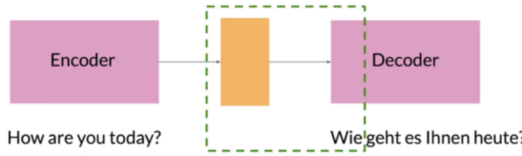
**Neural Machine Translation**

## * Seq2Seq

- Encoder and Decoder  LSTMs used by both.

Seq2Seq model

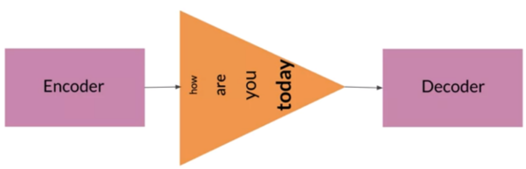
- Introduced by Google in 2014
- Maps variable-length sequences to fixed-length memory
- LSTMs and GRUs are typically used to overcome the vanishing gradient problem




The orange rectangle in the above figure represents the encoders final hidden state, which tries to capture all the info collected from each input step before feeding it to the decoder. Initial state for the decoder.

Major limitation is the information bottleneck

The information bottleneck




Long sequence as individual inputs begin stacking up inside the encoders final hidden states, because Seq2Seq uses a fixed length memory, longer sequences become problematic. Another issue surfaces as the later input steps in the sequence are given more importance.

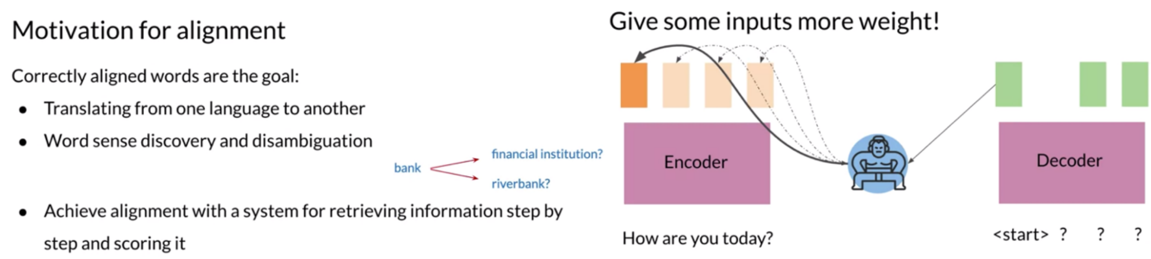
So the power of Seq2Seq which lies in its ability to let inputs and outputs be different sizes, becomes its weakness when the input itself is a large size. Because the encoder hidden states is of a fixed size, and longer inputs become bottlenecked on their way to the decoder.

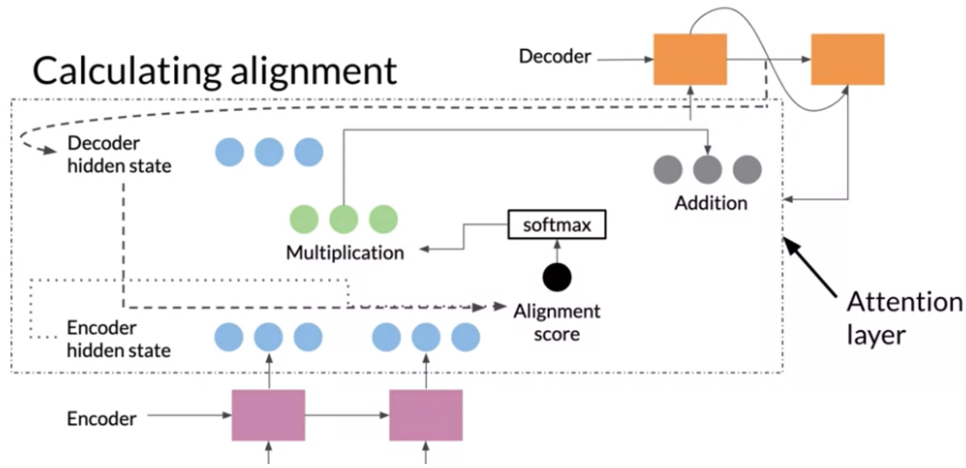
Solution: focus attention in the right place

- 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


---

## Alignment



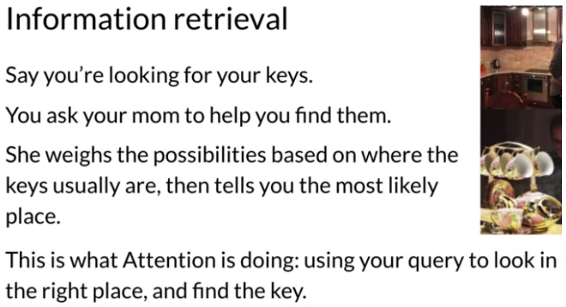



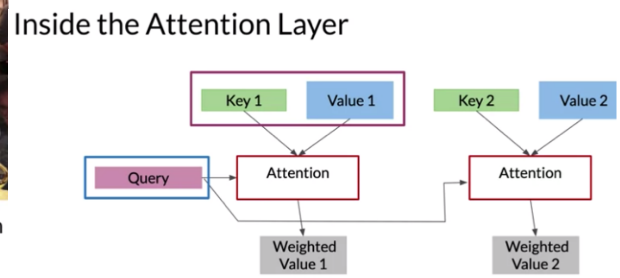

First, get all of the available hidden states ready for the encoder and do the same for the first hidden states of the decoder. In the simplified example, there are two encoder hidden states and one decoder hidden states. Next, score each of the encoder hidden states by getting its dot product between each encoder state and decoder hidden states. If one of the scores is higher than the others, it means that this hidden state will have more influence than the others on the output. Then you will run scores through softmax, so each score is transformed to a number between 0 and 1, this gives you your attention distribution. Take each encoder hidden state, and multiply it by its softmax score, which is a number between 0 and 1, this results in the alignments vector. Almost there, now just add up everything in the alignments vector to arrive at what's called the context vector. This guy is what you feed into the decoder, so you can see what's happening here can be distilled down to a few mathematical operations that are scoring words based on their importance. This is the magic of attention.

---

## Attention

- Attention is an added layer that lets a model focus on what's important
- Queries, Values, and Keys are used for information retrieval inside the Attention layer
- This flexible system finds matches even between languages with very different grammatical structures

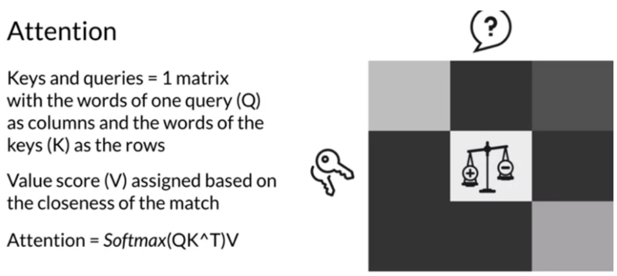


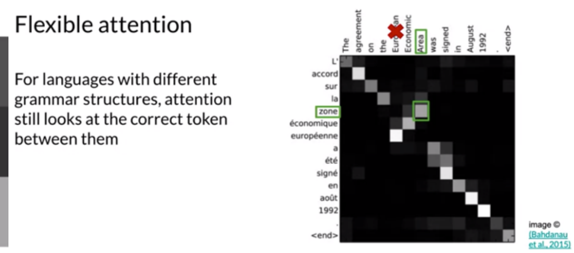




Query comes from decoder hidden states

Key/value comes from encoder hidden states

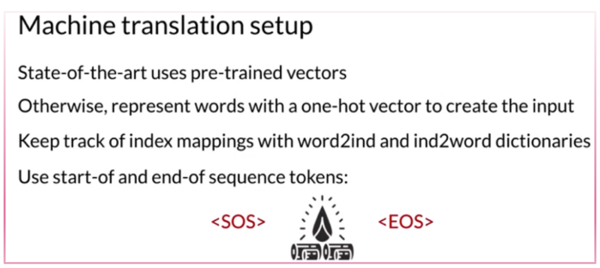
Dot product key and query, similar vectors have higher value. Weighted sum given to each value is determined by prob that key matches the query. Run attention weight through softmax, fit a distribution between 0 and 1. Scale dot product attention.

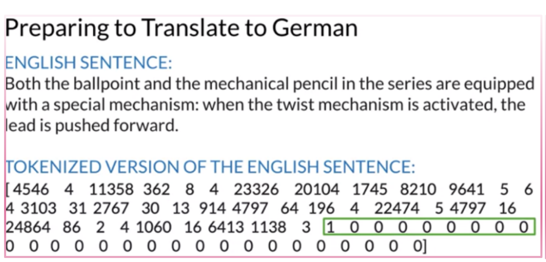


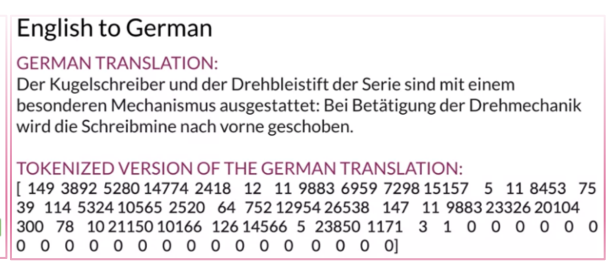




## Setup for Machine Translation

Data – Dictionary of English to German translation

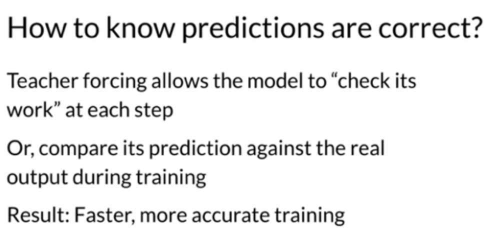


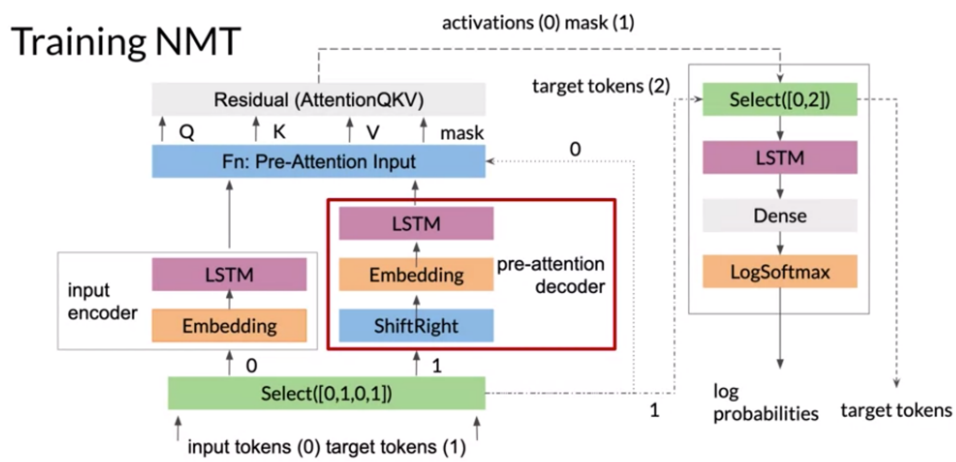




## Training an NMT with attention

Teacher Forcing provides faster training and higher accuracy by allowing the model to use the decoder's actual output to compare its predictions against.



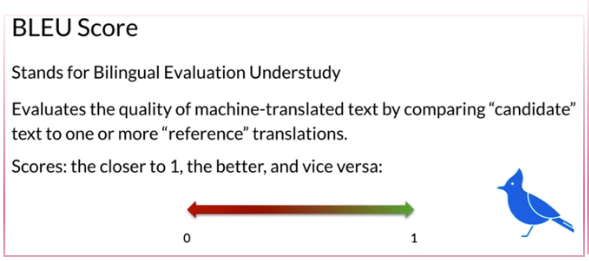


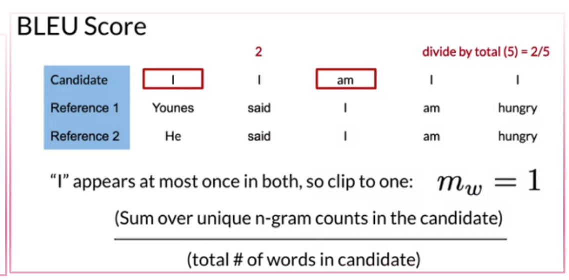




---

### Evaluation for Machine Translation

Bleu score = Bilingual Evaluation understudy



## BLEU score is great, but...

Consider the following:

- BLEU doesn't consider semantic meaning
- BLEU doesn't consider sentence structure:

"Ate I was hungry because!"

```
import numpy as np               # import numpy to make numerical computations.
import nltk                      # import NLTK to handle simple NL tasks like tokenization.
from nltk.util import ngrams
nltk.download('punkt')
import math
from collections import Counter  # import the Counter module.
!pip3 install 'sacrebleu'        # install the sacrebleu package.
import sacrebleu                 # import sacrebleu in order compute the BLEU score.
import matplotlib.pyplot as plt
```

### 1.2 Defining the BLEU Score

You have seen the formula for calculating the BLEU score in this week's lectures. More formally, we can express the BLEU score as:

$$BLEU = BP\Big(\prod_{i=1}^{4} precision_i\Big)^{(1/4)}$$

with the Brevity Penalty and precision defined as:

$$BP = min\Big(1, e^{(1-(ref/cand))}\Big)$$

$$precision_i = \frac{\sum_{snt \in cand} \sum_{i \in snt} min\Big(m^i_{cand}, m^i_{ref}\Big)}{w^i_t}$$

where:

- $m^i_{cand}$, is the count of i-gram in candidate matching the reference translation.
- $m^i_{ref}$, is the count of i-gram in the reference translation.
- $w^i_t$, is the total number of i-grams in candidate translation.

## ROUGE = recall oriented understudy for Gisting Evaluation

### ROUGE

Recall-Oriented Understudy for Gisting Evaluation

Evaluates quality of machine text

Measures precision and recall between generated text and human-created text

### ROUGE evaluation

| | | | | | | |
|---|---|---|---|---|---|---|
| Model | The | cat | had | striped | orange | fur |
| Reference | The | cat | had | orange | fur | |

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

Precision = How much of the model text was relevant?

### Recall in ROUGE

| | | | | | | |
|---|---|---|---|---|---|---|
| Model | The | cat | had | striped | orange | fur |
| Reference | The | cat | had | orange | fur | |

$$\frac{\text{(Sum of overlapping unigrams in model and reference)}}{\text{(total \# of words in reference)}} \quad \frac{5}{5}$$

### Precision in ROUGE

| | | | | | | |
|---|---|---|---|---|---|---|
| Model | The | cat | had | striped | orange | fur |
| Reference | The | cat | had | orange | fur | |

$$\frac{\text{(Sum of overlapping unigrams in model and reference)}}{\text{(total \# of words in model)}} \quad \frac{5}{6}$$

Precision = 0.83

### Problems in ROUGE

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

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

- BLEU score compares "candidate" against "references" using an n-gram average
- BLEU doesn't consider meaning or structure
- ROUGE measures machine-generated text against an "ideal" reference

---

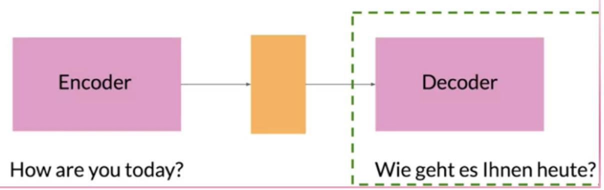
## Sampling and Decoding

How to construct the translated sentence.

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

here's a reminder of where your model is in the process when sampling, and decoding comes into play. After all the necessary calculations have been performed on the encoder hidden states, and your model is ready to predict the next token, how will you choose to do it? With the most probable token or by taking a sample from a distribution

### Seq2Seq model




### 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...

Ich habe Hunger.

I am hungry.

I am, am, am, am...

### Random sampling

| am | full | hungry | I | the |
|---|---|---|---|---|
| 0.05 | 0.3 | 0.15 | 0.25 | 0.25 |

Often a little too random for accurate translation!

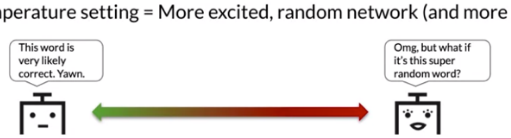
Solution: Assign more weight to more probable words, and less weight to less probable words.

### Temperature

In sampling, temperature is a parameter allowing for more or less randomness in predictions

Lower temperature setting = More confident, conservative network

Higher temperature setting = More excited, random network (and more mistakes)




### Beam search decoding

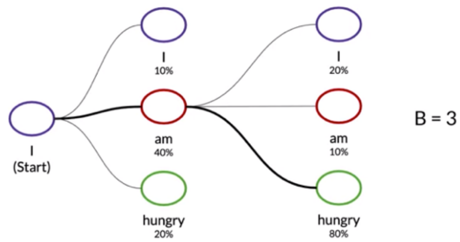
A broader, more exploratory decoding alternative

Selects multiple options for the best input based on conditional probability

Number of options depends on a predetermined beam width parameter B

Selects B number of best alternatives at each time step

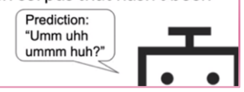
### Beam search example




### Problems with beam search

Since the model learns a distribution, that tends to carry more weight than single tokens

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




### Problems with beam search

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

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

"I like the United States, because the Uhm is big. "

Even with 11^2 good translations, the most probable one will still be "Uhm."

### Minimum Bayes Risk (MBR)

Compares 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 ✨

### Example: MBR Sampling

To generate the scores for 4 samples:

1. Calculate similarity score between sample 1 and sample 2
2. Calculate similarity score between sample 1 and sample 3
3. Calculate similarity score between sample 1 and sample 4
4. Average the score of the first 3 steps (Usually a weighted average)
5. Repeat until all samples have overall scores

---

# References

- **Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (Raffel et al, 2019)**

- Reformer: The Efficient Transformer (Kitaev et al, 2020)

- Attention Is All You Need (Vaswani et al, 2017)

- Deep contextualized word representations (Peters et al, 2018)

- The Illustrated Transformer (Alammar, 2018)

- The Illustrated GPT-2 (Visualizing Transformer Language Models) (Alammar, 2019)

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al, 2018)

- How GPT3 Works - Visualizations and Animations (Alammar, 2020)