week 1:

**neural machine translation (NMT)**

**seq2seq**

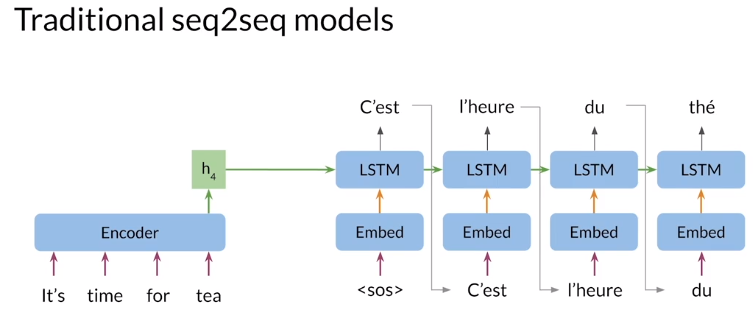
- maps variable-length sequences to fixed-length memory

- inputs and outputs can have different lengths

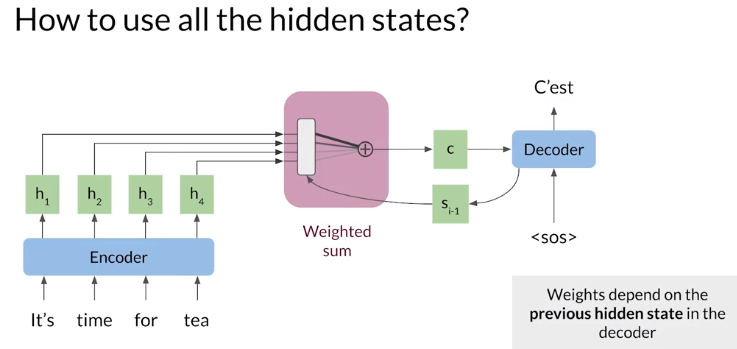
- LSTMs and GRUs to avoid vanishing and exploding gradient problems

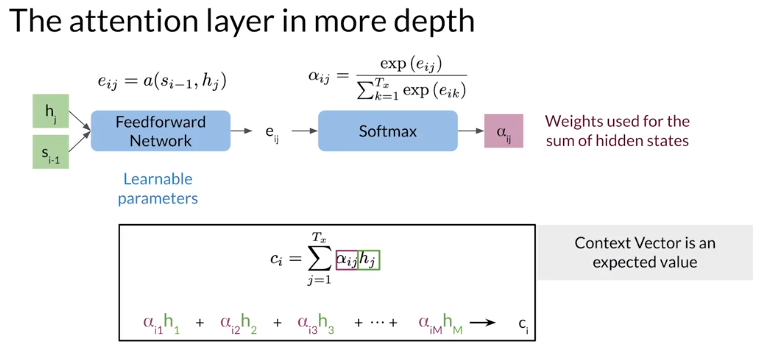
encoder - decoder

Information bottleneck: a fixed amount of information goes from the encoder to the decoder



**Seq2seq model with attention**





The sequential nature of models you learned in the previous course (RNNs, LSTMs, GRUs) does not allow for parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples

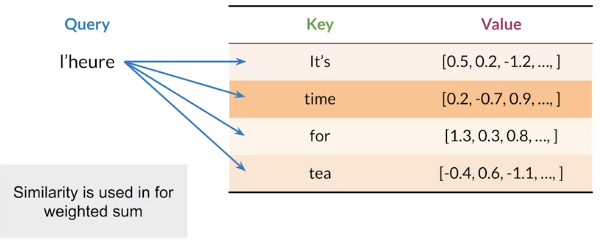
if you rely on sequences and you need to know the beginning of a text before being able to compute something about the ending of it, then you can not use parallel computing. You would have to wait until the initial computations are complete. This is not good, because if your text is too long;

* it will take a long time for you to process it
* you will lose a good amount of information mentioned earlier in the text as you approach the end.

**Queries, Keys, Values, and Attention**

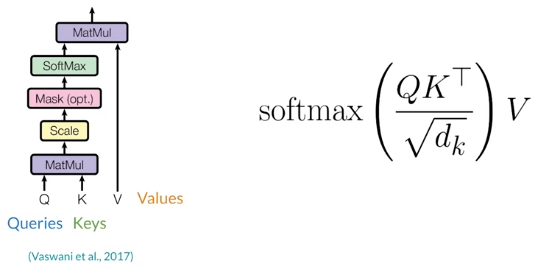
There have been multiple variations on attention with some models don’t rely on RNN

Retrieving information using Query, Key, Value introduced in 2017 paper Attention is all you need

Quetry, key and value are represented by embedding vectors

The similarity between words is called alignment

**Scaled dot-product attention**



The queries for each step are packed together into a matrix Q, so attention can be computed simultaneously for each query

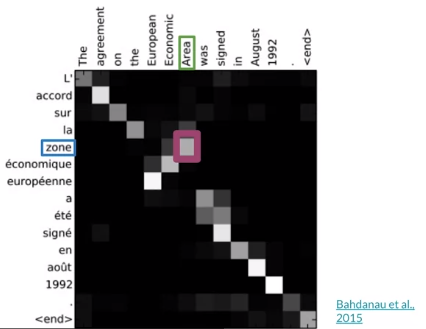
The keys and values are also packed into matrices K and V, you can think of the keys and values as being the same

1. Queries and keys are multiplied together to get a matrix of alignments course (similarity between Q and K)
2. Then scaled by the square root of the key vector dimension, improves performance for larger model sizes, seen as a regularization constant
3. Scale scores are converted to weights using the SoftMax function
4. Weights and value matrices are multiplied to get the attention vectors for each query

Just two matrix multiplications and a SoftMax, no neural networks, much faster to compute, but means the alignments between the source (key) and target (query) languages must be learned elsewhere

Alignment is learned in the input embeddings or in other linear layers before the attention layer

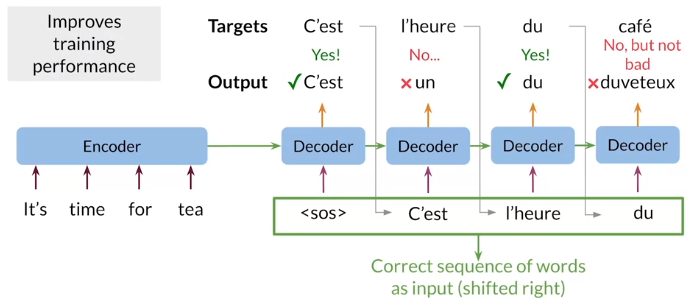
Alignment between languages



Similar words have large weights

Flexible attention: works for languages with different grammar structures

**Teacher Forcing**

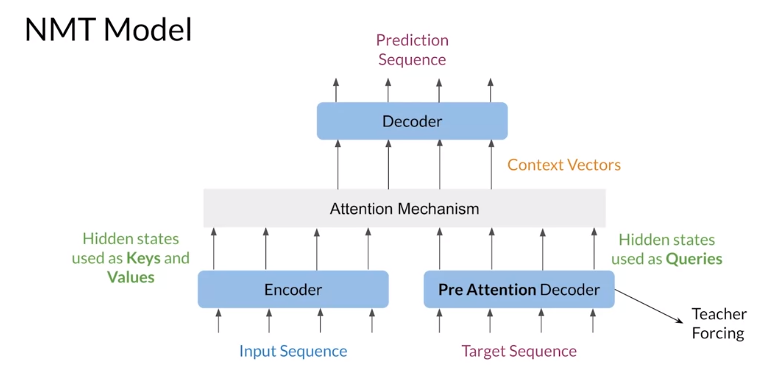
curriculum learning: Slowly start using decoder outputs over time, so that leads into training you are no longer feeding in the target words

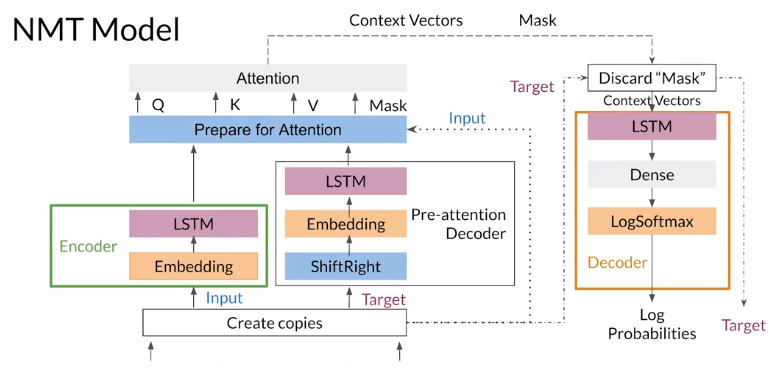
**MNT Model with Attention**

The decoder must pass the hidden state to the Attention Mechanism, difficult to implement, so a pre-attention decoder is introduced

Using 2 decoders:

* Pre-attention decoder to provide hidden states
* Post-attention decoder provides the translation



Shift Right: shift each sequence to the right and add start token

**BLEU Score (Bilingual Evaluation Understudy)**

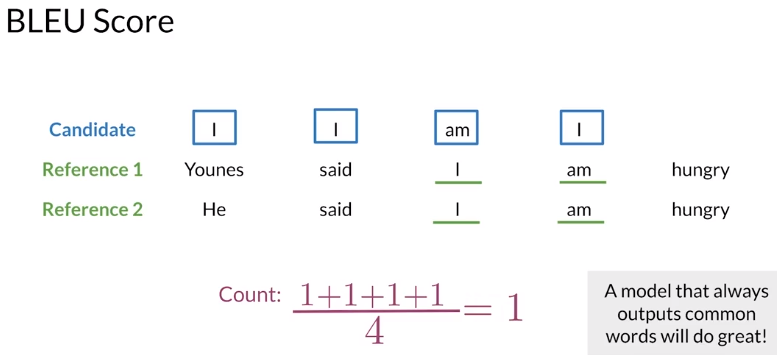
Compares candidate translations to reference (human) translations

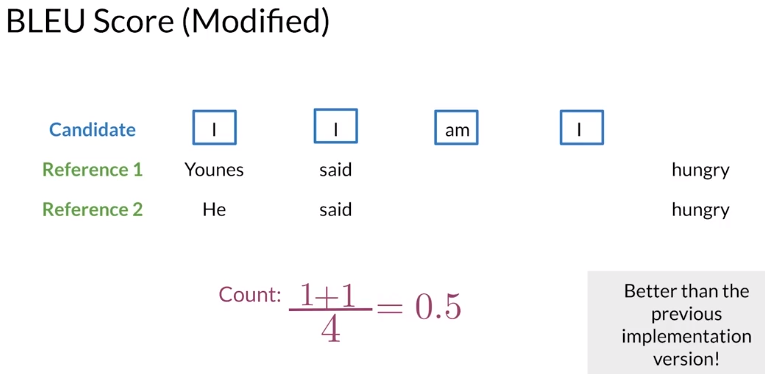
Closer to 1, the better

Using N-Grams, below using unigrams

BLEU doesn’t consider semantic meaning

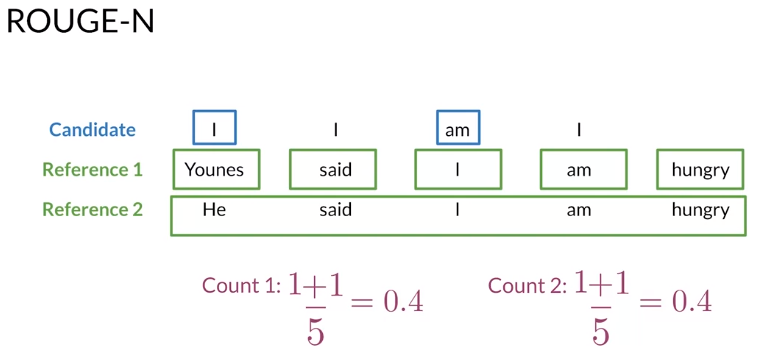
BLUE doesn’t consider sentence structure

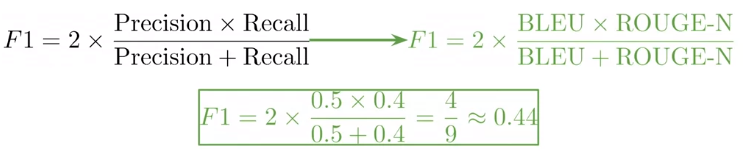




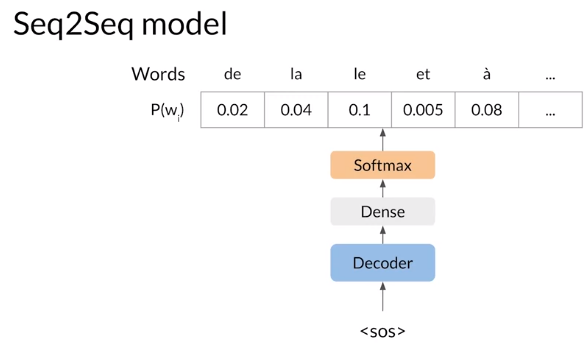
ROUGE-N Score (Recall-Oriented Understudy for Gisting Evaluation)

How many words from reference appear in the candidate translations





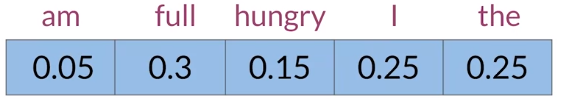
**Sampling and decoding**



Greedy decoding: selects the most probable word at each step, but the best word at each step may not be the best for longer sequences, can be fine for shorter sequences, but limited by inability to look further down the sequence



**Random sampling**



Often a little too random for accurate translation

Solution: assign more weight to more probable words, and less weight to less probable words

**Temperature**:

* Scale 0-1
* Can control for more or less randomness in predictions
* Lower temperatures settings: more confident, conservative network
* Higher temperatures settings: more excited, random network

**Beam Search**

Most probable translation is not the one with the most probable word at each step

Solution: calculate probability of multiple possible sequences

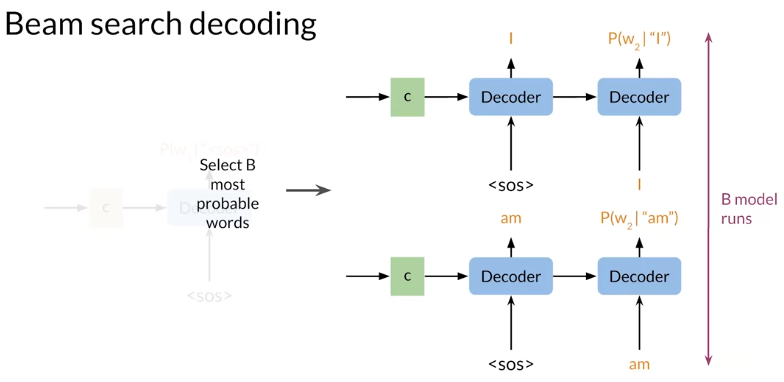
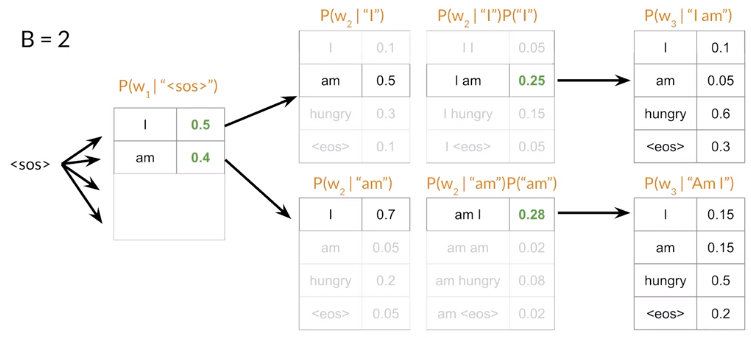
Beam width B determines number of sequences you keep

Until all B most probable sequences end with <EOS>

Beam search with B=1 is greedy decoding

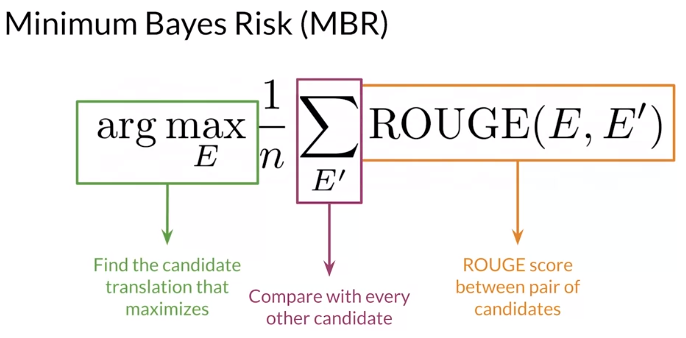
Problems:

* Penalizes long sequences, so you should normalize by the sentence length
* Computationally expensive and consumes a lot of memory



**Minimum Bayes Risk (MBR)**

* Generate several candidate translations
* Assign a similarity to every pair using a similarity score (such as ROUGE)
* Select the sample with the highest average similarity
* Better performance than random sampling and greedy decoding



**Week 2:**