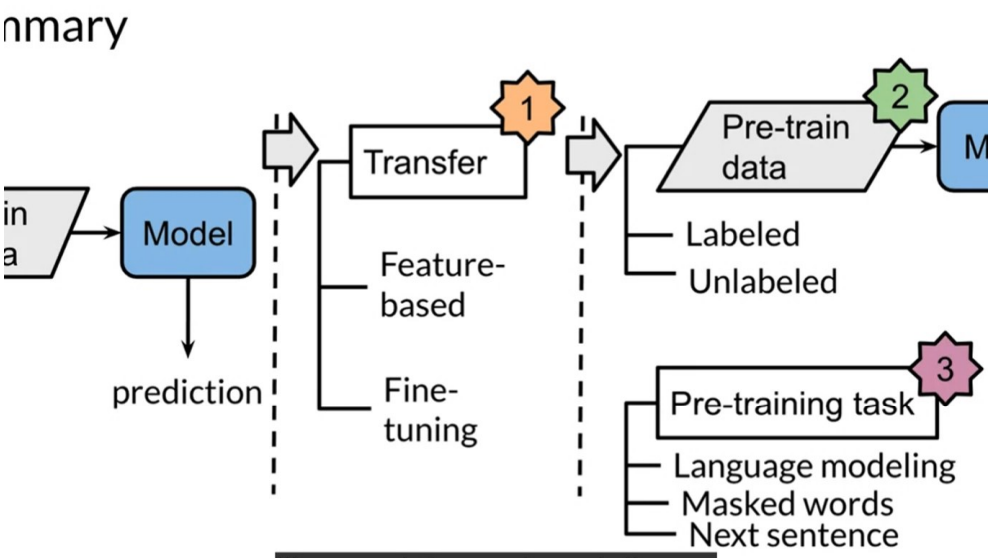
Pre-train data



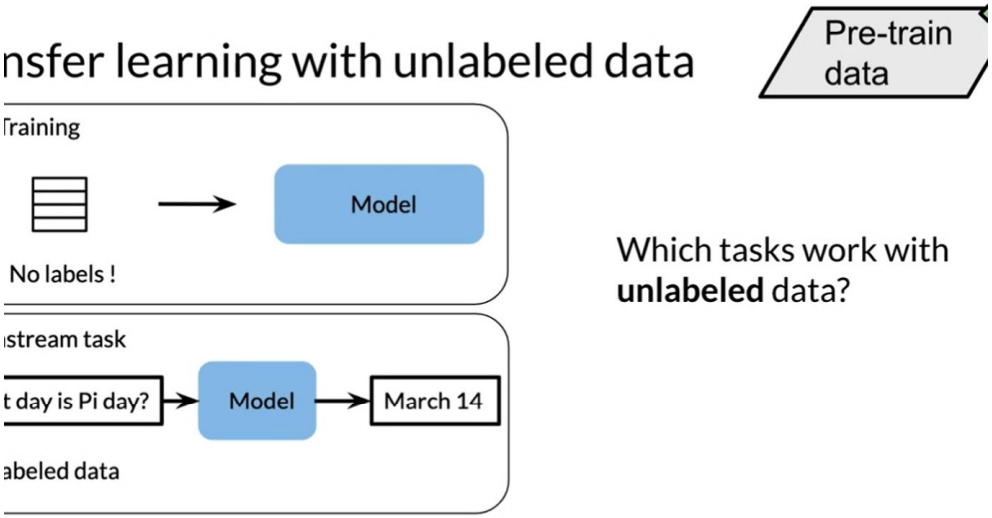
and as you can see that there is way



Then as you can see, the targets is friend,

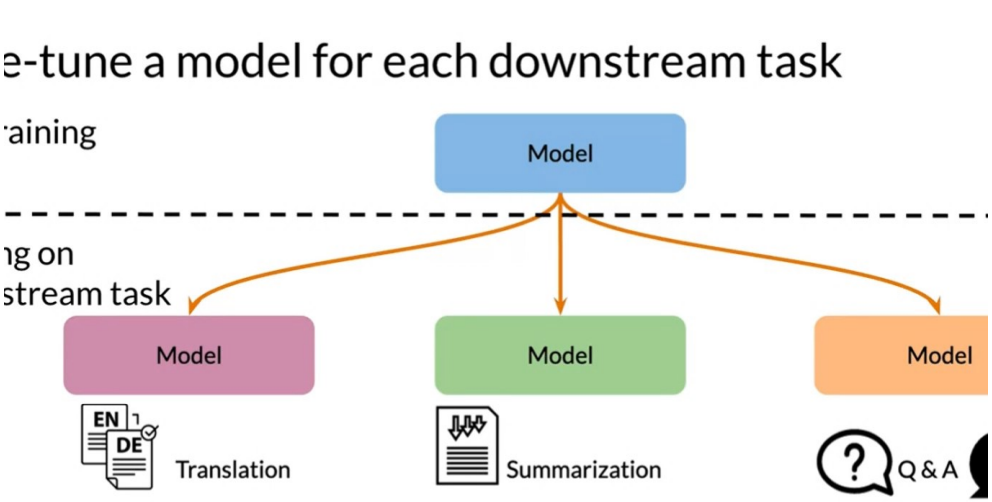


like mask toward modeling or next sentence prediction.



Which tasks work with **unlabeled** data?

The question is, which tasks work with unlabeled data?



or to do question answering.

tline

:BOW ELMo GPT BERT T5

were solving for and how they

plearning.ai

ntinuous Bag of Words

... they were on the right side of the street

Fixed window Fixed window

'on"
'the"
'side"
'of"

"right"

Fully-connected (Feed Forward) neural network

which is the center word.

1:17

learning.ai

Press Esc to exit full screen

deeplearning.ai

ELMo, GPT
BERT, T5

Now you will get a overview
of some models such as ElMo,

7:07

ntext

... right ...

... they were on the right ...

... they were on the right side of the street

the context after
the word and we use

learning.ai

ed more context?

… they were on the right side of the street.

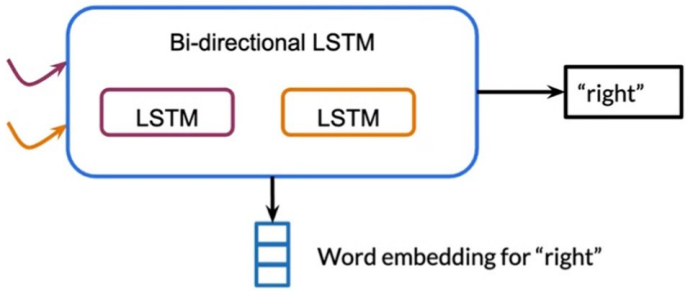
Fixed window Fixed window

… they were on the right side of history.

it could be instead of the streets,

Mo: Full context using RNN

gislators believed that they were on the ____ side of history so they changed the law.

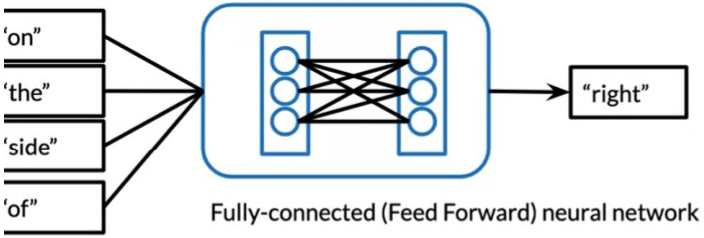


You get the word embedding for

ntinuous Bag of Words

… they were on the right side of the street

Fixed window Fixed window

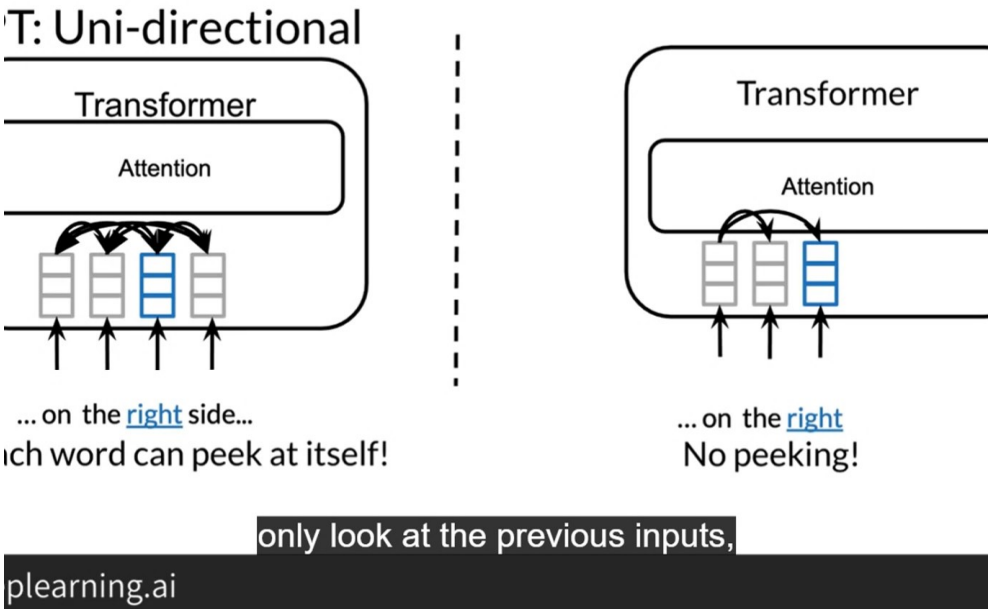


a neural network and you predict the outputs,

e all context words

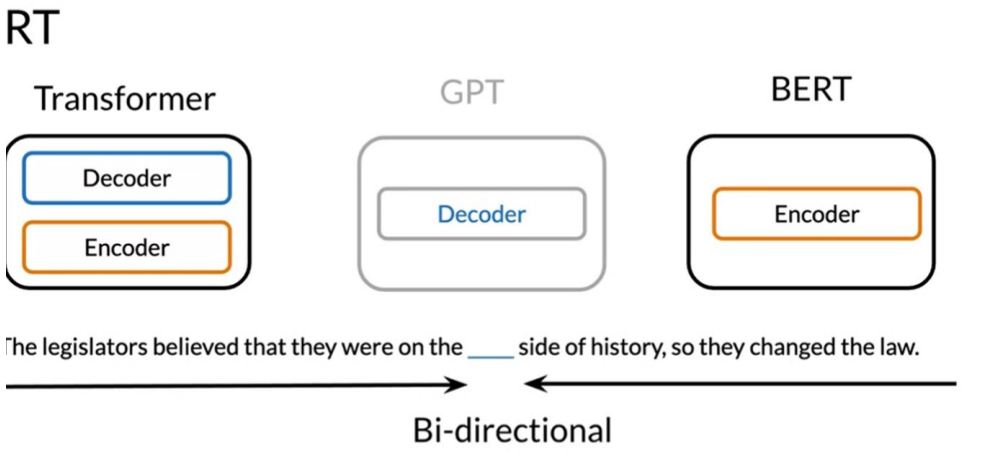
legislators believed that they were on the right side of history, so they changed the lav

Over here, it starts from



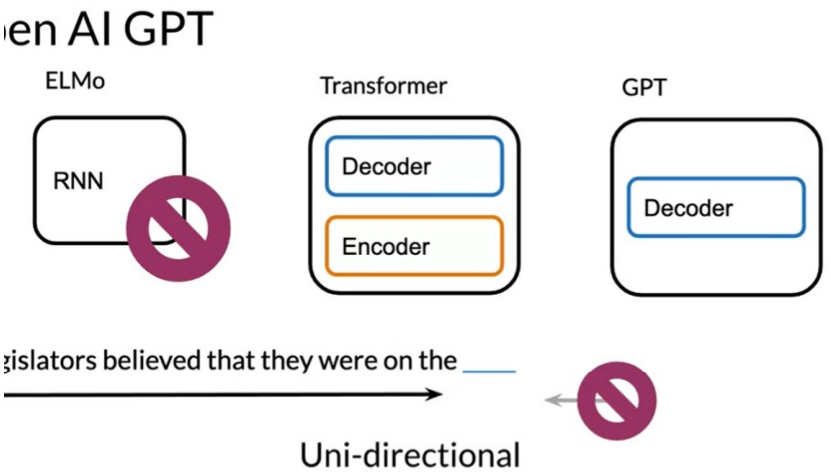
only look at the previous inputs,

plearning.ai



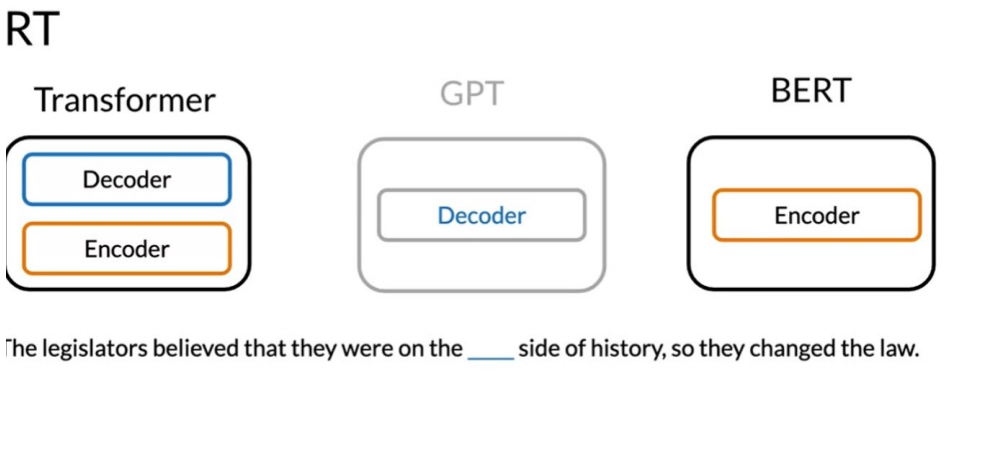
bi-directional representations

plearning.ai



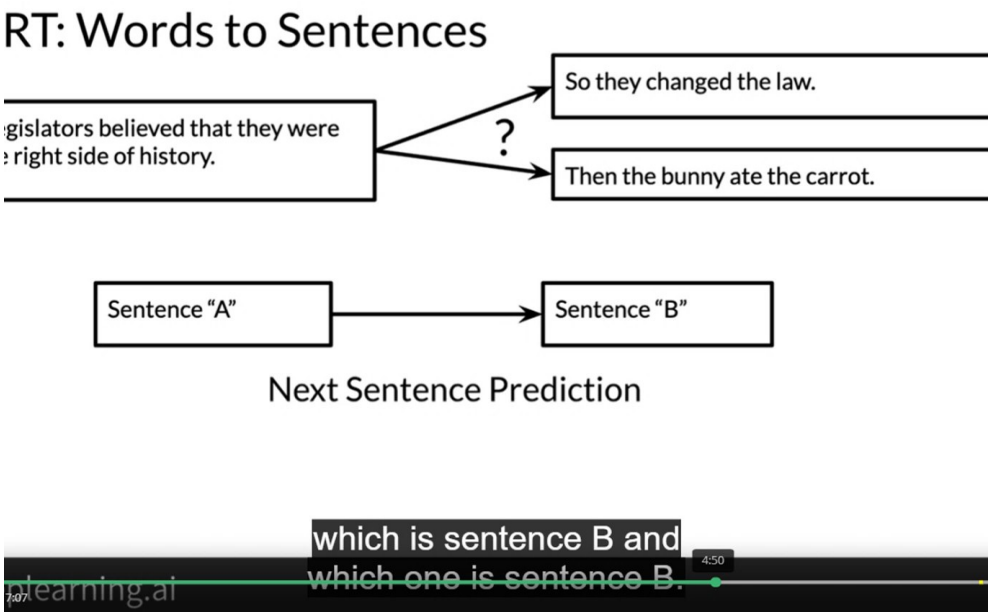
as capturing longer-term dependencies.

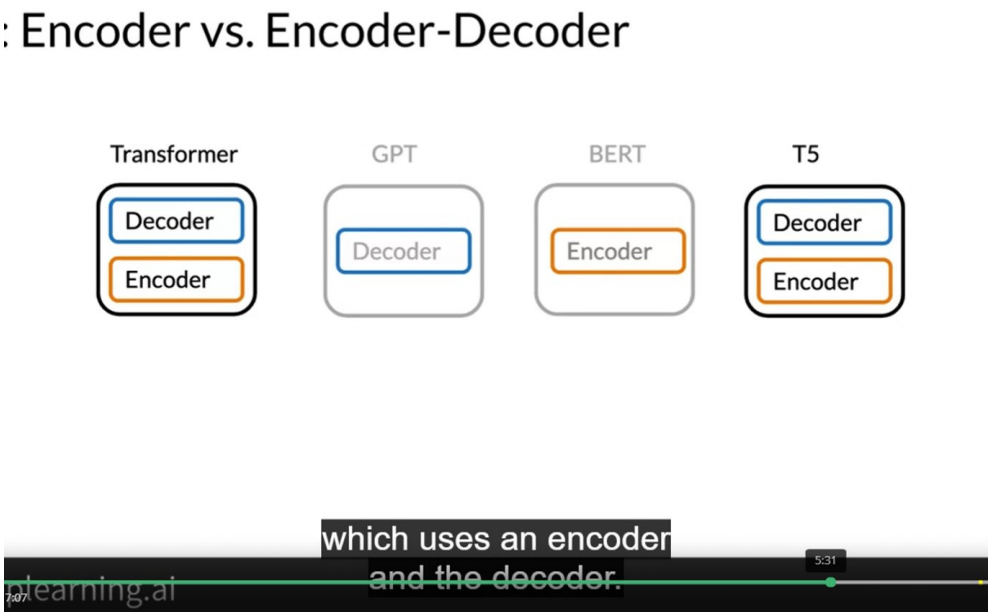
plearning.ai

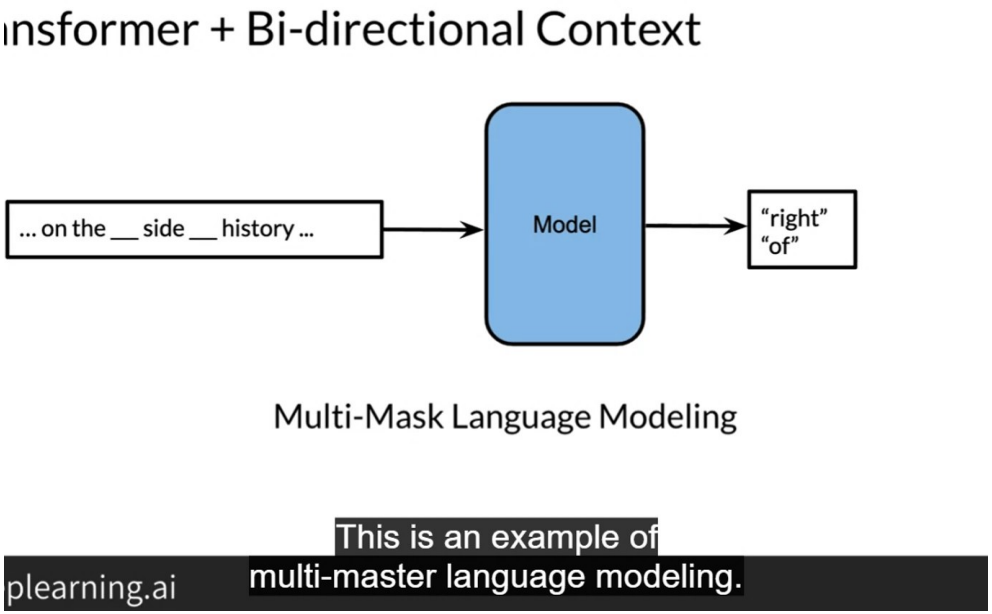


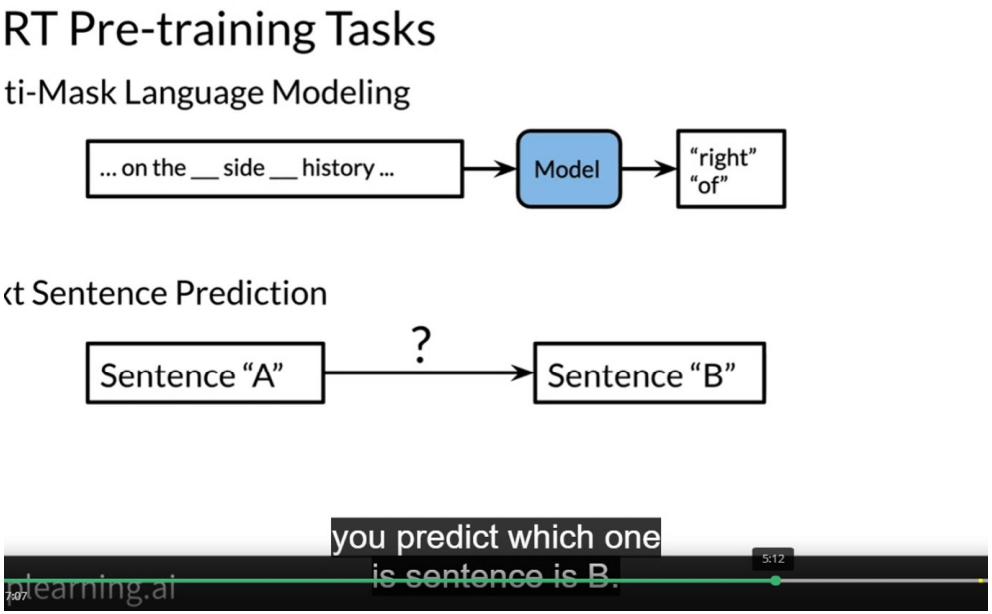
As it works over here,

plearning.ai

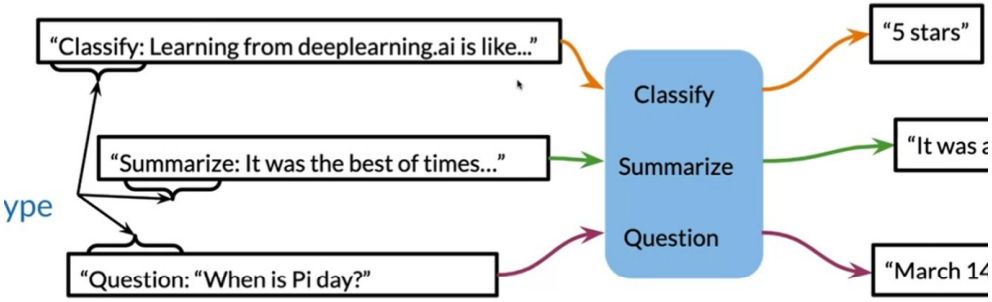

RT: Words to Sentences
gislators believed that they were
right side of history.
?
So they changed the law.
Then the bunny ate the carrot.
Sentence “A”
Sentence “B”
Next Sentence Prediction
which is sentence B and
which one is sentence B.
4:50
plearning.ai


Encoder vs. Encoder-Decoder
Transformer
GPT
BERT
T5
Decoder
Encoder
Decoder
Encoder
Decoder
Encoder
which uses an encoder
and the decoder.
5:31
plearning.ai


nsformer + Bi-directional Context
... on the ___ side ___ history ...
Model
“right”
“of”
Multi-Mask Language Modeling
This is an example of
multi-master language modeling.
plearning.ai


RT Pre-training Tasks
ti-Mask Language Modeling
... on the ___ side ___ history ...
Model
“right”
“of”
xt Sentence Prediction
?
Sentence “A”
Sentence “B”
you predict which one
is sentence is B.
5:12
plearning.ai

: Text-to-Text



ype

basically tell the model
that's it's going to classify

5:50

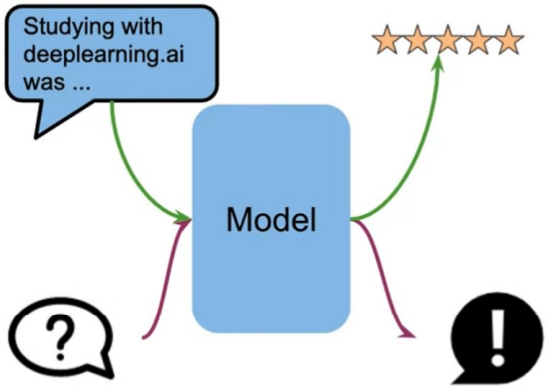
)utline

Learn about the BERT architecture

Understand how BERT pre-training works

the BERT architecture, and then,

: Multi-task

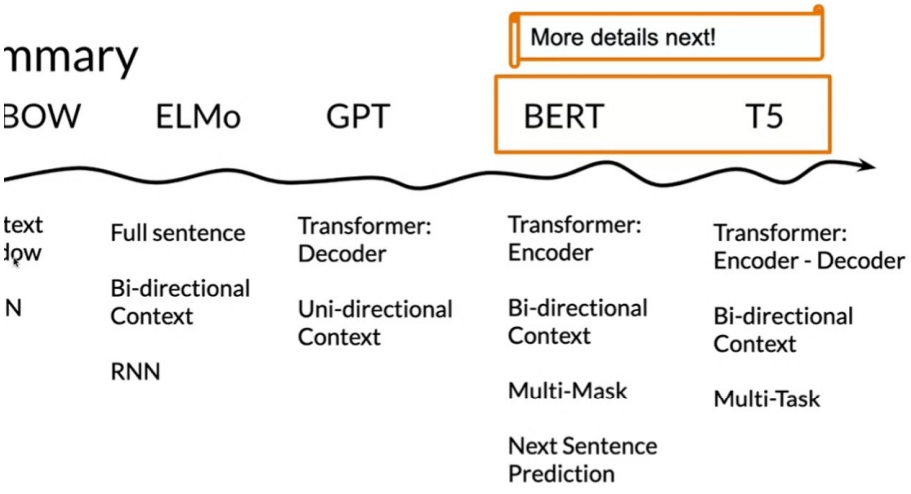


How?

How can you do this?

5:50

mmary



then it used fixed context window

ERT

A multi layer bidirectional transformer

Positional embeddings

BERT_base:

- 12 layers (12 transformer blocks)
- 12 attentions heads
- 110 million parameters

BERT base which has 12 layers or 12 transformer blocks,

ERT pre-training

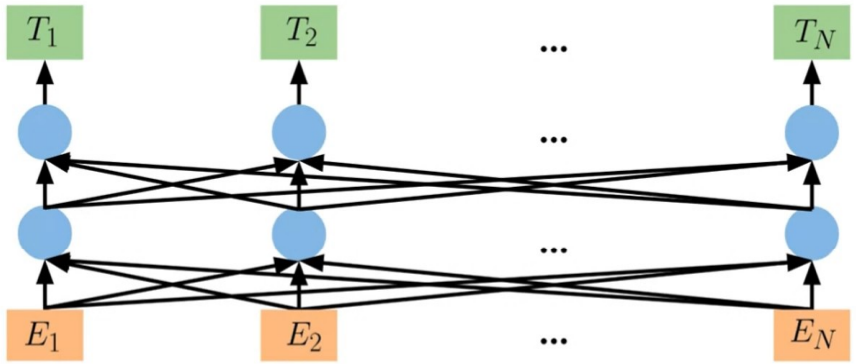
fter school Lukasz does his _____ in the library.

Masked language modeling (MLM)

Maybe work, maybe homework.

ERT

- Makes use of transfer learning/pre-training:



and then, you get your T_1, T_2, T_N.

ERT

A multi layer bidirectional transformer

Positional embeddings

BERT_base:

- 12 layers (12 transformer blocks)
- 12 attentions heads
- 110 million parameters

They have way, way more paramoters and way,

ummary

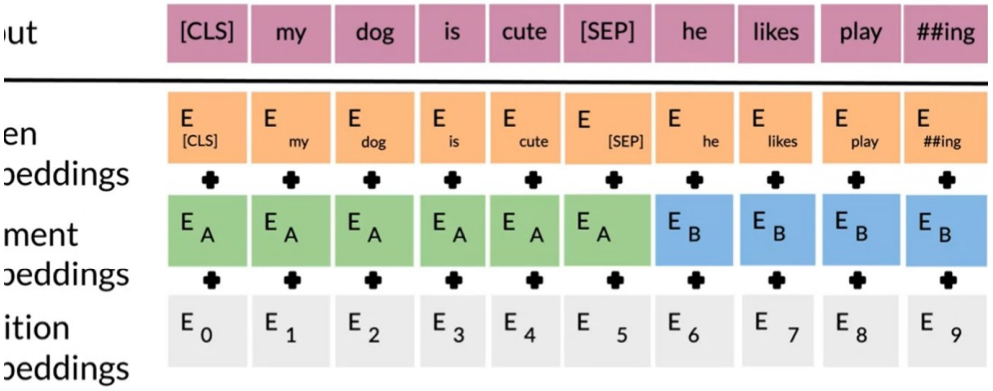
Choose 15% of the tokens at random: mask them 80% of the time, replace them with a random token 10% of the time, or keep as is 10 the time.

There could be multiple masked spans in a sentence

Next sentence prediction is also used when pre-training.

15 percent of the tokens at random.

rmalizing the input



and then you get your new inputs.

ERT pre-training

fter school Lukasz does his _____ in the library.

Masked language modeling (MLM)

by the embedding matrix, and then,

tline

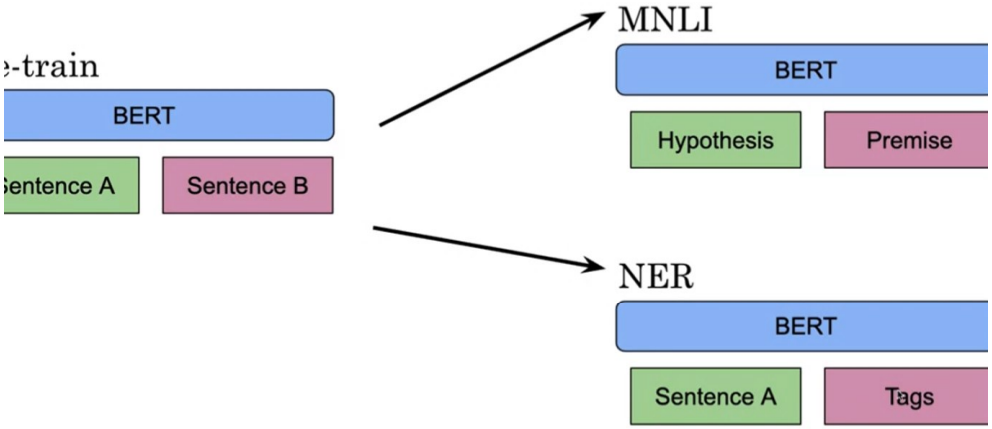
Understand how BERT inputs are fed into the model

Visualize the output

Learn about the BERT objective

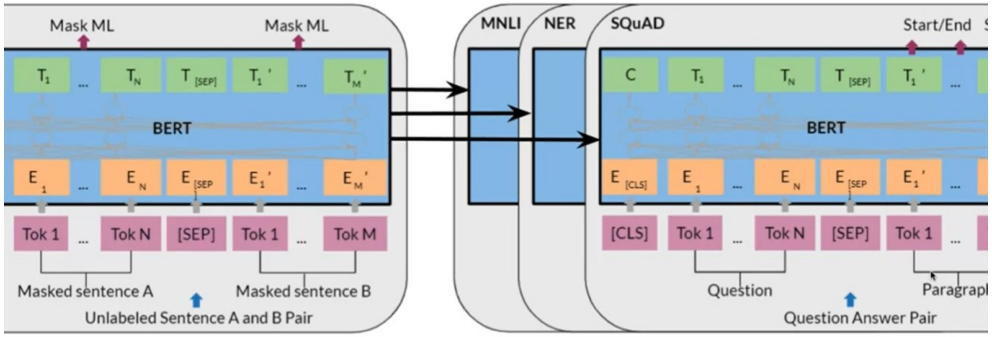
You're going to learn how Bert inputs are fed into

ne-tuning BERT: Outline



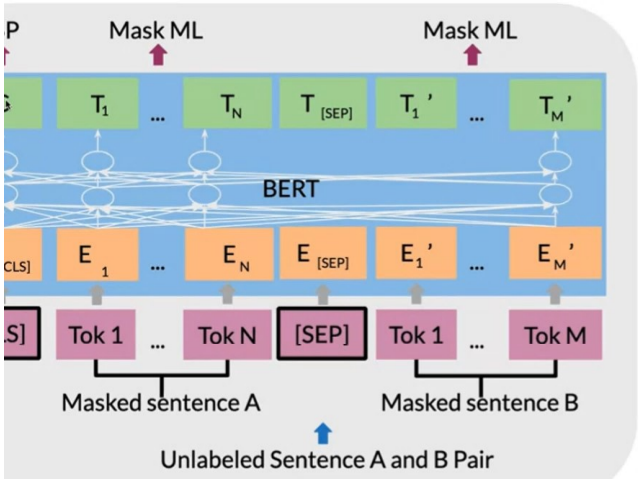
and then the corresponding tags over here.

outs



Over here, you'll have the paragraph and this will give

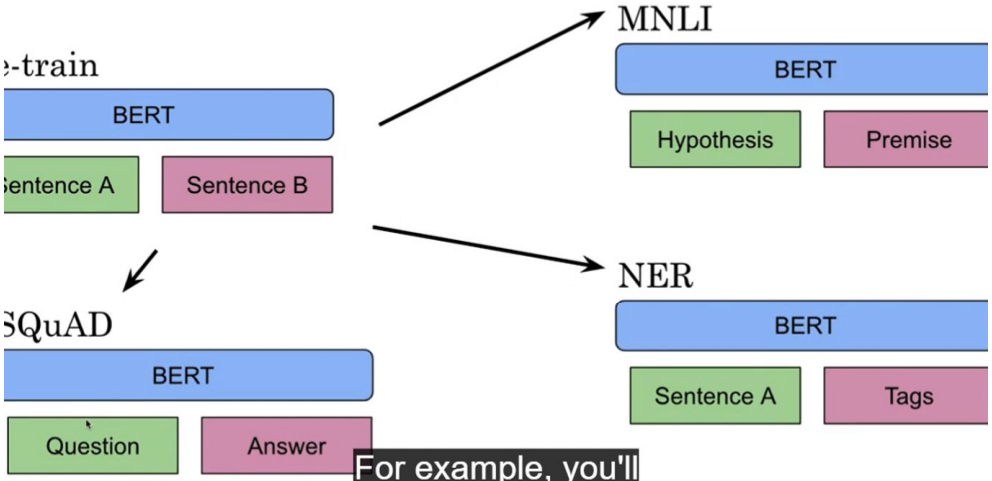
ualizing the output



- [CLS]: a spe classification sym added in front every input
- [SEP]: a spe separator token

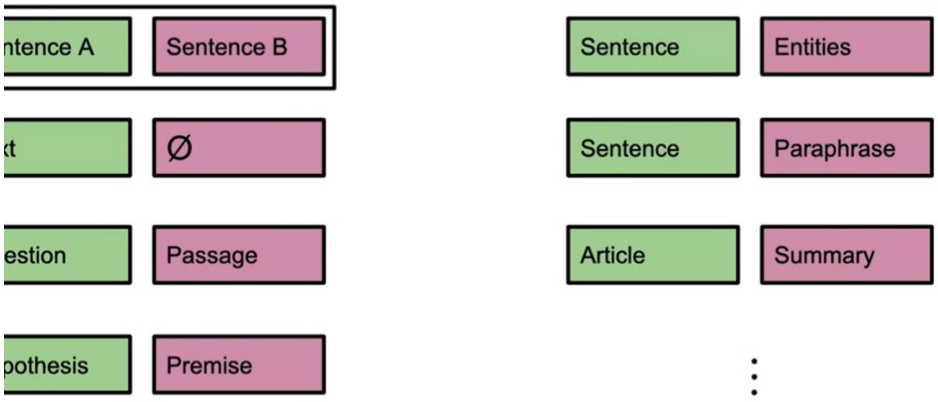
embedding which can be used for next sentence prediction.

ne-tuning BERT: Outline



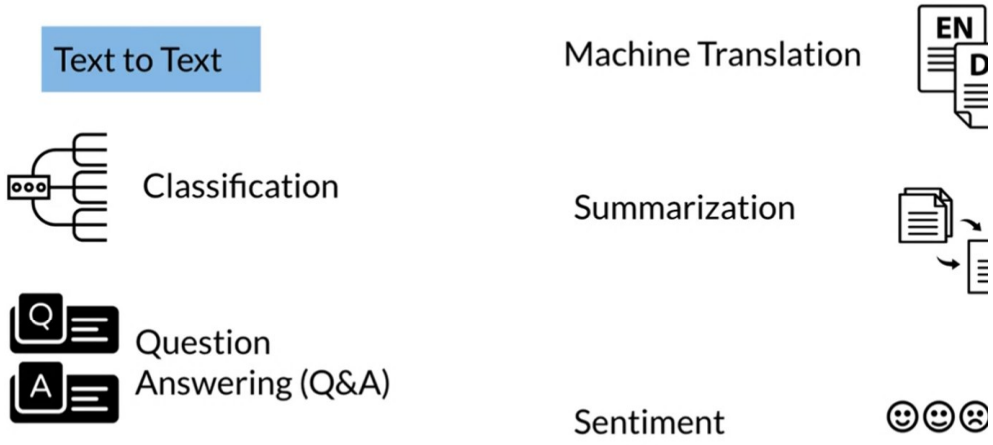
For example, you'll have your question over

mmary



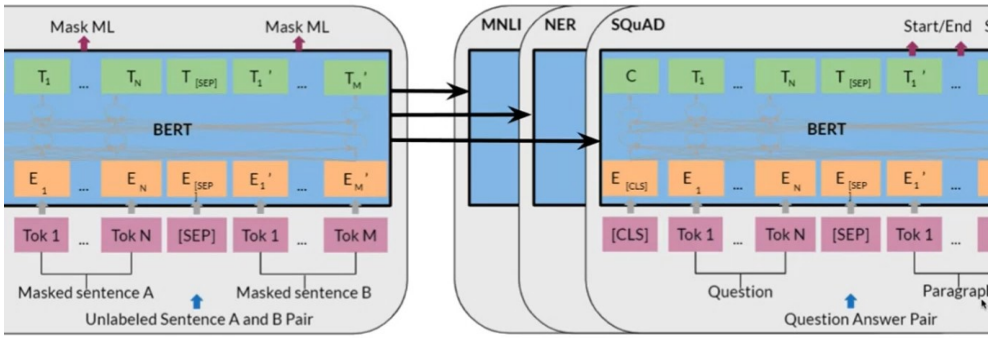
This is just the inputs into your BERT's model.

Transformer - T5 Model



And there are other applications that you can use T5 for, but

puts



the hypothesis and then the premise and so forth.

Outline

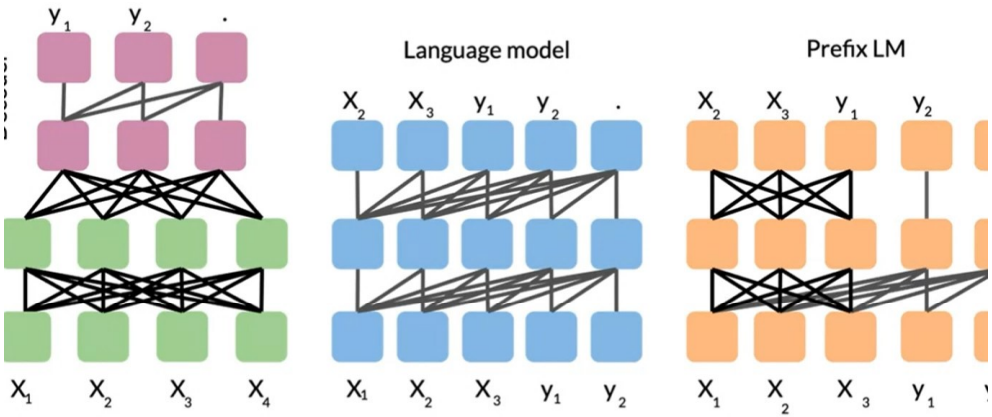
Understand how T5 works

Recognize the different types of attention used

Overview of model architecture

So we're going to understand how T5 works.

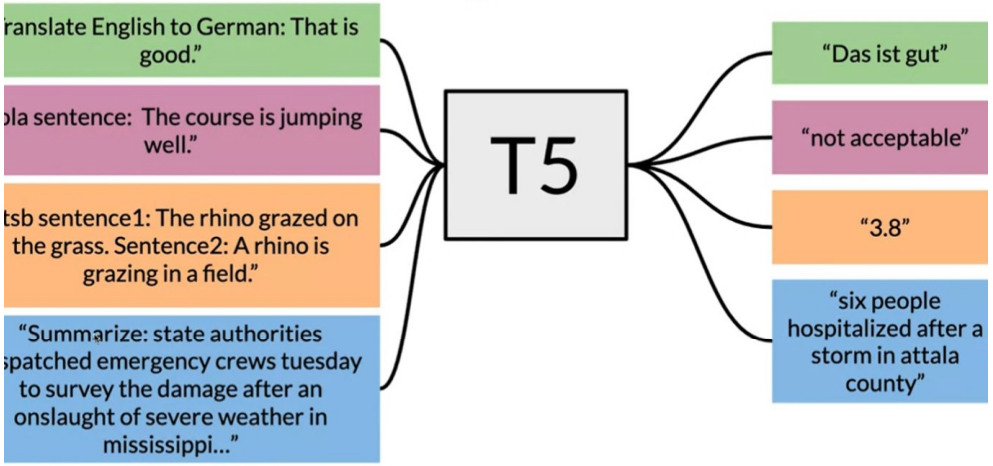
Model Architecture



So as you can see over here, it's doing causal masking.

plearning.ai

Multi-task training strategy



This is how it works.

Transformer - T5 Model

Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs

Thank you <X> me to your party <Y> week.

Targets

<X> for inviting <Y> last <Z>

So each bracket corresponds to a certain target.

plearning.ai

Summary

Prefix LM attention

Model architecture

Pre-training T5 (MLM)

You've seen the model architecture for T5.

plearning.ai

ulti-task Training Strategy

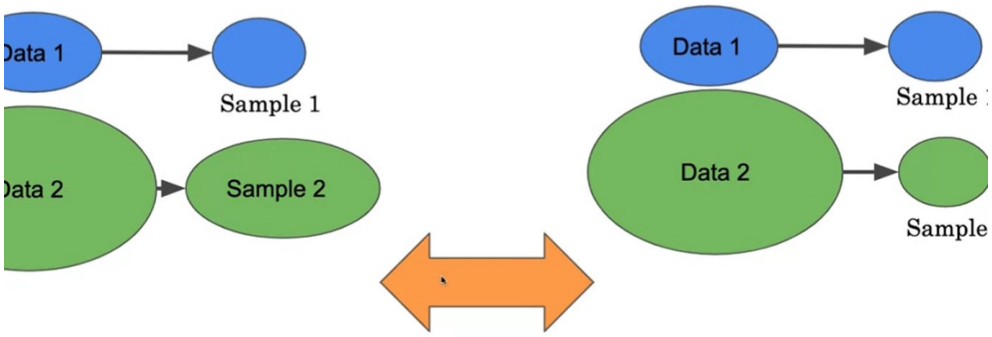
ine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
ll parameters	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
dapter layers, $d = 32$	80.52	15.08	79.32	60.40	13.84	17.88	15.54
dapter layers, $d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
dapter layers, $d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
dapter layers, $d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
radual unfreezing	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93

e Limits of Transfer learning with a unified text t
this is a table found
in the original paper,
2:40
5:57
learning.ai

ta Training Strategies

mples-proportional mixing

Equal mixing



Temperature-scaled mixing

the parameters to get
something in-between.
3:57
5:57
learning.ai

put and Output Format

lachine translation:

- translate English to German: That is good.

Predict entailment, contradiction , or neutral

- mnli premise: I hate pigeons hypothesis: My feelings towards pigeons are filled with animosity. target: entailment

Winograd schema

- The city councilmen refused the demonstrators a permit because *they* feared violence

e Limits of Transfer learning with a unified t
which is another way to
predict whether a pronoun,
plearning.ai

put and Output Format

lachine translation:

- translate English to German: That is good.

Predict entailment, contradiction , or neutral

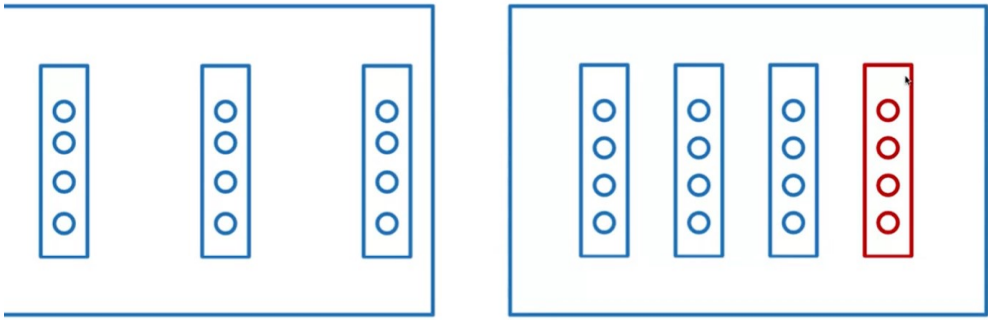
- mnli premise: I hate pigeons hypothesis: My feelings towards pigeons are filled with animosity. target: entailment

Winograd schema

- The city councilmen refused the demonstrators a permit because *they* feared violence

e Limits of Transfer learning with a un
your model and then it will be
2:32
5:57
learning.ai

adual unfreezing vs. Adapter layers



Gradual unfreezing Adapter layers

e Limits of Transfer learning with a unified t

add a neural network to

plearning.ai

eneral Language Understanding Evaluation

A collection used to train, evaluate, analyze natural language understanding systems

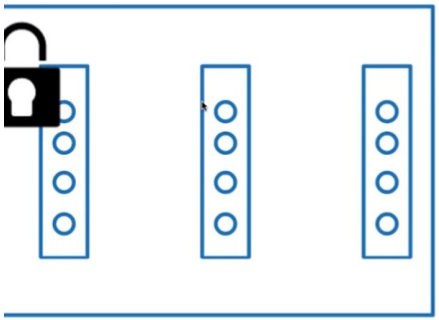
Datasets with different genres, and of different sizes and difficulties

Leaderboard

So people can use the data sets, and

plearning.ai

adual unfreezing vs. Adapter layers



Gradual unfreezing

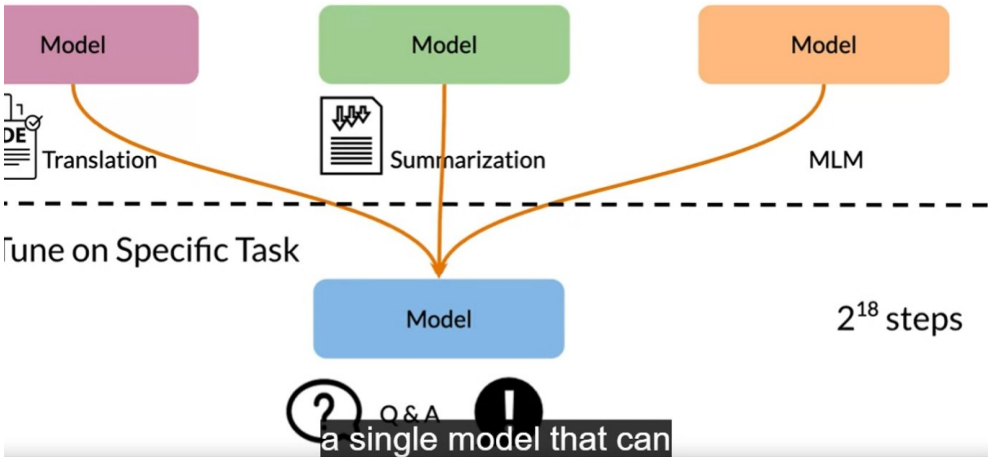
e Limits of Transfer learning with a un

Then you freeze this one and then you freeze this one,

plearning.ai

ne-tuning

aining



a single model that can

plearning.ai

eneral Language Understanding Evaluation

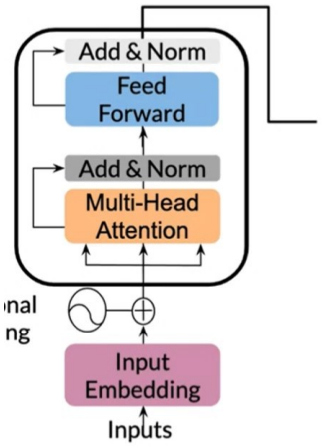
Drive research

Model agnostic

Makes use of transfer learning

because you have access to several datasets.

ansformer encoder



Feedforward:

```
[
    LayerNorm,
    dense,
    activation,
    dropout_middle,
    dense,
    dropout_final
]
```

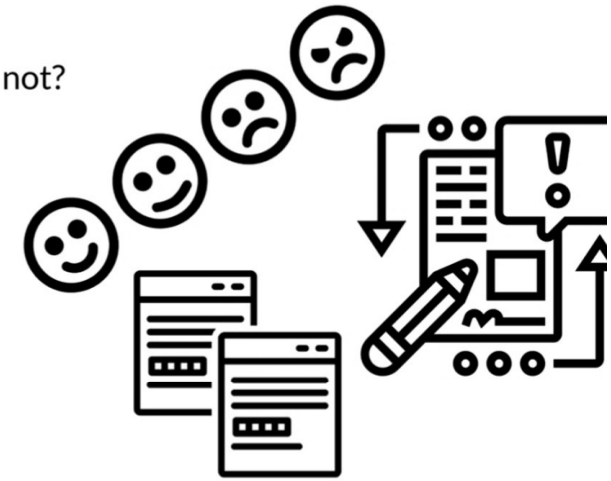
Encoder block:

```
[
    Residual(
        LayerNorm
        attention
        dropout_,
    ),
    Residual(
        feed_forw
    )
]
```

the encoder block which has the feed forward and two residual connections.

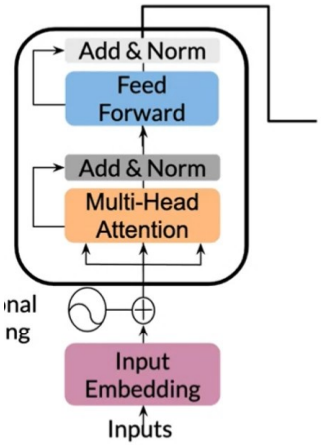
sks Evaluated on

Sentence grammatical or not?
Sentiment
Paraphrase
Similarity
Questions duplicates
Answerable
Contradiction
Entailment
Winograd (co-ref)



And also for the Winograd Schema, which is basically trying to identify whether,

ansformer encoder



Feedforward:

```
[
    LayerNorm,
    dense,
    activation,
    dropout_middle,
    dense,
    dropout_final
]
```

you have a layerNorm, a dense, followed by an activation.

plementing Q&A with T5

Load a pre-trained model

Process data to get the required inputs and outputs: "question: Q context: C" as input and "A" as target

Fine tune your model on the new task and input

Predict using your own model

the inputs, and
then you predict using your own model.

plearning.ai

ıg Text Sequences

sks In NLP:

Writing Books

Chatbots

building intelligent agents for
conversations like chatbots.

plearning.ai

ıta examples

estion: What percentage of the French population today is non - European ?

ıtext: Since the end of the Second World War , France has become an ethnically diverse country .

roximately five percent of the French population is non - European and non - white . This d
ach the number of non - white citizens in the United States (roughly 28 – 37 % , depending on how Latinos are cl
Demographics of the United States) . Nevertheless , it amounts to at least three million people , and has forced th
nic diversity onto the French policy agenda . France has developed an approach to dealing with ethnic proble
s in contrast to that of many advanced , industrialized countries . Unlike the United States , Britain , or ev
erlands , France maintains a " color - blind " model of public policy . This means that it targets virtually no policies
ial or ethnic groups . Instead , it uses geographic or class criteria to address issues of social inequalities . It has , ho
oped an extensive anti - racist policy repertoire since the early 1970s . Until recently , French policies focused pr
sues of hate speech — going much further than their American counterparts — and relatively less on is
mination in jobs , housing , and in provision of goods and services .

get: Approximately five percent

You'll have a context and
then you'll have a target.

1:43

plearning.ai

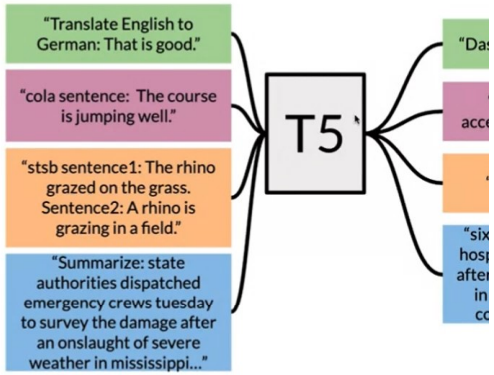
plementing Q&A with T5

Load a pre-trained model

Process data to get the required inputs and outputs: "question: Q context: C" as input and "A" as target

Fine tune your model on the new task and input

Predict using your own model



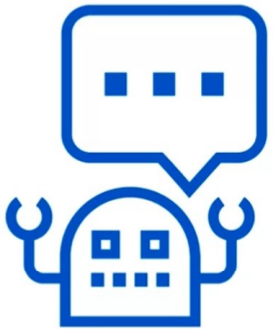
And remember this is the T5
that you'll be working with.

plearning.ai

atbots

ontext Windows:

User 1: What's for dinner?
Chatbot: Who's cooking, you or me?
User 1: Hey now chatbot.
Chatbot: I hope it's not hay, that's what horses eat.



It's going to be fun.

nsformer Issues

Attention on sequence of length L takes L^2 time and memory

L=100	L^2 = 10K	(0.001s at 10M ops/s)
L=1000	L^2 = 1M	(0.1s at 10M ops/s)
L=10000	L^2 = 100M	(10s at 10M ops/s)
L=100000	L^2 = 10B	(1000s at 10M ops/s)

N layers take N times as much memory

GPT-3 has 96 layers and new models will have more

For example, if L equals 100,

ng Text Sequences

sks In NLP:

Writing Books

Chatbots

train a real working chatbot
in this week's assignments.

nsformer Issues

Attention on sequence of length L takes L^2 time and memory

N layers take N times as much memory

GPT-3 has 96 layers and new models will have more

GPT-3, for example,
already has 96 layers,

:mory with N Layers

Activations need to be stored for backprop

Big models are getting bigger

Compute vs memory tradeoff

less memory and I'll show
you how to do that next.

plearning.ai

hat does Attention do?

lect Nearest Neighbors (K,Q) and return corresponding V

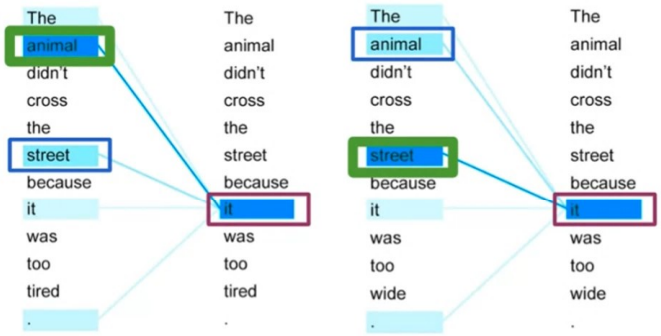


image ©
(Transformer: A N
Network Archite
Language Unders

It again refers to either the streets or
the animal, but

plearning.ai

:ention Complexity

Attention: softmax(QK^T)V

Q, K, V are all [L, d_model]

QK^T is [L, L]

Save compute by using area of interest for large L

and those immediately around
it, by using attention.

plearning.ai

hat does Attention do?

lect Nearest Neighbors (K,Q) and return corresponding V

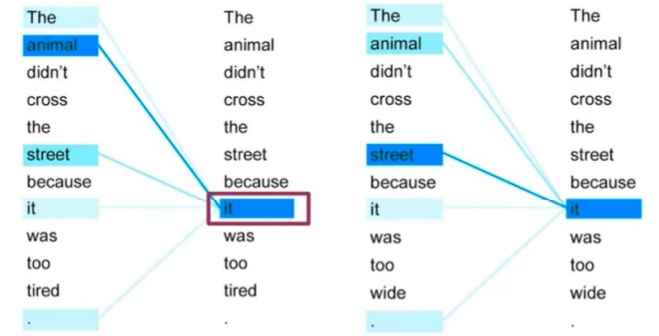


image ©
(Transformer: A N
Network Archite
Language Unders

Take the word it attention is focused on

plearning.ai

arest Neighbors

urse:

Natural Language Processing with Classification and Vector Space

ssons:

- KNN
- Hash Tables and Hash Functions
- Locality Sensitive Hashing
- Multiple Planes

Please feel free to skip ahead if you're familiar with both KNN and LSH.

plearning.ai

arest Neighbors

mpute the nearest neighbor to q among vectors $\{k_1, ..., k_n\}$

Attention computes $d(q, k_i)$ for i from 1 to n which can be slow

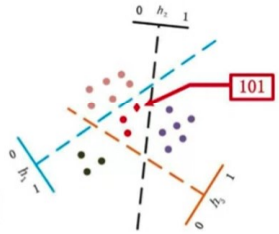
Faster *approximate* uses locality sensitive hashing (LSH)

Locality sensitive: if q is close to k_i:

$hash(q) == hash(k_i)$

Achieve by randomly cutting space

$hash(x) = sign(xR)$ R: [d, n_hash_bins]



When choosing the hash you want to make the buckets roughly the same size.

plearning.ai

hat does Attention do?

lect Nearest Neighbors (K,Q) and return corresponding V

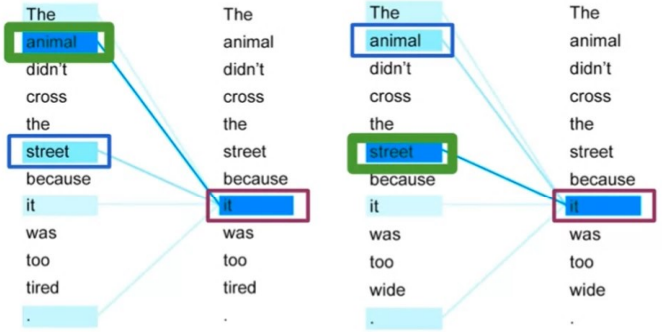


image ©
(Transformer: A N
Network Archite
Language Unders

and you can ignore the other words like it and was and tired.

plearning.ai

arest Neighbors

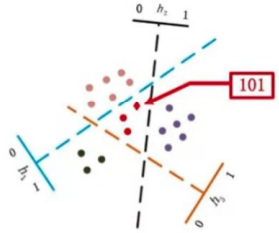
mpute the nearest neighbor to q among vectors $\{k_1, ..., k_n\}$

Attention computes $d(q, k_i)$ for i from 1 to n which can be slow

Faster *approximate* uses locality sensitive hashing (LSH)

Locality sensitive: if q is close to k_i:

$hash(q) == hash(k_i)$



This helps you group similar query and key vectors together,

plearning.ai

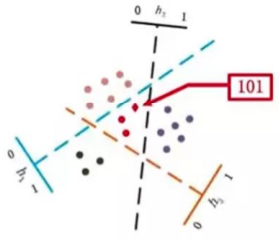
H Attention

andard Attention:

$$\mathrm{A}(Q, K, V) = \mathrm{softmax}(QK^T)V$$

H Attention:

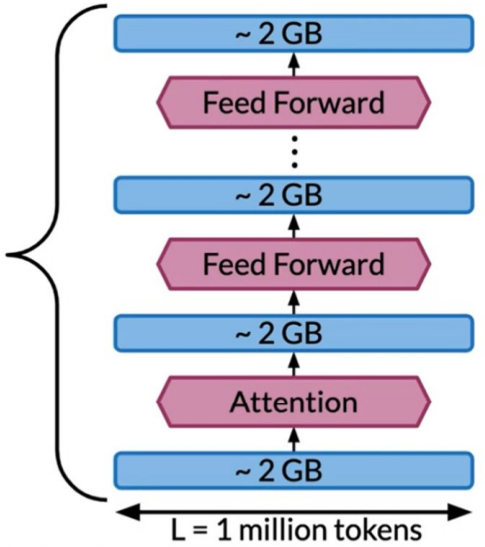
- Hash Q and K
- Standard attention within same-hash bins
- Repeat a few times to increase probability of key in the same bin



to increase the probability of finding Q and K in the same bin.

emory Efficiency

12 x Attention
12 x Feed-Forward



The size of the inputs and outputs for each layer will also be 2GB.

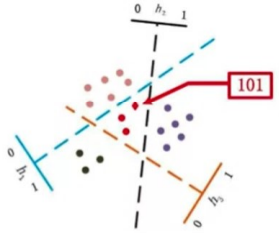
H Attention

andard Attention:

$$\mathrm{A}(Q, K, V) = \mathrm{softmax}(QK^T)V$$

H Attention:

- Hash Q and K
- Standard attention within same-hash bins
- Repeat a few times to increase probability of key in the same bin



You can repeat this process multiple times

H Attention

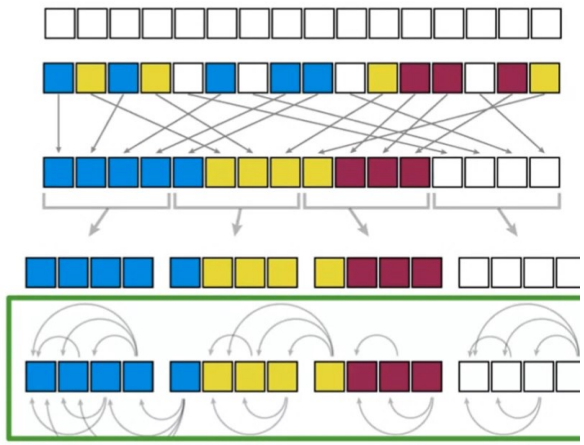
Sequence of Queries = Keys

LSH bucketing

Sort by LSH bucket

Chunk sorted sequence to parallelize

Attend within same bucket of wn chunk and previous chunk



And the adjacent chunks discovers the case with a hash bucket that is split over

versible layers equations

andard Transformer:

$y_a = x + \text{Attention}(x)$ $\quad y_b = y_a + \text{FeedFwd}(y_a)$

eversible:

$y_1 = x_1 + \text{Attention}(x_2)$ $\quad y_2 = x_2 + \text{FeedFwd}(y_1)$

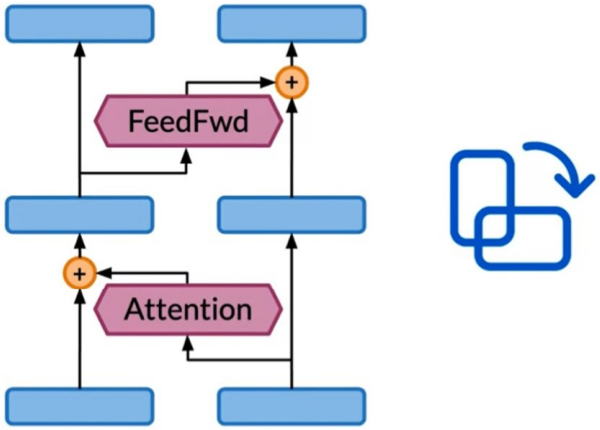
compute x_1, x_2 from y_1, y_2:

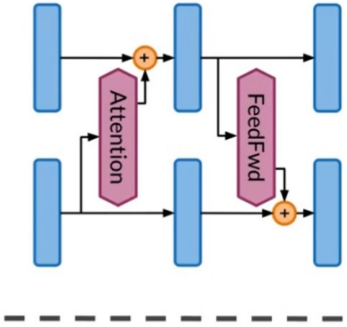
$x_1 = y_1 - \text{Attention}(x_2)$ $\quad x_2 = y_2 - \text{FeedFwd}(y_1)$

x_1 equals Y_1 minus attention of x_2

plearning.ai

versible layers equations

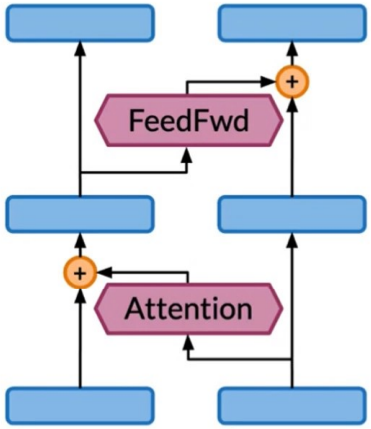


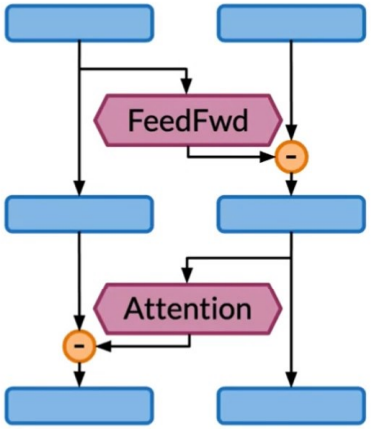


something like this, so that information

plearning.ai

versible layers

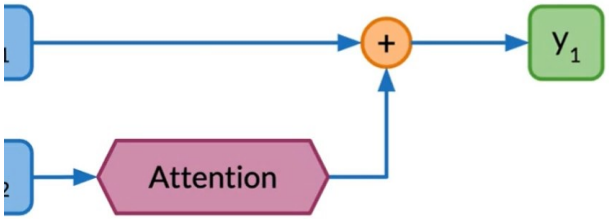




about caching for the backwards pass.

plearning.ai

versible layers equations



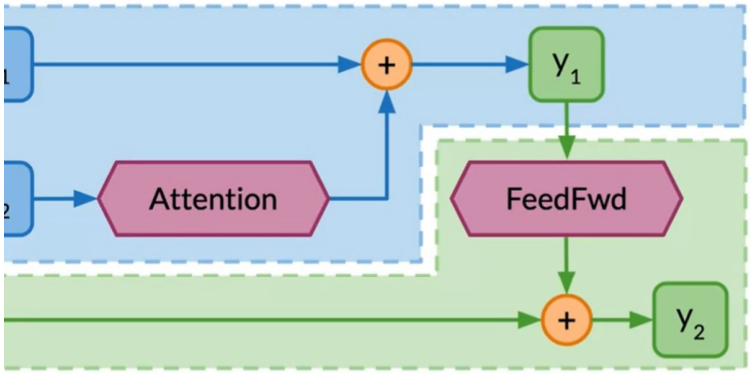
$y_1 = x_1 + \text{Attenti}$

$y_2 = x_2 + \text{FeedFv}$

Restating the equations I showed you earlier,

plearning.ai

versible layers equations



$y_1 = x_1 + \text{Attenti}$

$y_2 = x_2 + \text{FeedFv}$

$x_2 = y_2 - \text{FeedFw}$

$x_1 = y_1 - \text{Attenti}$

going to calculate
x_2 before x_1,

plearning.ai

former

e Reversible Transformer

1 GPU
(16 GB)

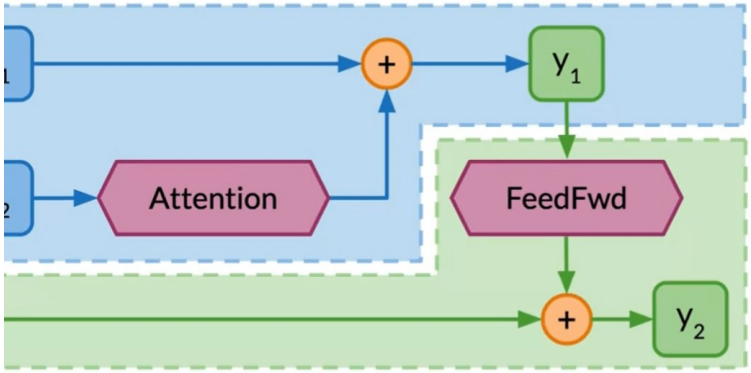
L = 1 million tokens

That's enough to
fit an entire book,

0:24

plearning.ai

versible layers equations



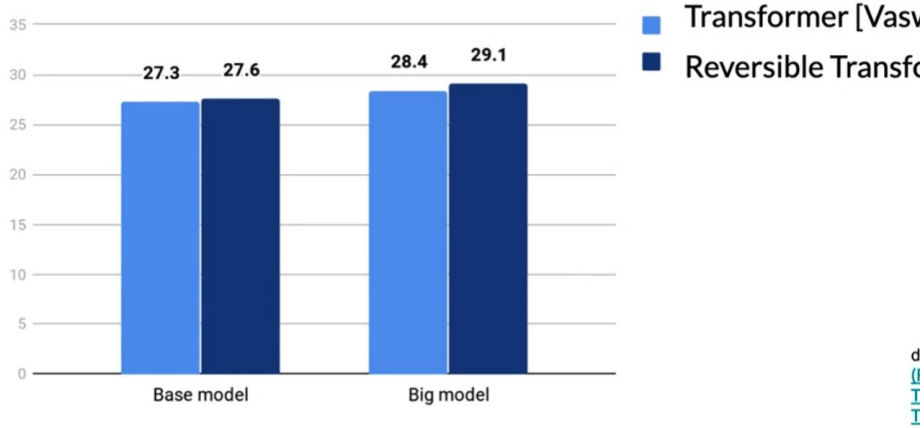
$y_1 = x_1 + \text{Attenti}$

$y_2 = x_2 + \text{FeedFv}$

That's a forward pass for
irreversible residual block.

plearning.ai

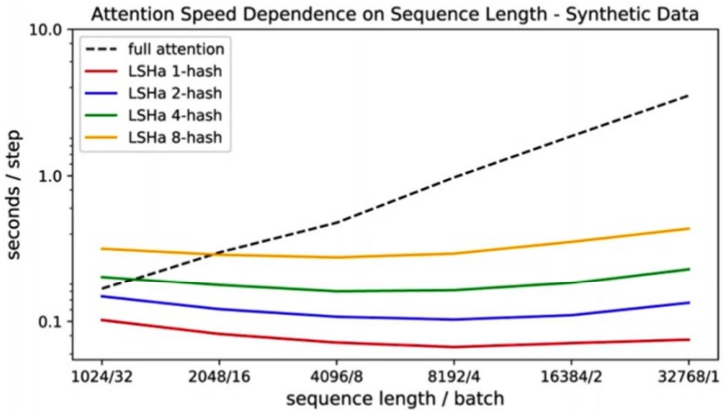
versible Transformer: BLEU Scores



this can be applied anywhere
you use a transformer model.

plearning.ai

former



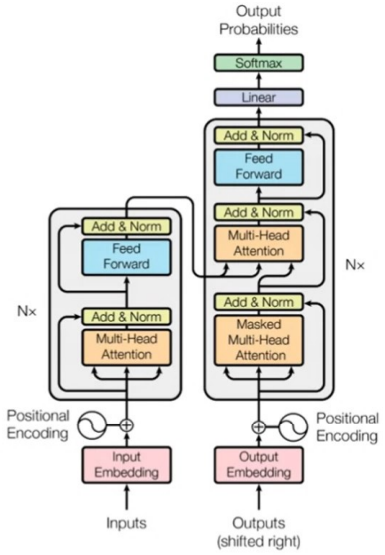
longer than fewer hashes,

plearning.ai

former

LSH Attention

Reversible Layers



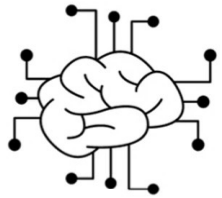
to more efficiently use
the memory available.

plearning.ai

tline

Neural networks and forward propagation

Structure for sentiment analysis



plearning.ai