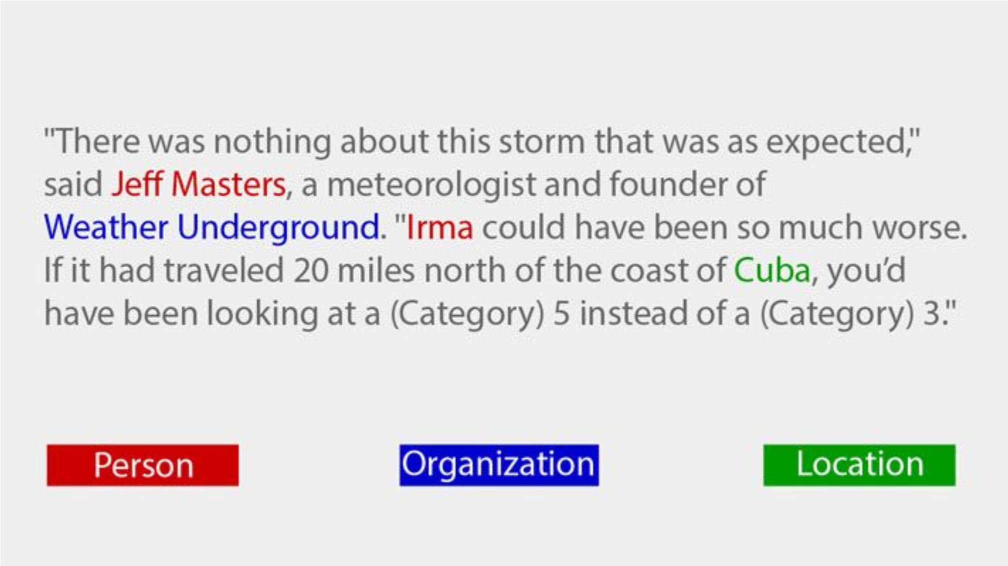
Natural Language Processing Workshop

Amy Hemmeter, MSA Class of ‘18
Machine Learning Engineer, Quicken Loans

Sequence Tagging

- Where you provide labels for a series of tokens
- Examples: part-of-speech tagging, named entity recognition



What does the data look like?

- CONLL format, BIO labels

What does the data look like?

- CONLL format, BIO labels

We ‘re going on vacation to San Diego

What does the data look like?

- CONLL format, BIO labels

We ‘re going on vacation to San-B-LOC Diego

What does the data look like?

- CONLL format, BIO labels

We ‘re going on vacation to
San-B-LOC Diego-I-LOC

What does the data look like?

- CONLL format, BIO labels

We-O ‘re-O going-O on-O vacation-O to-O San-B-LOC Diego-I-LOC

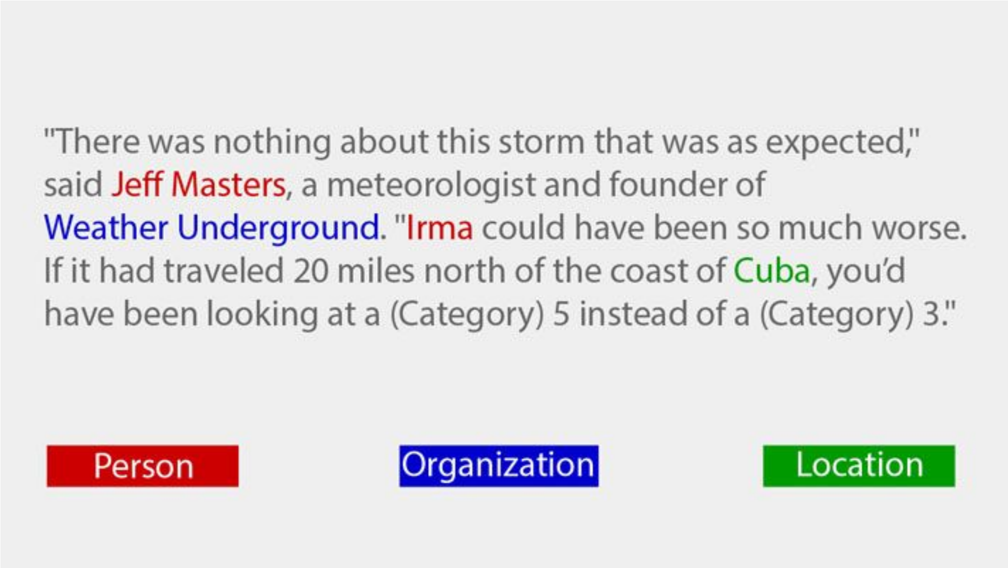
Quick Quiz

"There was nothing about this storm that was as expected," said Jeff Masters, a meteorologist and founder of Weather Underground. "Irma could have been so much worse. If it had traveled 20 miles north of the coast of Cuba, you'd have been looking at a (Category) 5 instead of a (Category) 3."

Person Organization Location

Quick Quiz

Jeff



Quick Quiz

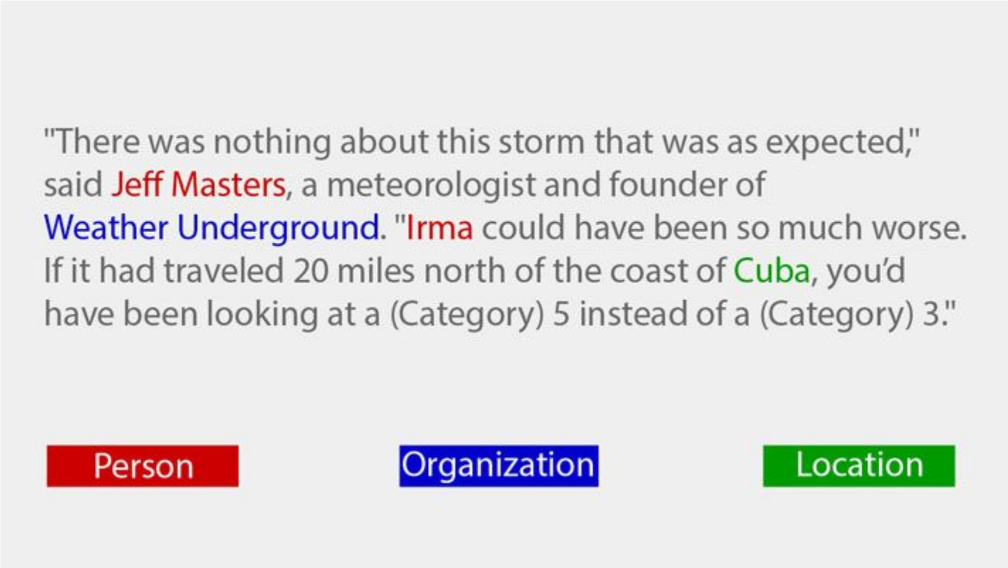
Underground

"There was nothing about this storm that was as expected," said Jeff Masters, a meteorologist and founder of Weather Underground. "Irma could have been so much worse. If it had traveled 20 miles north of the coast of Cuba, you'd have been looking at a (Category) 5 instead of a (Category) 3."

Person | Organization | Location

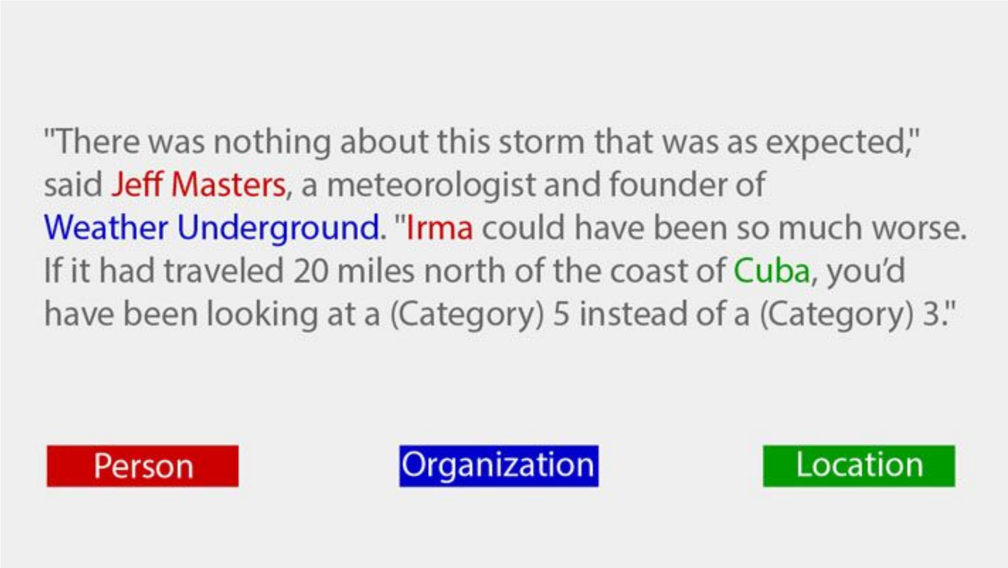
Quick Quiz

Cuba

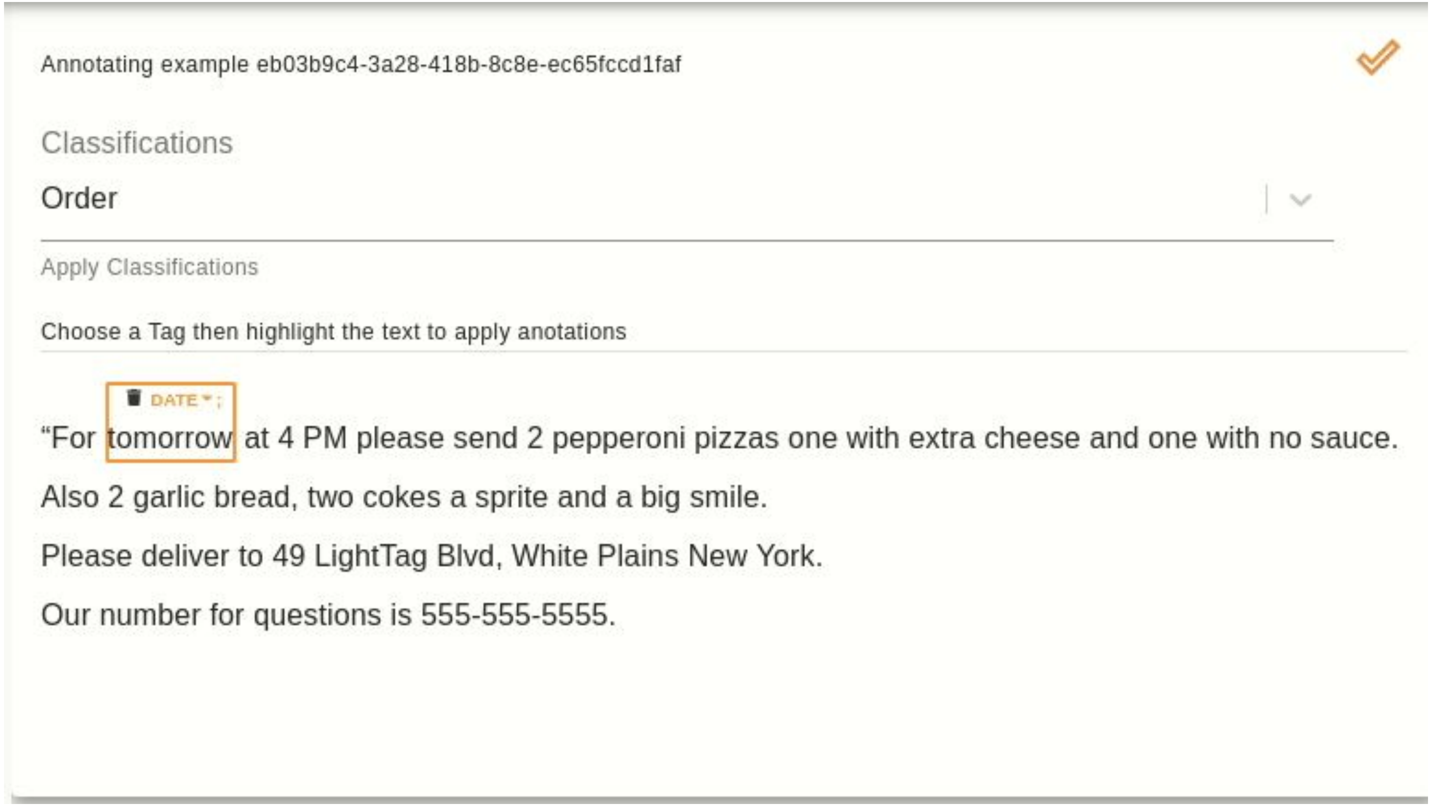


Quick Quiz

storm



In practice...



Annotating example eb03b9c4-3a28-418b-8c8e-ec65fccd1faf
Classifications
Order
Apply Classifications
Choose a Tag then highlight the text to apply anotations
DATE
“For tomorrow at 4 PM please send 2 pepperoni pizzas one with extra cheese and one with no sauce.
Also 2 garlic bread, two cokes a sprite and a big smile.
Please deliver to 49 LightTag Blvd, White Plains New York.
Our number for questions is 555-555-5555.

Recall Language Modeling

$$P(\text{“the”} \mid \text{“I went to”})$$

Recall Language Modeling

“I-O went-O to-O the-?…”

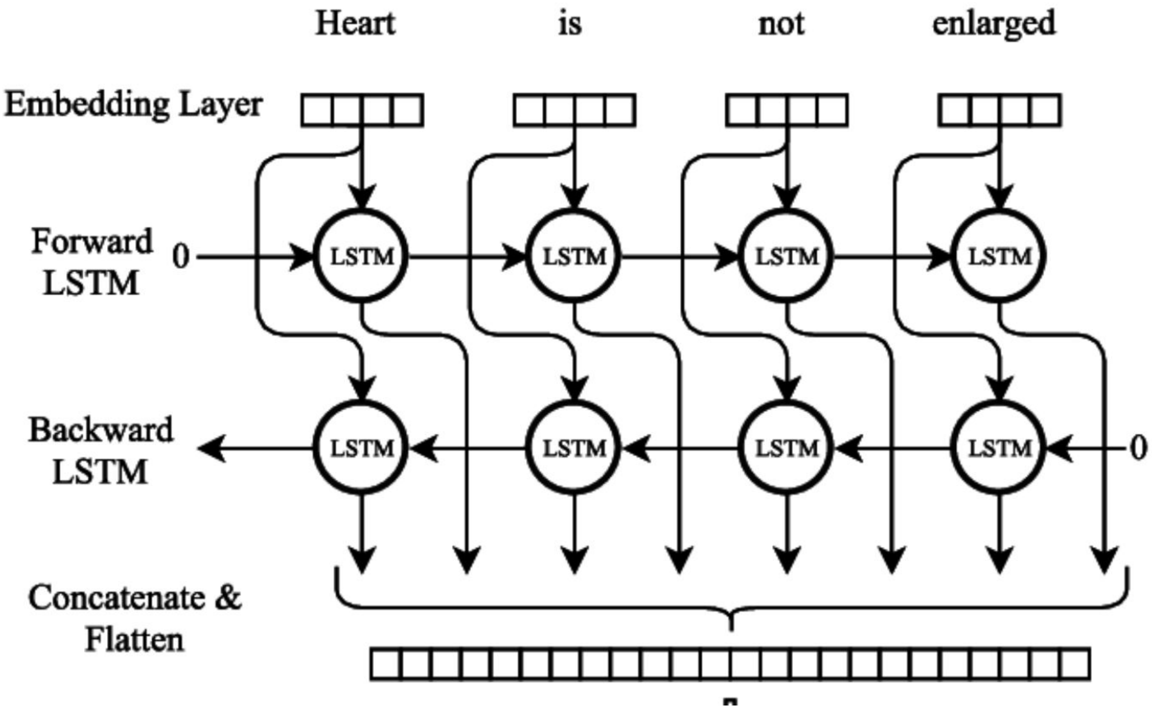
Recall Language Modeling

"I went to **the Ohio State University**"

Recall Language Modeling

"I went to the **University of Michigan**"

Tagger Architecture - biLSTM



Source: [Cornegruta et al.]

Tagger Output

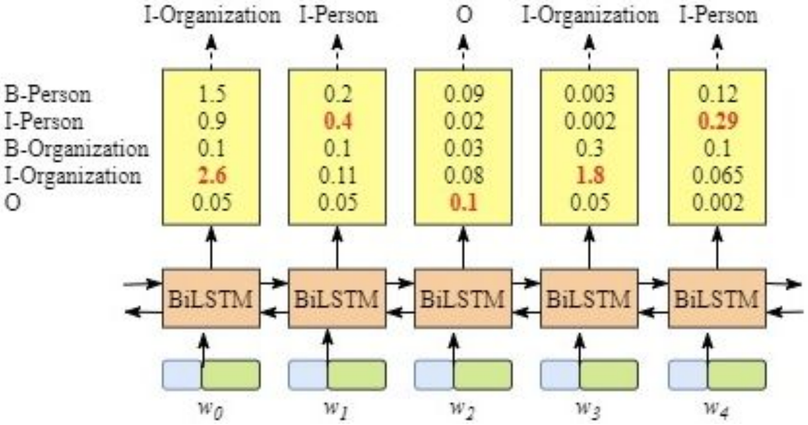
Real labels:

“I-O went-O to-O the-O University-B-INST of-I-INST Michigan-I-INST”

Model output:

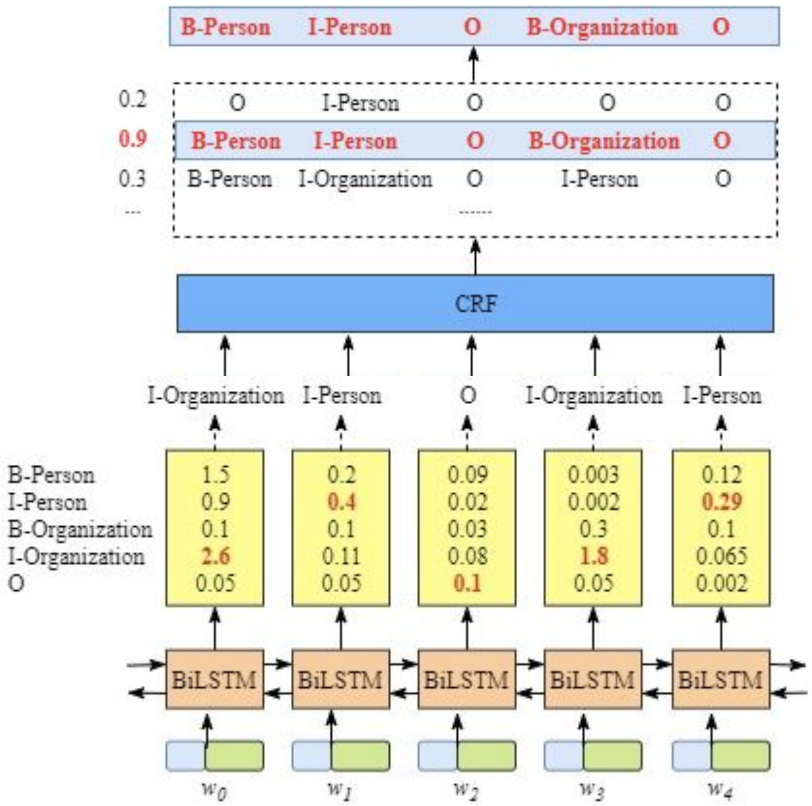
“I-O went-O to-O the-O University-I-INST of-I-INST Michigan-B-INST

Conditional Random Field



Source: [CreateMomo]

Conditional Random Field



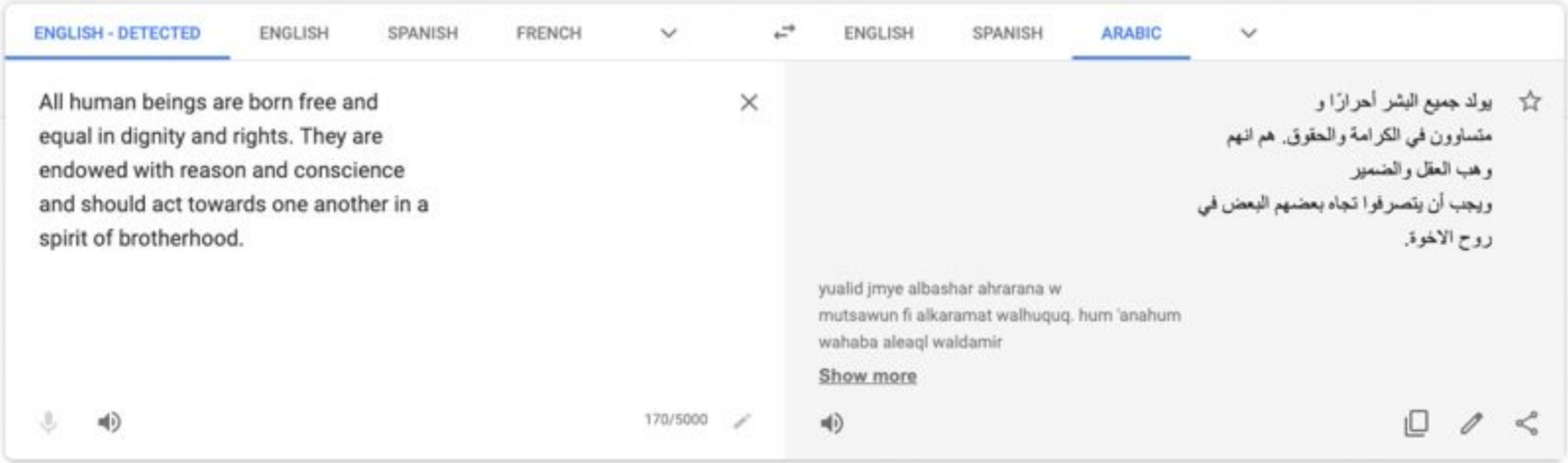
Source: [CreateMomo]

Evaluating a Tagger

- Accuracy
  - (TP + TN) / TP + TN + FP + FN
- Precision
  - TP/(TP + FP)
- Recall
  - TP/(TP + FN)
- F1-score
  - F1 = (2 * precision * recall) / precision + recall

Machine Translation

Machine Translation



1950s Machine Translation

Neural Machine Translation

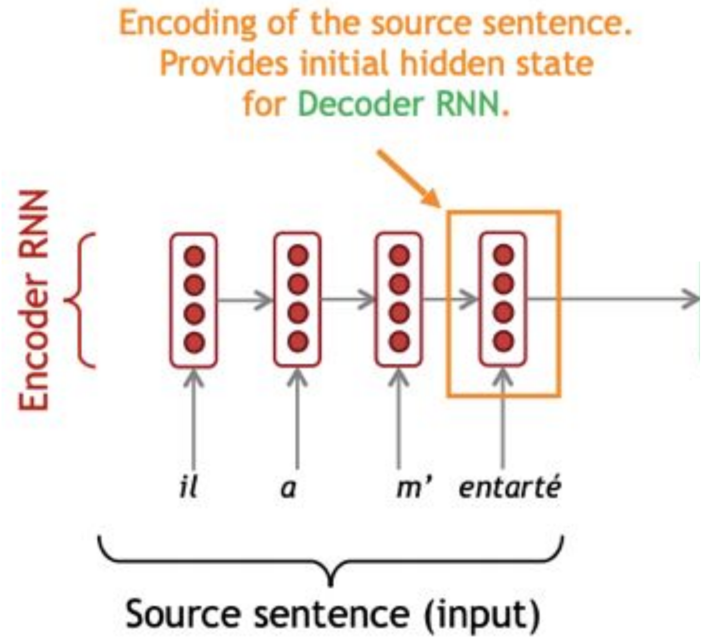
- MT with a single neural network
- The architecture is called a *sequence-to-sequence* (*seq2seq*) model or an *encoder-decoder* model

The sequence-to-sequence model

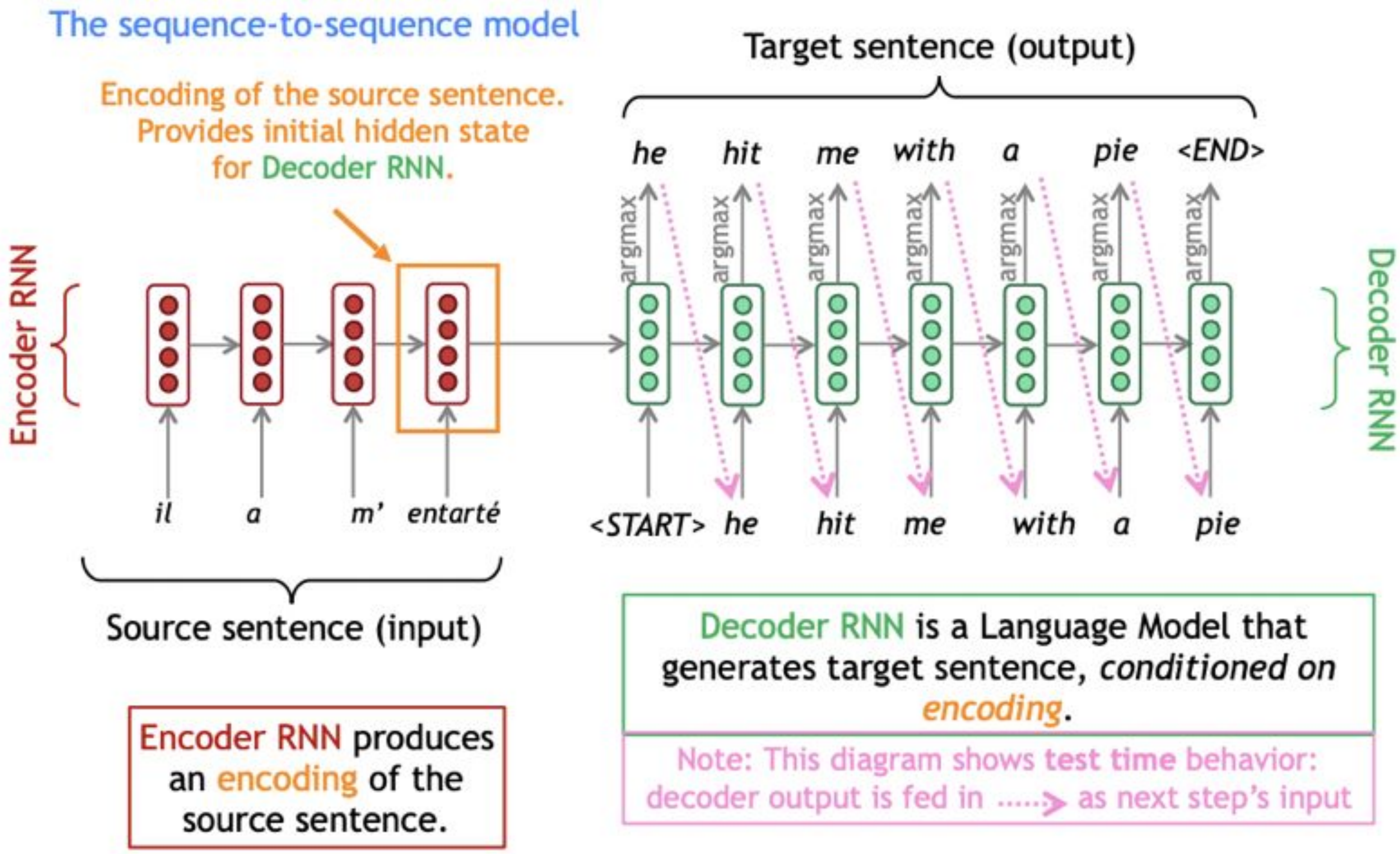
il a m’ entarté

Source sentence (input)

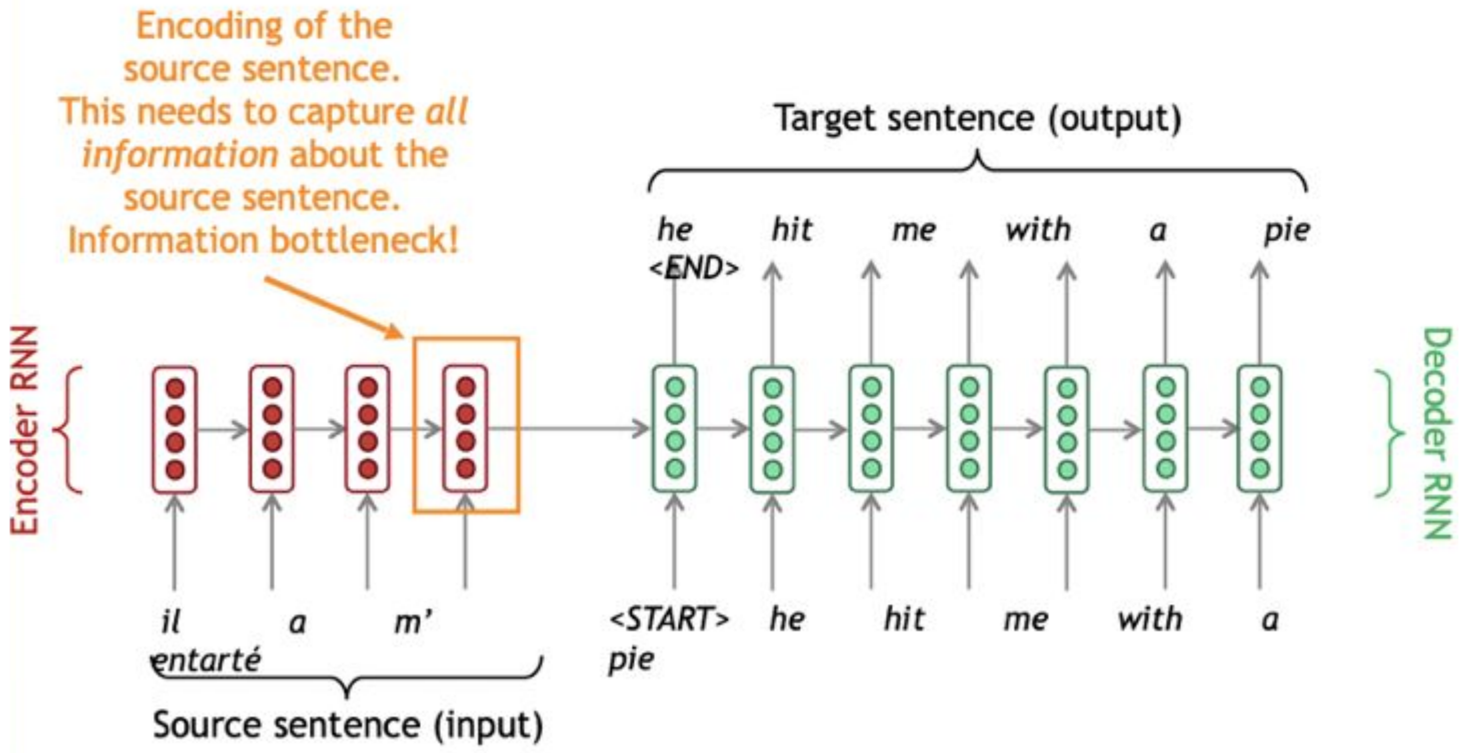
The sequence-to-sequence model



Encoder RNN produces an encoding of the source sentence.

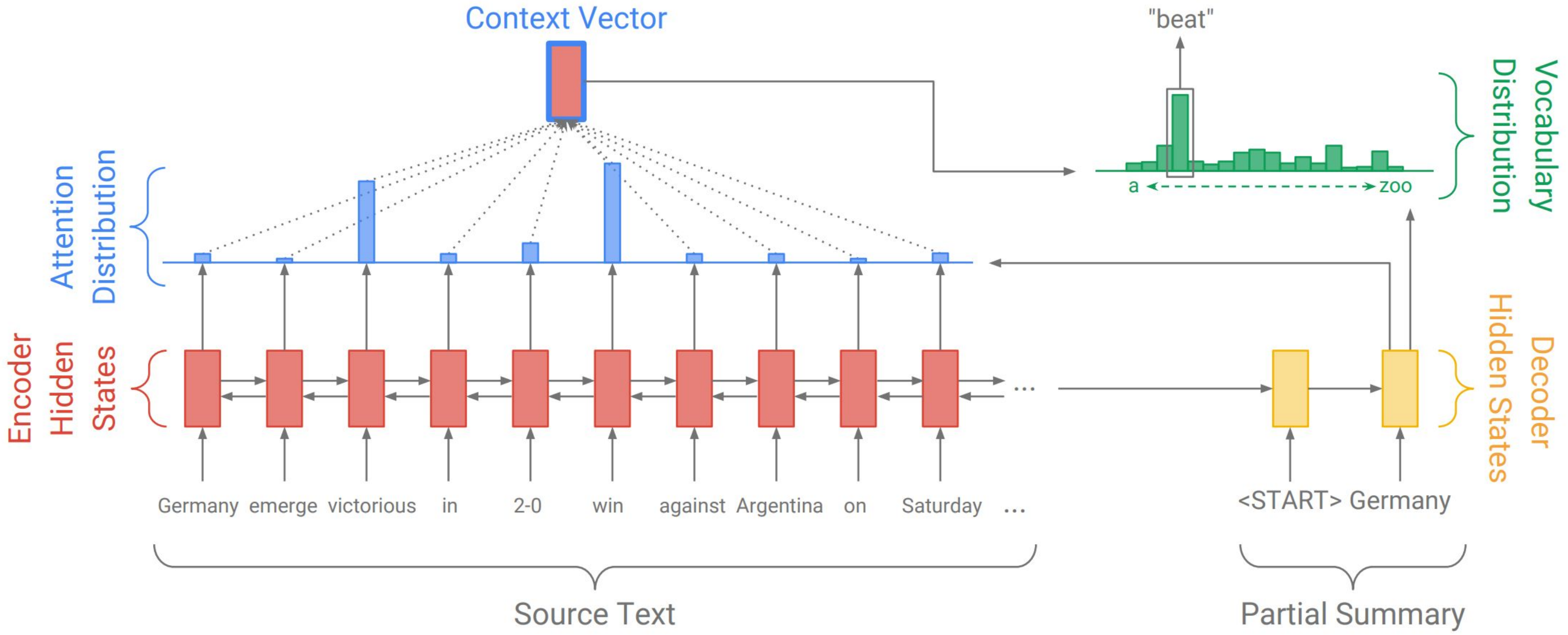

The sequence-to-sequence model
Encoding of the source sentence.
Provides initial hidden state
for Decoder RNN.
Target sentence (output)
he hit me with a pie <END>
argmax
Encoder RNN
Decoder RNN
il a m' entarté
<START> he hit me with a pie
Source sentence (input)
Encoder RNN produces an encoding of the source sentence.
Decoder RNN is a Language Model that generates target sentence, *conditioned on* encoding.
Note: This diagram shows **test time** behavior: decoder output is fed in as next step's input

Sequence-to-sequence: the bottleneck problem



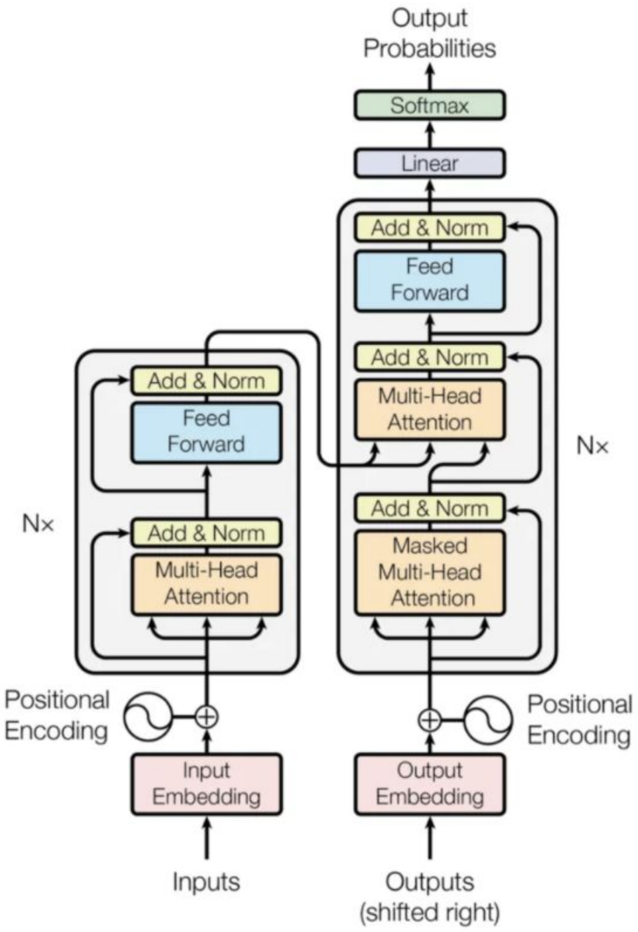
Attention

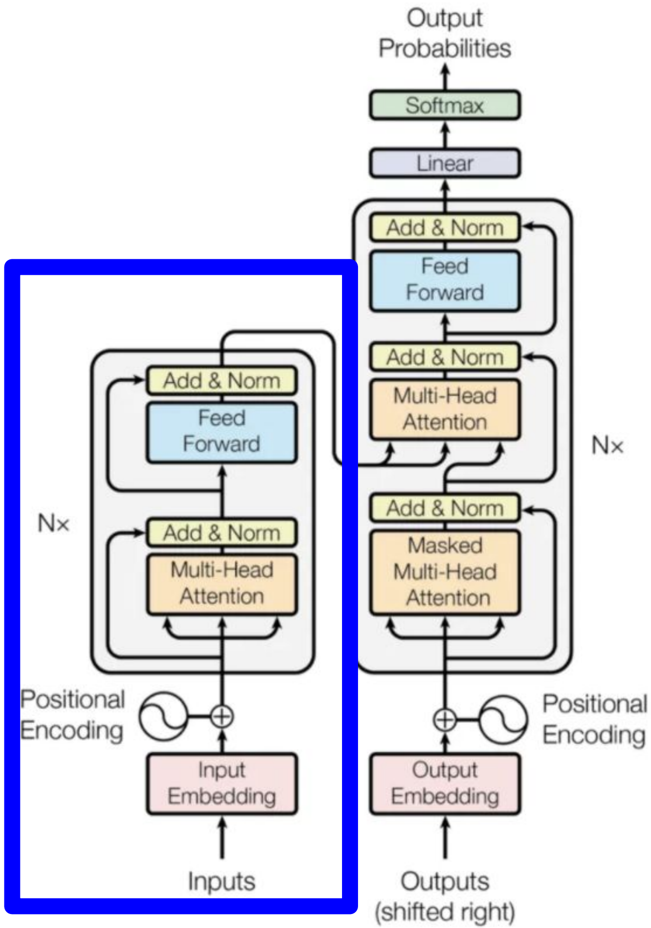
- Attention solves the bottleneck problem
- It includes a context matrix that includes information about a word’s importance in the sentence

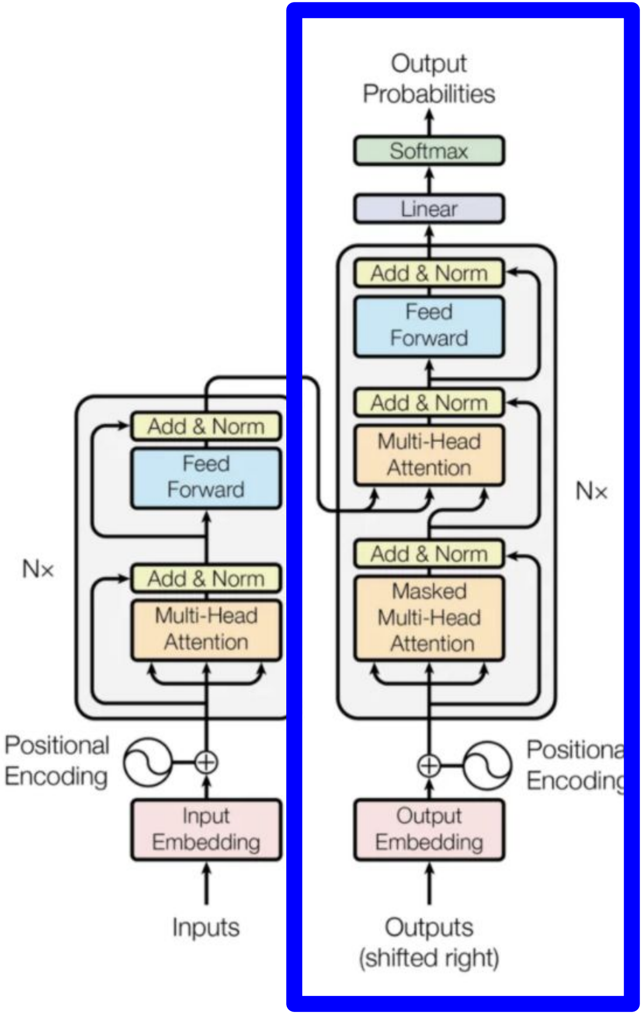


Transformers

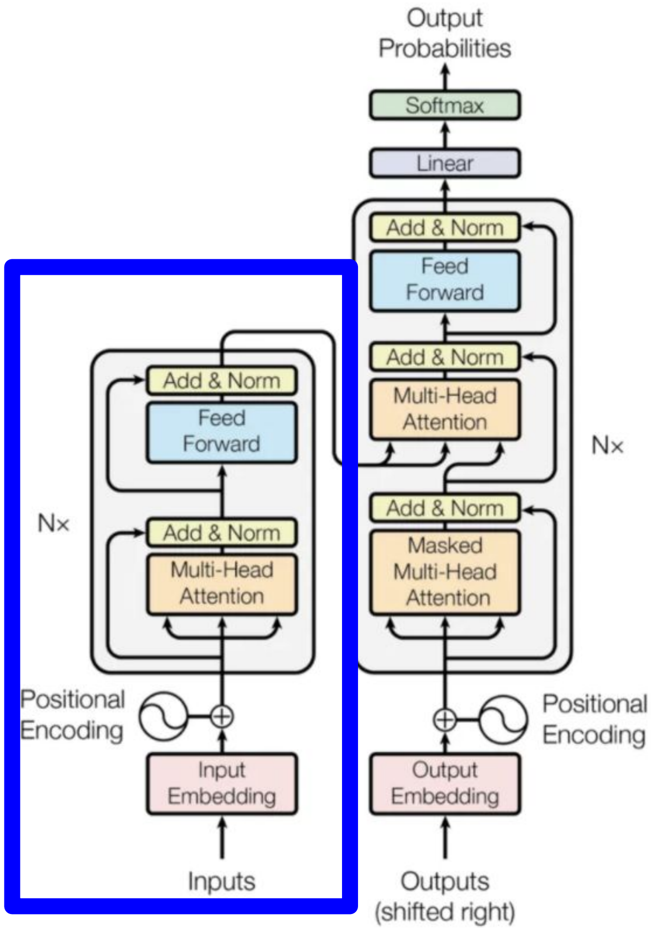
- Previously, RNNs had to be processed sequentially
- Transformers are able to maintain some contextual information while also being able to be processed in parallel


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

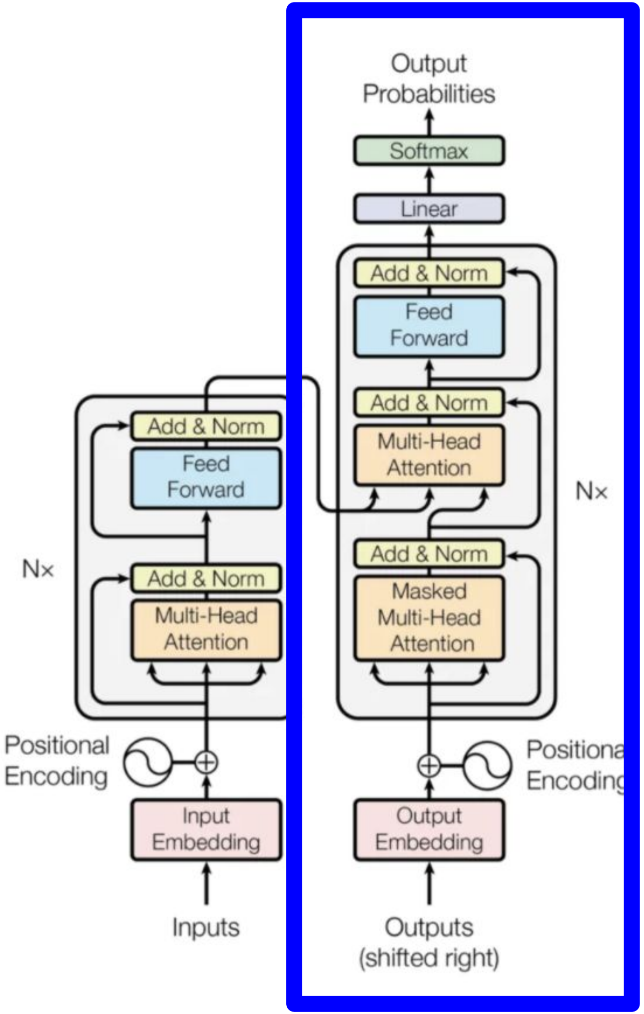

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)



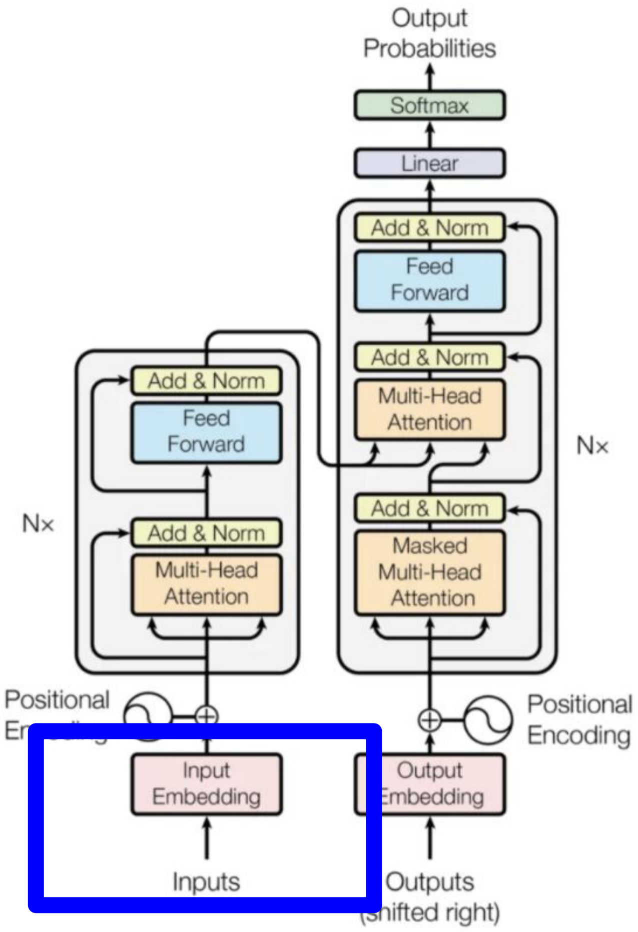
Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

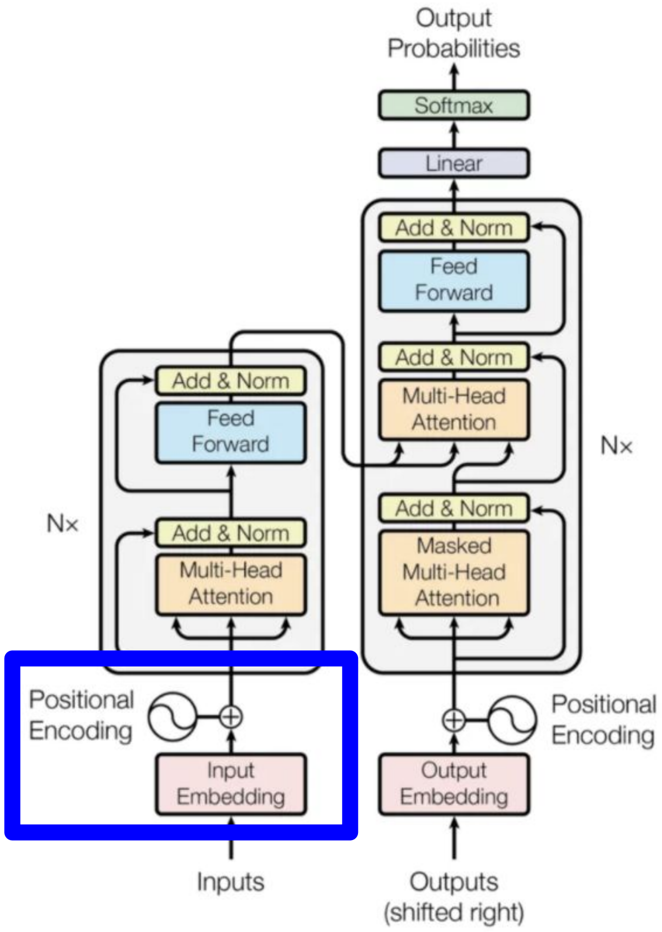


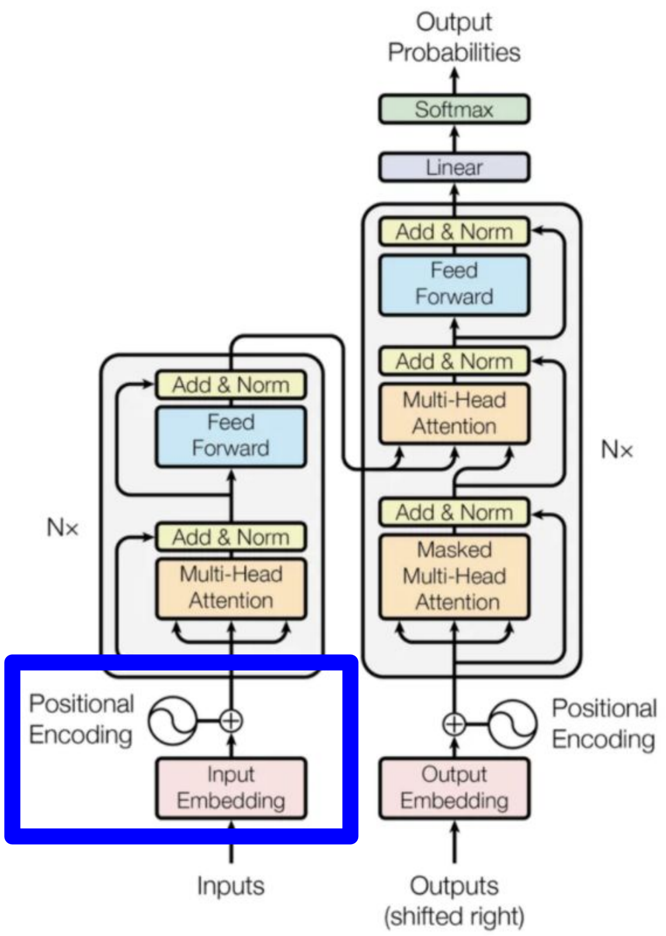
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
Outputs (shifted right)
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs

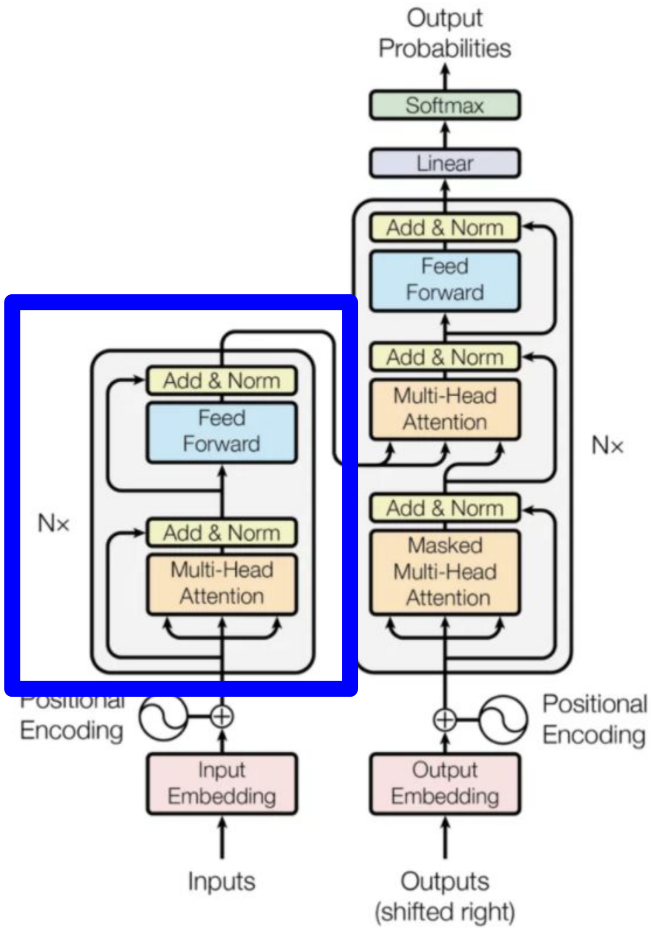


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Positiona
Encoding
Output
Embedding
Outputs
(shifted right)
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Input
Embedding
Inputs

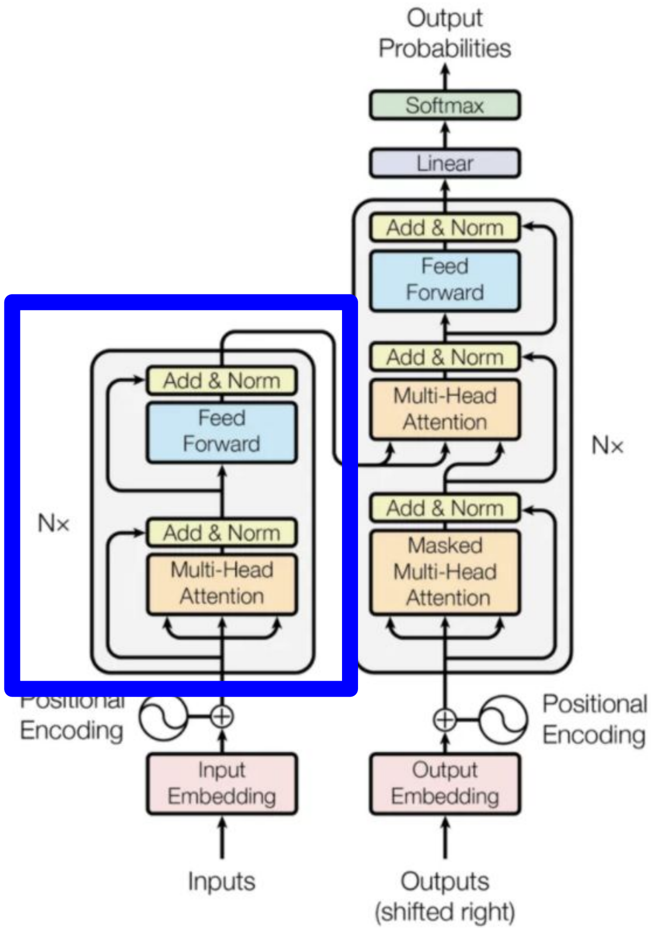

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Output
Embedding
Positional
Encoding
Inputs
Outputs
(shifted right)

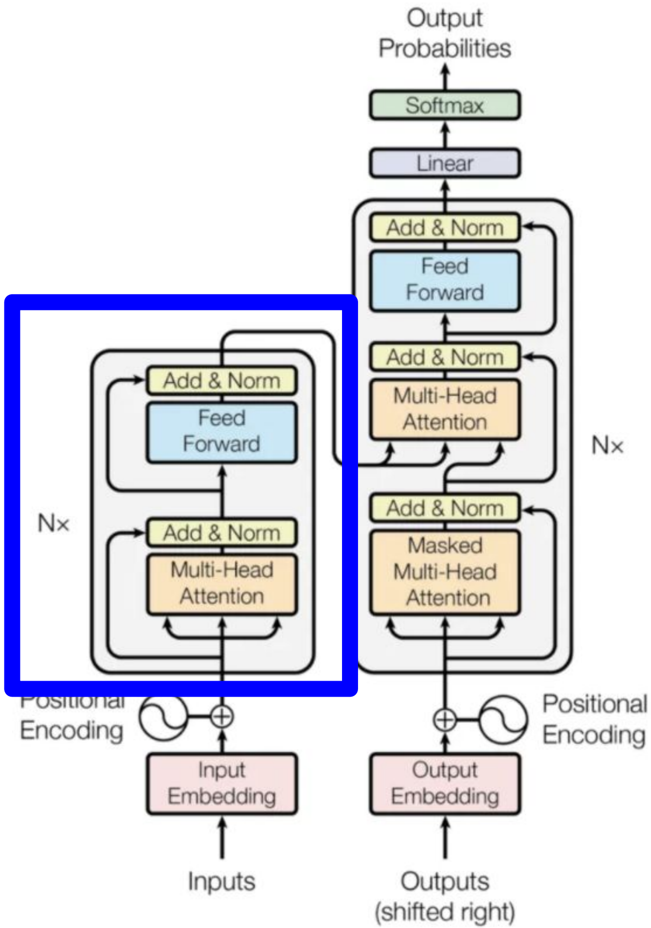

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Output Embedding
Positional Encoding
Inputs
Outputs (shifted right)

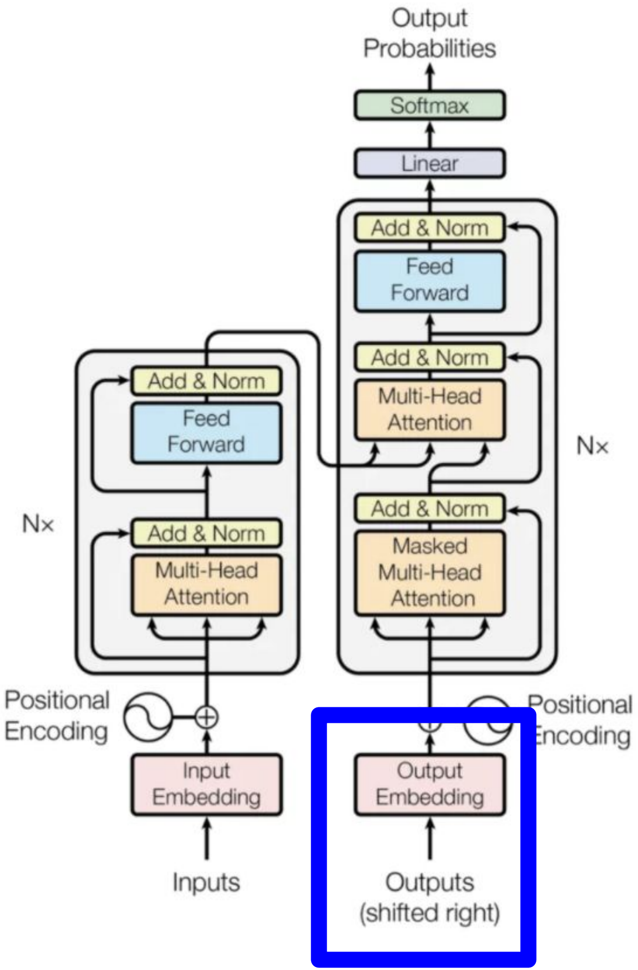


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

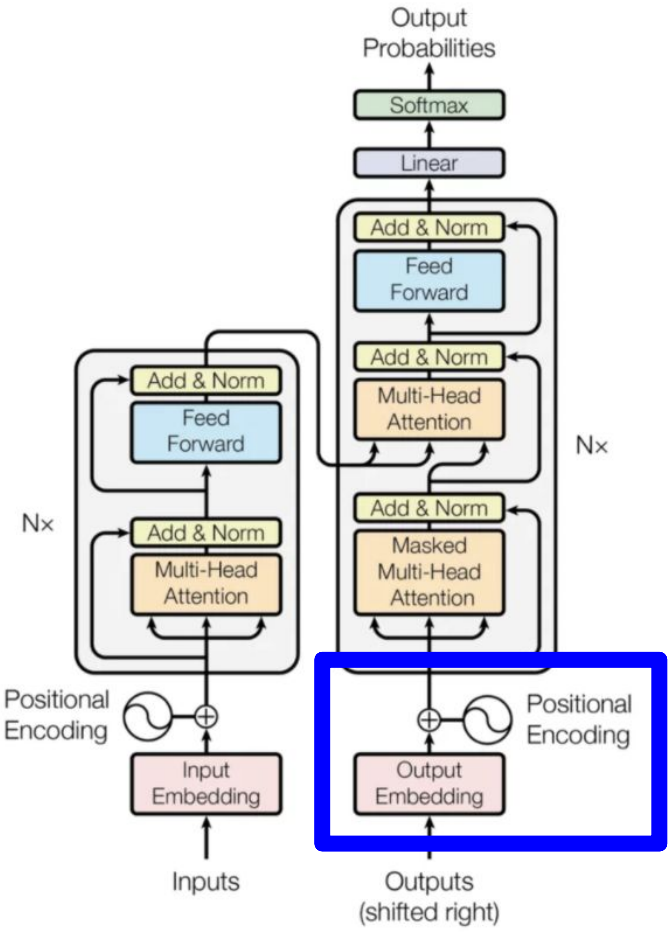


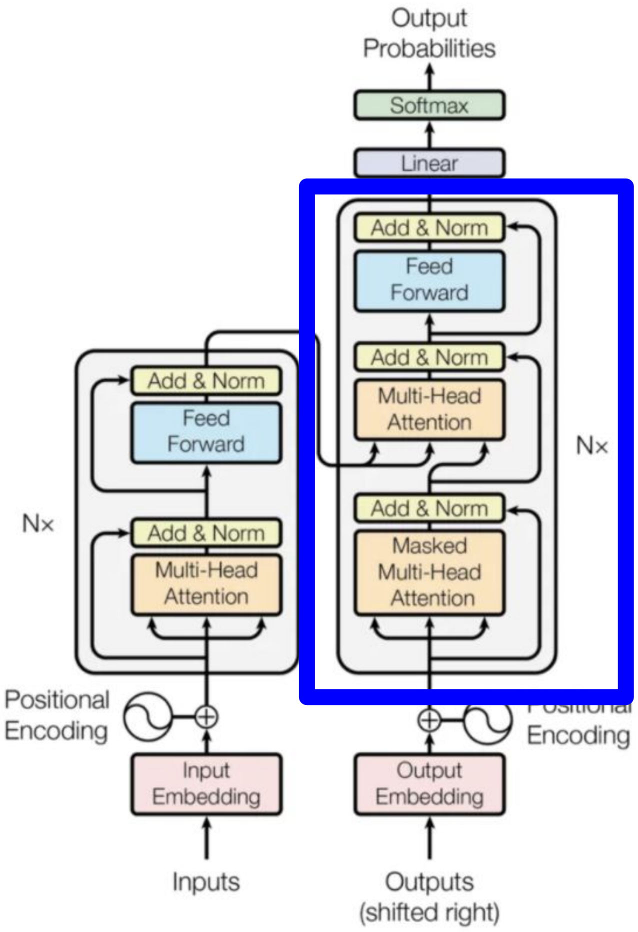
Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Input
Embedding
Positional
Encoding
Output
Embedding
Inputs
Outputs
(shifted right)

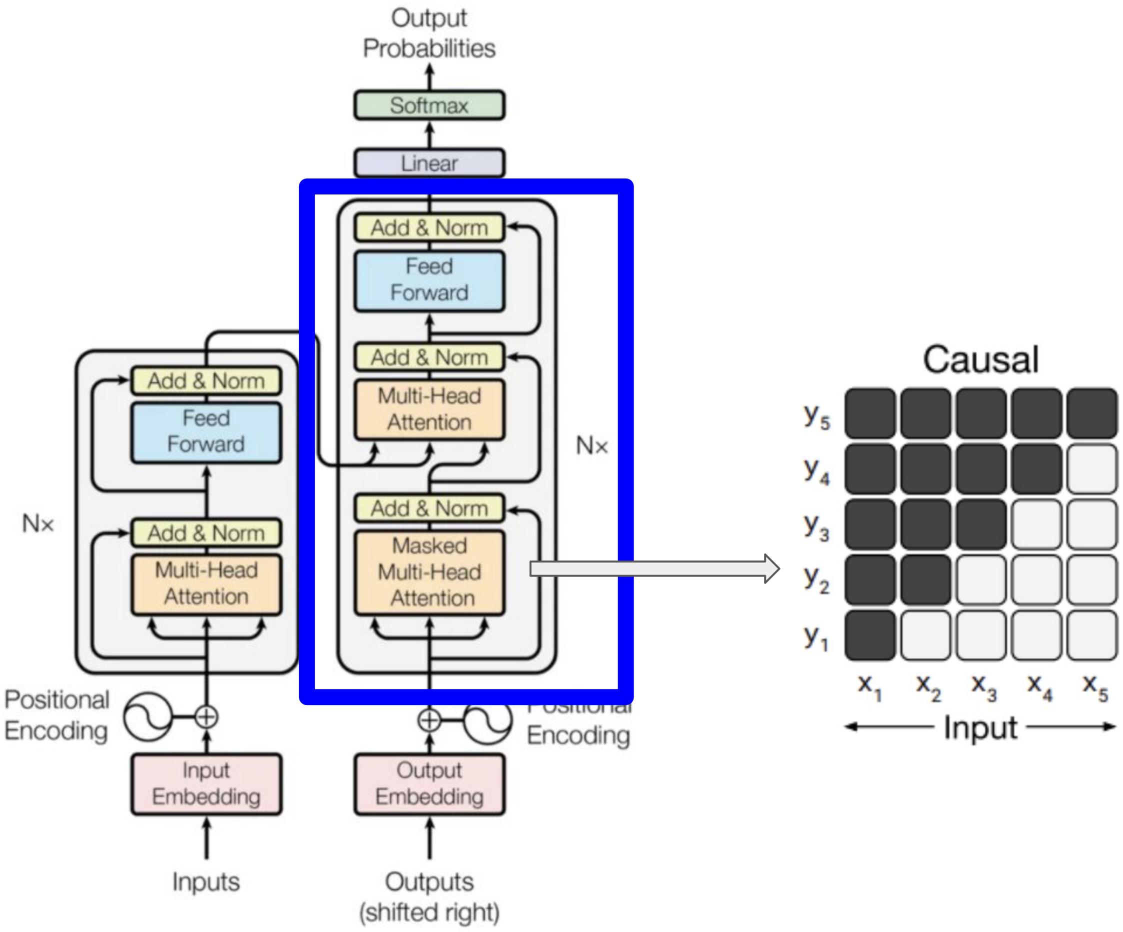

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

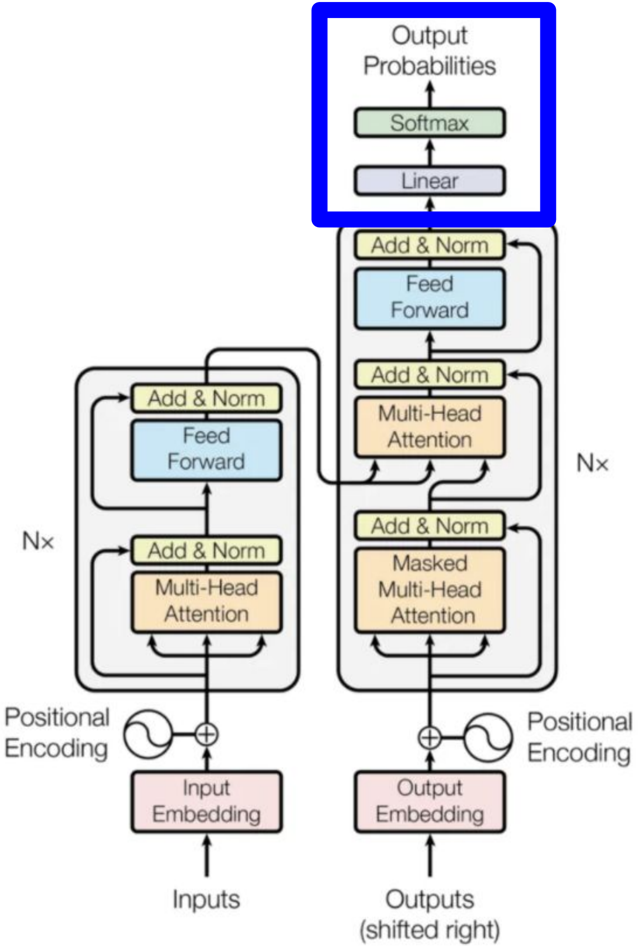


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs
(shifted right)


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
Positional Encoding
Output Embedding
Outputs (shifted right)


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Causal
y5
y4
y3
y2
y1
x1 x2 x3 x4 x5
Input


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

Applications of Transformers

- Machine Translation
- Text Summarization
- Language modeling
- Contextual word embeddings

Contextual Embeddings

Issues with Word Vectors

“Did I show you this **clip** of a dog skateboarding?”

“I need to get a chip **clip**”

“He runs at a good **clip**”

“I have to **clip** my dog’s nails”

Issues with Word Vectors

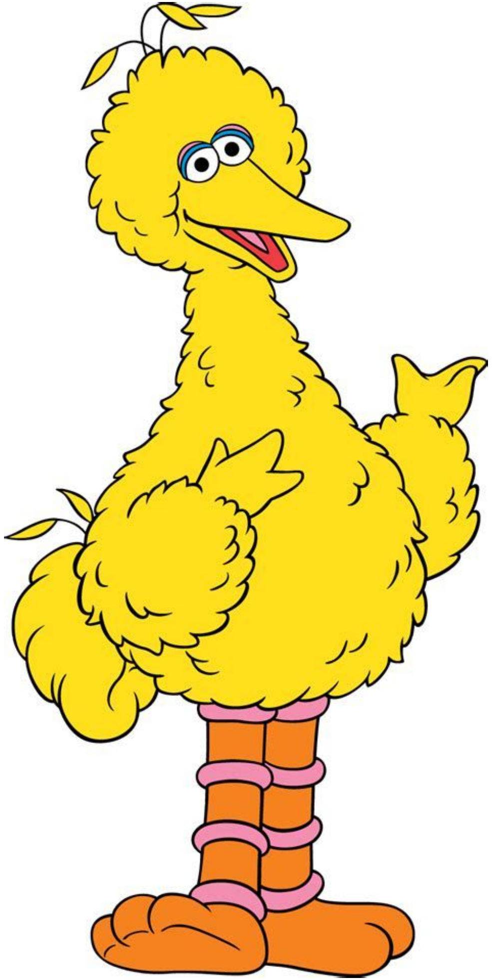
“Did I show you this **clip** of a dog skateboarding?”

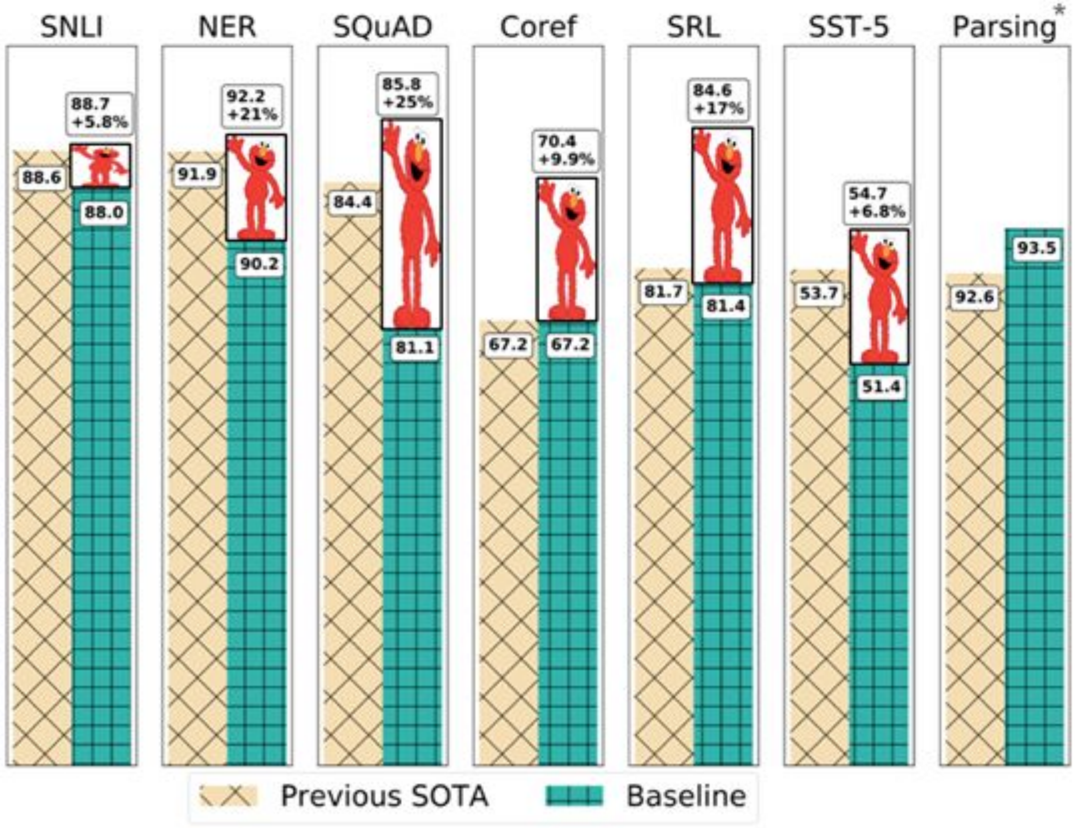
“I need to get a chip **clip**”

“He runs at a good **clip**”

“I have to **clip** my dog’s nails”

Contextual Word Embeddings





*Kitaev and Klein, ACL 2018 (see also Joshi et al., ACL 2018)

Preprocessing in NLP

Reasons NLP is hard

Reasons NLP is hard

1. Words are not numbers

Reasons NLP is hard

1. Words are not numbers

2. Input can be different lengths

Reasons NLP is hard

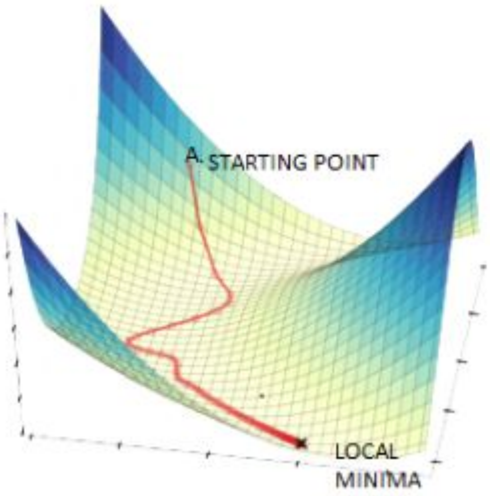
1. Words are not numbers

2. **Input can be different lengths***

*https://github.com/blester125/A2D-NLP-Talk-Feb-27-2020

Gradient Descent Review (I'm sure of it this time!)

- How to find the minimum of your loss function
- Uses the derivative of the loss function with respect to your parameters to take a step in a direction towards the minimum
- How most machine learning algorithms work
- The rate at which you move down this slope is called the learning rate

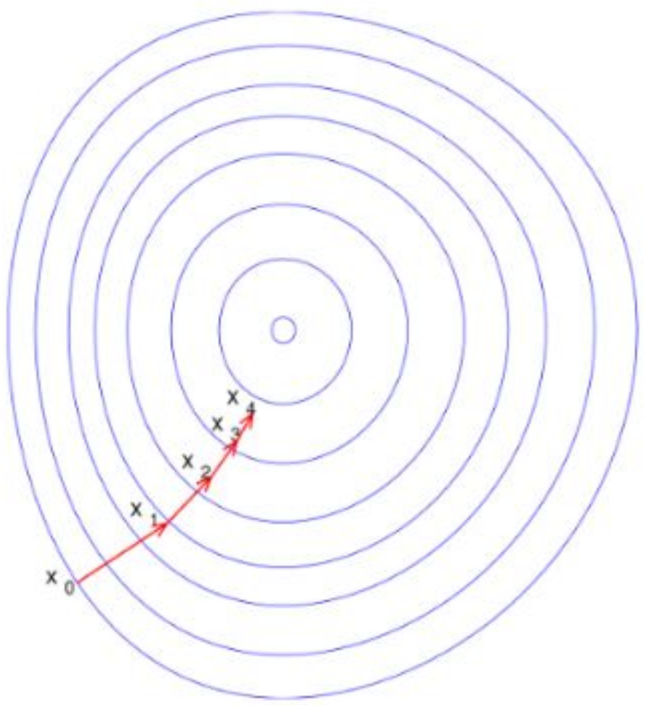


Picture from Data Science Central,
https://www.datasciencecentral.com/profiles/blogs/alternatives-to-the-gradient-descent-algorithm

Gradient Descent

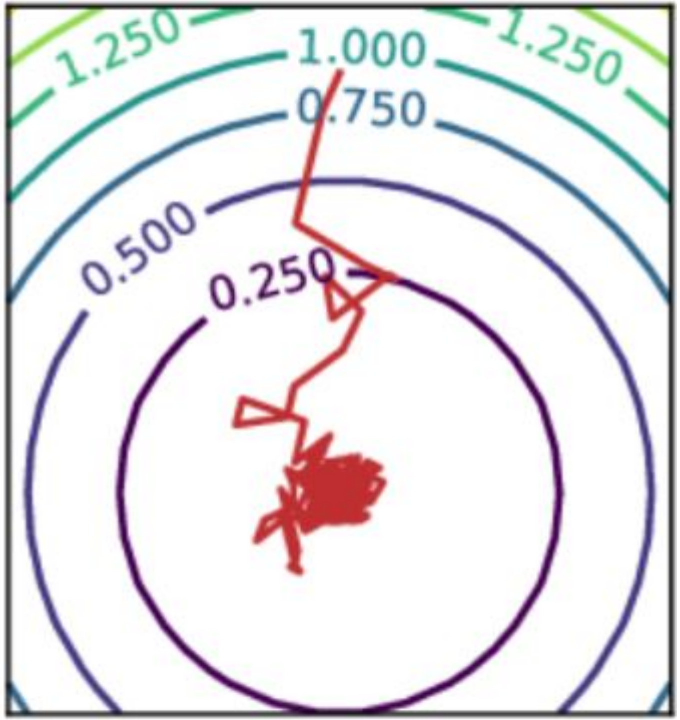
- Three ways of running gradient descent on your training data:
  - Full gradient descent
  - Stochastic gradient descent
  - Batched gradient descent

Full Gradient Descent



- This process gives us the “true gradient”
- Calculate the loss for each example in the training set and then use all of these losses to calculate the gradient
  - You need to run each example before you update any parameters
  - Very slow and compute-heavy!!

Stochastic Gradient Descent



Pros:

- You update your parameters after each step
- Much faster
- No need for any of the techniques we're about to talk about

Cons

- Your gradient could be wrong
- What works well for one example could hurt another example

Batched Gradient Descent

- Batching provides a happy medium
- You get to update your parameters more often
- You can get a better approximation of your gradient
- Minibatch size is now a hyperparameter for your deep learning model
- And now you need to learn some tricks….

Binary Logistic Regression

- We have two vectors of the same size representing the weights and the features respectively
- We take the sum of the features weighted by the weight to create a logit score
- In this example we’re going to ignore the activation function for now

Binary Logistic Regression

$$f = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$
$$w = \begin{bmatrix} 5 & 6 & 7 & 8 \end{bmatrix}$$
$$s =$$

Binary Logistic Regression

$$f = [\ 1 \quad 2 \quad 3 \quad 4\ ]$$
$$w = [\ 5 \quad 6 \quad 7 \quad 8\ ]$$
$$s = 5$$

Binary Logistic Regression

$$f = [\quad 1 \quad 2 \quad 3 \quad 4 \quad]$$
$$w = [\quad 5 \quad 6 \quad 7 \quad 8 \quad]$$
$$s = 17$$

Binary Logistic Regression

$$f = [\ \boxed{1\ \ 2\ \ 3\ \ 4}\ ]$$
$$w = [\ \boxed{5\ \ 6\ \ 7\ \ 8}\ ]$$
$$s = 70$$

Multi-Class Logistic Regression

$$f = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} \quad \end{bmatrix}$$

Multi-Class Logistic Regression

$$f = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} 5 & \end{bmatrix}$$

Multi-Class Logistic Regression

$$f = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} 17 & \end{bmatrix}$$

Multi-Class Logistic Regression

$$f = \begin{bmatrix} \boxed{1 \quad 2 \quad 3 \quad 4} \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} \boxed{70} & \end{bmatrix}$$

Multi-Class Logistic Regression

$$f = \begin{bmatrix} \boxed{1 \quad 2 \quad 3 \quad 4} \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} 70 & \boxed{110} \end{bmatrix}$$

Batched Multi-Class Logistic Regression

$$F = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} \quad \end{bmatrix}$$

Batched Multi-Class Logistic Regression

$$F = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} 70 & \\ & \end{bmatrix}$$

Batched Multi-Class Logistic Regression

$$F = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} 70 & 110 \\ & \end{bmatrix}$$

Batched Multi-Class Logistic Regression

$$F = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} 70 & 110 \\ 382 & \end{bmatrix}$$

Batched Multi-Class Logistic Regression

$$F = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$W = \begin{bmatrix} 5 & 9 \\ 6 & 10 \\ 7 & 11 \\ 8 & 12 \end{bmatrix}$$

$$s = \begin{bmatrix} 70 & 110 \\ 382 & 614 \end{bmatrix}$$

Why did I show you that math?

- To give you an idea for how we can vectorize the inputs for the data -- this is the source of the speed gains
- To help you visualize so that padding will make more sense to you
- To show you that in order for that math to work, your feature inputs need to be the same size

A Fly in the Ointment

Consider the following sentences:

[The dog ran very fast]

[The cat slept]

What is padding?

- Meaningless tokens we can insert into our batches for sentences that are not the same length

[The dog ran very fast]

[The cat slept <PAD> <PAD>]

“dog” : [1 2 4 3 5]

“dog” : [1 2 4 3 5]

“cat” : [1 3 4 3 5]

“dog” : [1 2 4 3 5]

“cat” : [1 3 4 3 5]

<PAD> : ?

“dog” : [1 2 4 3 5]

“cat” : [1 3 4 3 5]

<PAD> : [0 0 0 0 0]

Lengths vector

- Because zeros can still mess things up for us, we need to keep track of the lengths of our original input in a lengths vector.

[The dog ran very fast]

[The cat slept <PAD> <PAD>]

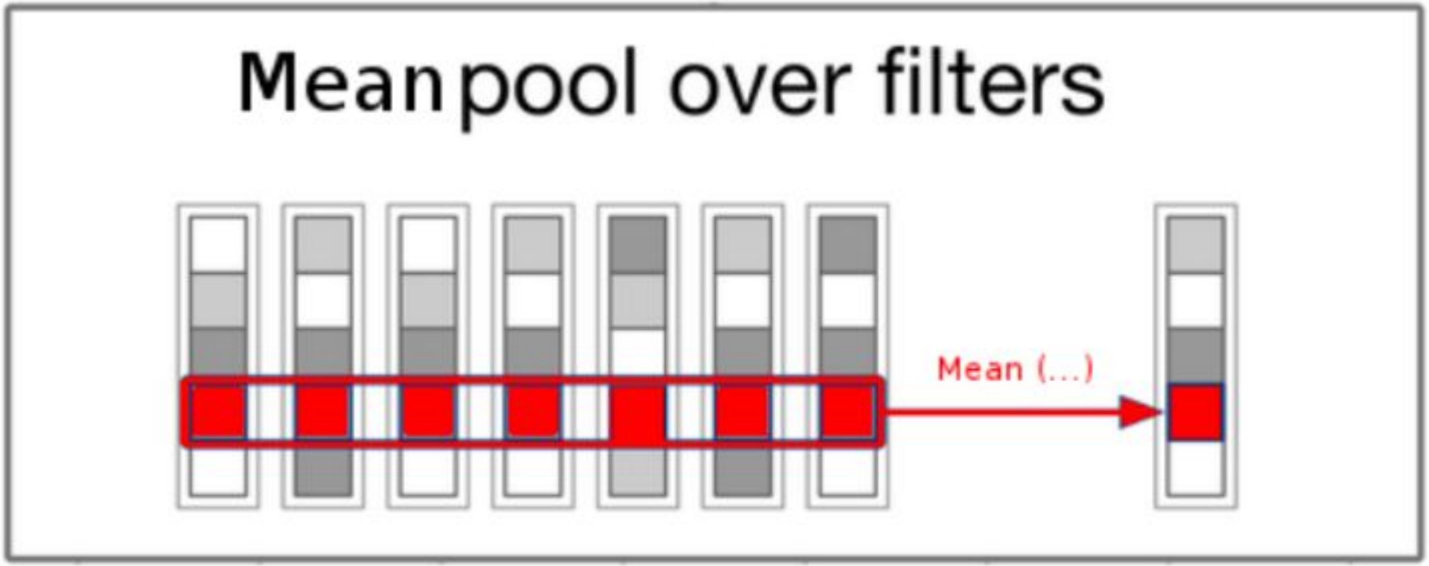
L = [5 3]

Quiz

Knowing what we know about padding and stochastic gradient descent, why don’t we need to use padding for stochastic gradient descent?

You have to be careful

Mean Pooling



Mean Pooling

$$\begin{bmatrix} 1 & 10 & 8 & 17 & 13 & 17 \end{bmatrix} = \frac{66}{6} = 11.0$$

Mean Pooling

$$\begin{bmatrix} 1 & 10 & 8 & 17 & 13 & 17 \end{bmatrix} = \frac{66}{6} = 11.0$$

$$\begin{bmatrix} 22 & 24 & 9 & 13 \end{bmatrix} = \frac{68}{4} = 17.0$$

Mean Pooling

$$\begin{bmatrix} 1 & 10 & 8 & 17 & 13 & 17 \end{bmatrix} = \frac{66}{6} = 11.0$$

$$\begin{bmatrix} 22 & 24 & 9 & 13 \end{bmatrix} = \frac{68}{4} = 17.0$$

$$\begin{bmatrix} 5 & 4 & 8 & 9 & 10 & 34 \\ 6 & 3 & 1 & 4 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{66}{6} \\ \frac{68}{6} \end{bmatrix} = \begin{bmatrix} 11.0 \\ 11.\bar{3} \end{bmatrix}$$

Mean Pooling

```
>>> x
array([[ 1, 10,  8, 17, 13, 17],
       [22, 24,  9, 13,  0,  0]])
>>> np.mean(x, axis=1)
array([11.        , 11.33333333])
>>> lengths
array([6, 4])
>>> np.sum(x, axis=1) / lengths
array([11., 17.])
```

Now that we have a more holistic view of the pipeline...

Let’s look at some code!

Things you already know that are helpful!

-

Things you already know that are helpful!

- CLUSTERING

Things you already know that are helpful!

- CLUSTERING
- Cosine similarity

Things you already know that are helpful!

- CLUSTERING
- Cosine similarity
- Naive Bayes, Logistic Regression, Support Vector Machines

Some final tips

- Word2vec is actually pretty good - pretrained embeddings are great and all you need 95% of the time
- You can concatenate word embeddings to improve their performance sometimes
- Theory is great, coding from scratch is great - look for packages that do what you want to do before sinking time into a project
- **Pay attention** to preprocessing because those mistakes will be what bite you in the end, not your in-depth knowledge of deep learning architectures
- Garbage In, Garbage Out is still the biggest part of doing this kind of data science, like all of data science

Any last questions? :)