Transformers and Pretrained Language Models

Self-Attention Networks: Transformers

Language Acquisition

English young adults know in the range of 30,000 to 100,000 words

- 7 to 10 new words per day
- Part of them are learnt by conversation
- Very few words through explicit lexical instruction

But mature speakers have an active vocabulary of about 2,000 words!!

What about the rest?

Language Acquisition

The most part of them are acquired as by-product of reading

- Meaning of new words is acquired through lexical diversity of the texts
- At some points the vocabulary growth *exceeds* the rate of new words read
- Grounding words in the real world through perception plays also an important role

These facts are in support of the distributional hypothesis

Moreover, knowledge about meaning can be used long after its acquisition

Pretraining

The process of learning some sort of representation of meaning for words or sentences by *processing very large amounts of text*

- As a result we have a ***Pretrained Language Model***

Transformers are the most common architecture for building a Pretrained Language Model

- Self-Attention
- Positional encoding

Transformers

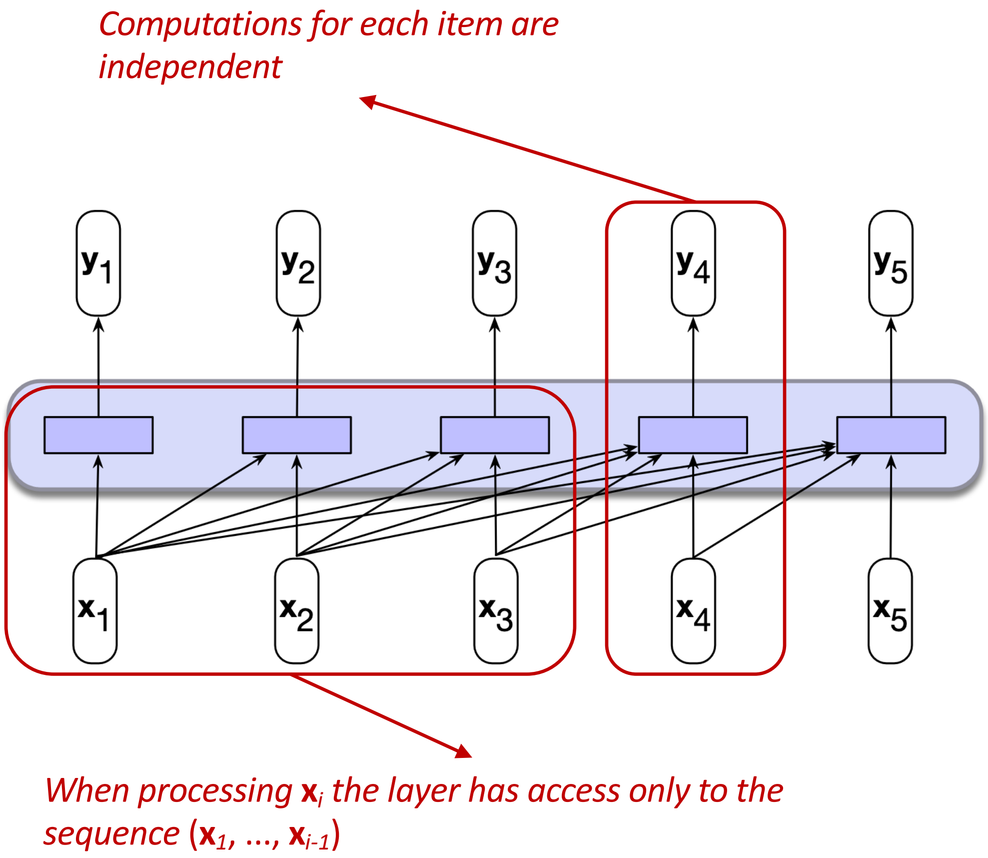
Transformers manage long distance dependencies as LSTM do, but they are ***parallel feed forward networks***

- They map ($\mathbf{x}_1$, ..., $\mathbf{x}_n$) → ($\mathbf{y}_1$, ..., $\mathbf{y}_n$)
- Made by many *transformer blocks*
- Each transformer block is made up by linear layers, FFN and *self-attention layers*

Self-attention layer

Allow for language modeling and autoregressive language generation

Easily parellizable



Self-attention

The core idea is to compare an item with the sequence of all the preceding ones in the same sequence using a dot-product score

$$\begin{aligned}
\mathrm{score}(\mathbf{x}_i, \mathbf{x}_j) &= \mathbf{x}_i \cdot \mathbf{x}_j \\
\alpha_{ij} &= \mathrm{softmax}(\mathrm{score}(\mathbf{x}_i, \mathbf{x}_j)) \;\; \forall j \leq i \\
&= \frac{\exp(\mathrm{score}(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{k=1}^{i} \exp(\mathrm{score}(\mathbf{x}_i, \mathbf{x}_k))} \;\; \forall j \leq i
\end{aligned}$$

$$\mathbf{y}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{x}_j$$

Self-attention

Self-attention in Transformer is more sophisticated, and it is linked to the *roles* the same word embedding can play during this process

- ***Query***: the current focus of attention when being compared to all of the other preceding inputs.

Self-attention

Self-attention in Transformer is more sophisticated, and it is linked to the *roles* the same word embedding can play during this process

- ***Key***: a *preceding input* being compared to the current focus of attention.

Self-attention

Self-attention in Transformer is more sophisticated, and it is linked to the *roles* the same word embedding can play during this process

- ***Value***: when used to compute the output for the current focus of attention.

$$\mathbf{q}_i = \mathbf{W}^{\mathbf{Q}}\mathbf{x}_i;\ \ \mathbf{k}_i = \mathbf{W}^{\mathbf{K}}\mathbf{x}_i;\ \ \mathbf{v}_i = \mathbf{W}^{\mathbf{V}}\mathbf{x}_i$$

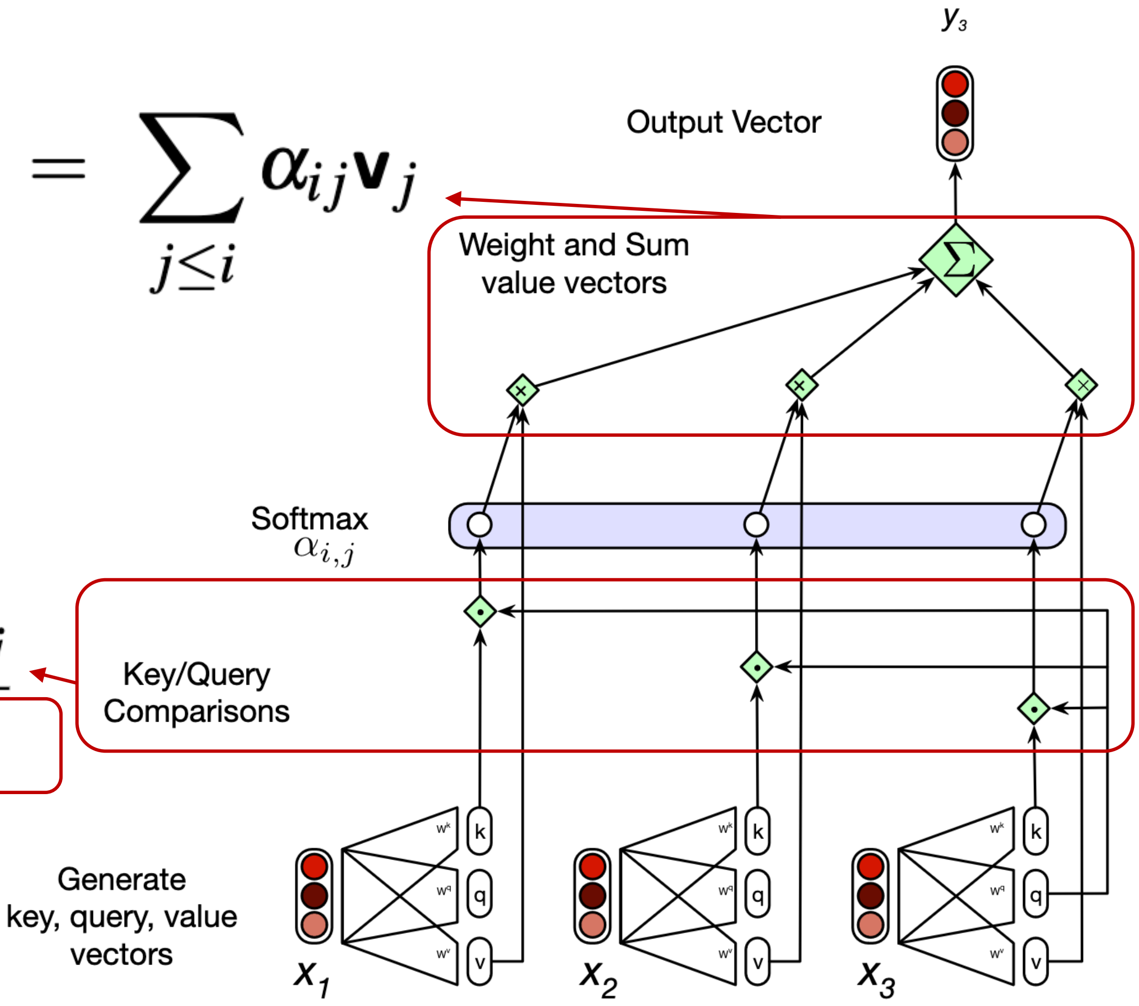
$\mathbf{q}_i$, $\mathbf{k}_i$, d_k dimensionality
$\mathbf{v}_i$, d_v dimensionality

Self-attention

$$\mathbf{y}_i = \sum_{j \le i} \alpha_{ij} \mathbf{v}_j$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\boxed{\sqrt{d_k}}}$$

Scaling avoids too large values for the dot-product



Self-attention

Parallel implementation using matrix products

$$\mathbf{Q} = \mathbf{XW^Q};\ \ \mathbf{K} = \mathbf{XW^K};\ \ \mathbf{V} = \mathbf{XW^V} \quad \mathbf{Q} \in \mathbb{R}^{N \times d_k}\ \mathbf{K} \in \mathbb{R}^{N \times d_k}\ \mathbf{V} \in \mathbb{R}^{N \times d_v}$$

$$\text{SelfAttention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\boxed{\mathbf{QK^T}}}{\sqrt{d_k}}\right)\mathbf{V}$$

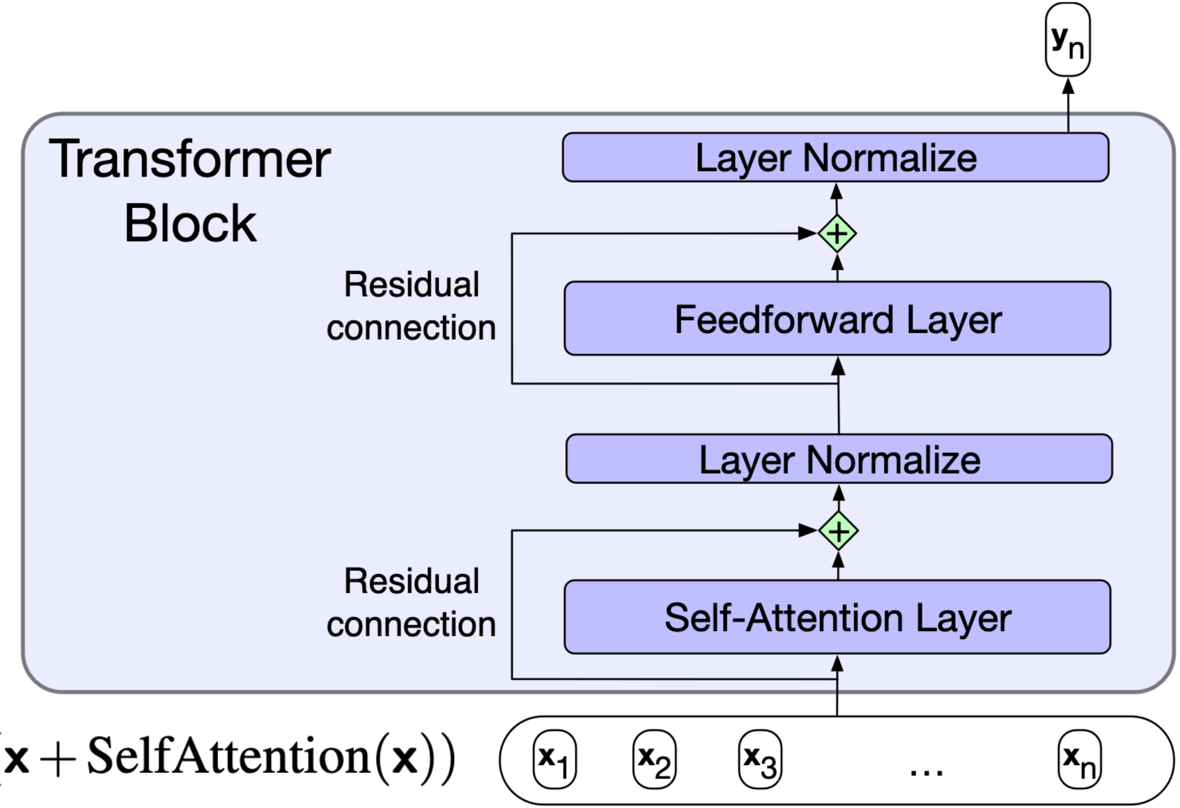
The product ***QK***$^{\mathsf{T}}$ *computes also the scores* ***<u>for the keys that follow the query</u>***

$$\begin{array}{cc} N & \begin{array}{|c|c|c|c|c|} \hline q1{\cdot}k1 & -\infty & -\infty & -\infty & -\infty \\ \hline q2{\cdot}k1 & q2{\cdot}k2 & -\infty & -\infty & -\infty \\ \hline q3{\cdot}k1 & q3{\cdot}k2 & q3{\cdot}k3 & -\infty & -\infty \\ \hline q4{\cdot}k1 & q4{\cdot}k2 & q4{\cdot}k3 & q4{\cdot}k4 & -\infty \\ \hline q5{\cdot}k1 & q5{\cdot}k2 & q5{\cdot}k3 & q5{\cdot}k4 & q5{\cdot}k5 \\ \hline \end{array} \\ & N \end{array}$$

The elements in the upper-triangular portion of the matrix are zeroed out

Transformer blocks

Residual connections are intended to reduce vanishing gradients



$$\mathbf{z} = \text{LayerNorm}(\mathbf{x} + \text{SelfAttention}(\mathbf{x}))$$

$$\mathbf{y} = \text{LayerNorm}(\mathbf{z} + \text{FFN}(\mathbf{z}))$$

Transformer blocks

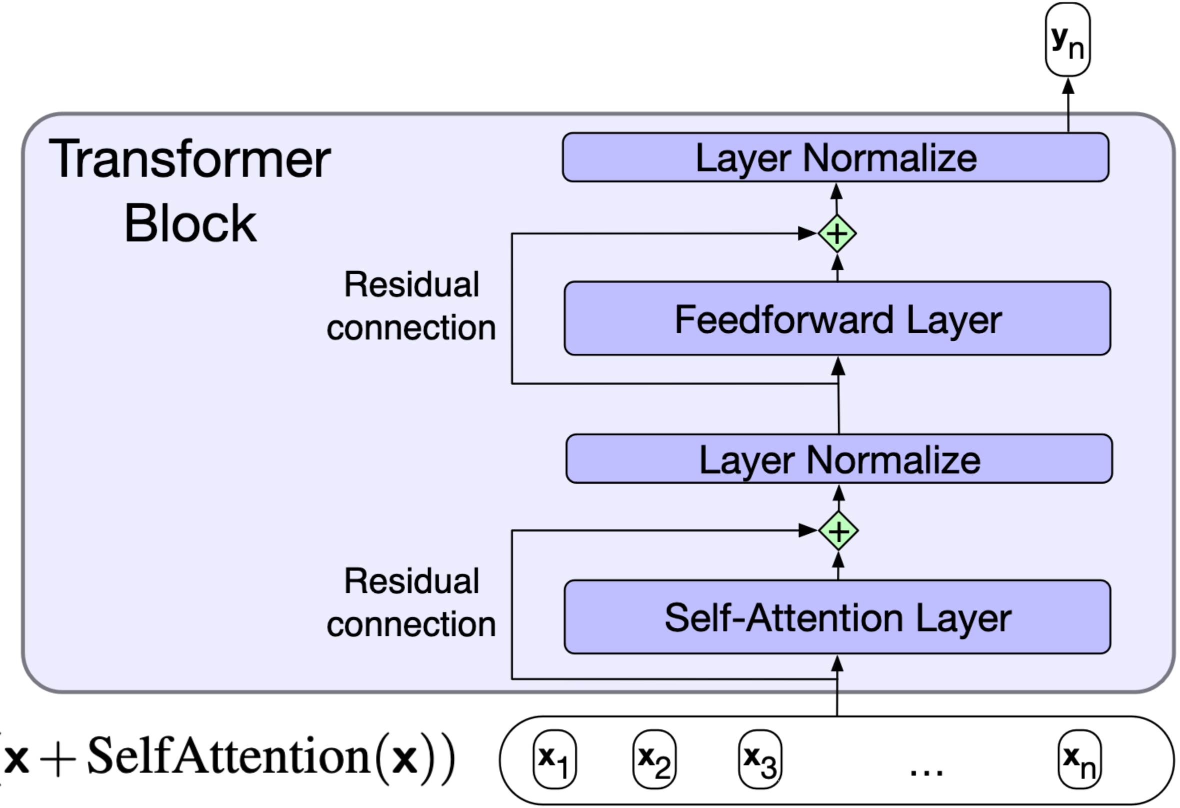
Layer normalization

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i$$

$$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i-\mu)^2}$$

$$\hat{\mathbf{x}} = \frac{(\mathbf{x}-\mu)}{\sigma}$$

$$\text{LayerNorm} = \gamma\hat{\mathbf{x}}+\beta$$



$$\begin{aligned}\mathbf{z} &= \text{LayerNorm}(\mathbf{x}+\text{SelfAttention}(\mathbf{x}))\\ \mathbf{y} &= \text{LayerNorm}(\mathbf{z}+\text{FFN}(\mathbf{z}))\end{aligned}$$

Multi-head attention

The different words in a sentence can relate to each other in many different ways simultaneously.

- i.e. syntactic, semantic, and discourse relationships

A single transformer block can not cope with all of them at the same time

Idea!! Many *parallel attention layers* at the same depth(heads)

Multi-head attention

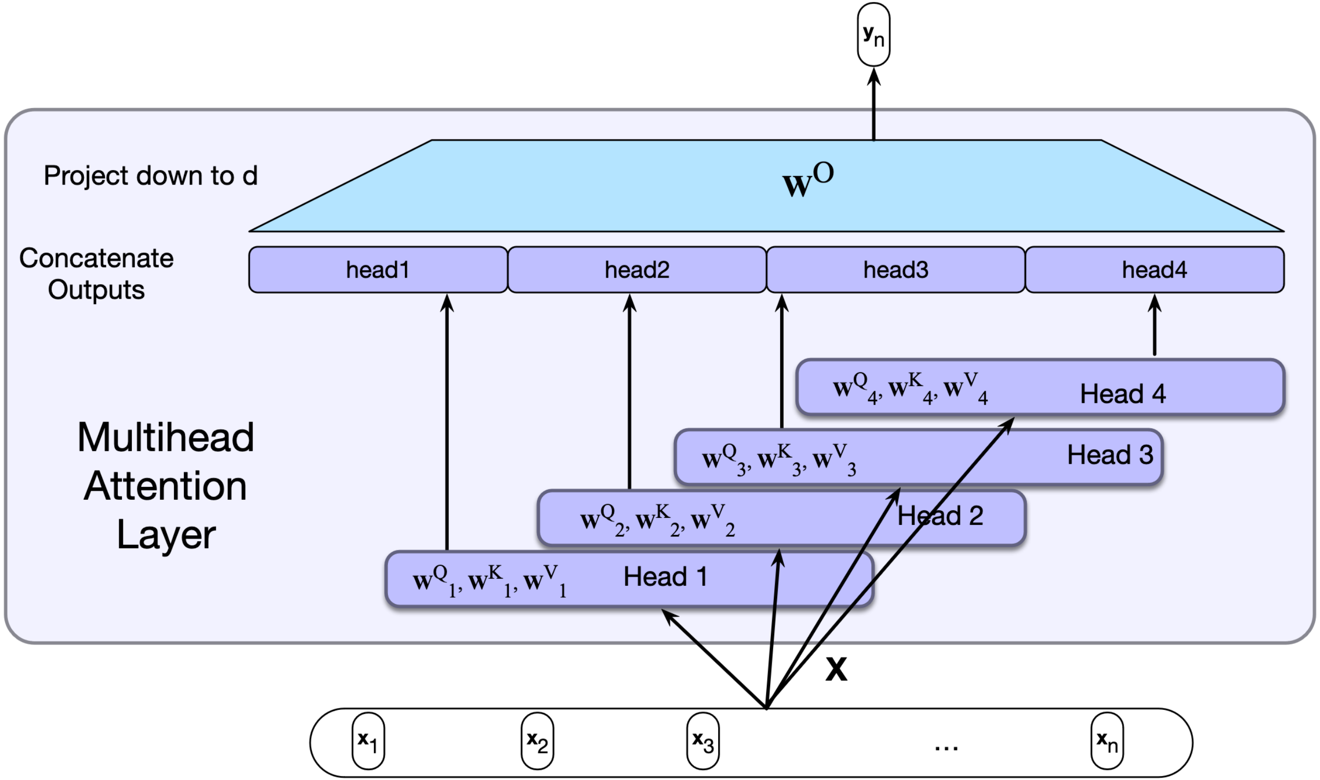
In the *i*-th head

$$\mathbf{W}_i^Q \in \mathbb{R}^{d \times d_k}$$

$$\mathbf{W}_i^K \in \mathbb{R}^{d \times d_k}$$

$$\mathbf{W}_i^V \in \mathbb{R}^{d \times d_v}$$

$$\mathbf{W}^O \in \mathbb{R}^{hd_v \times d}$$

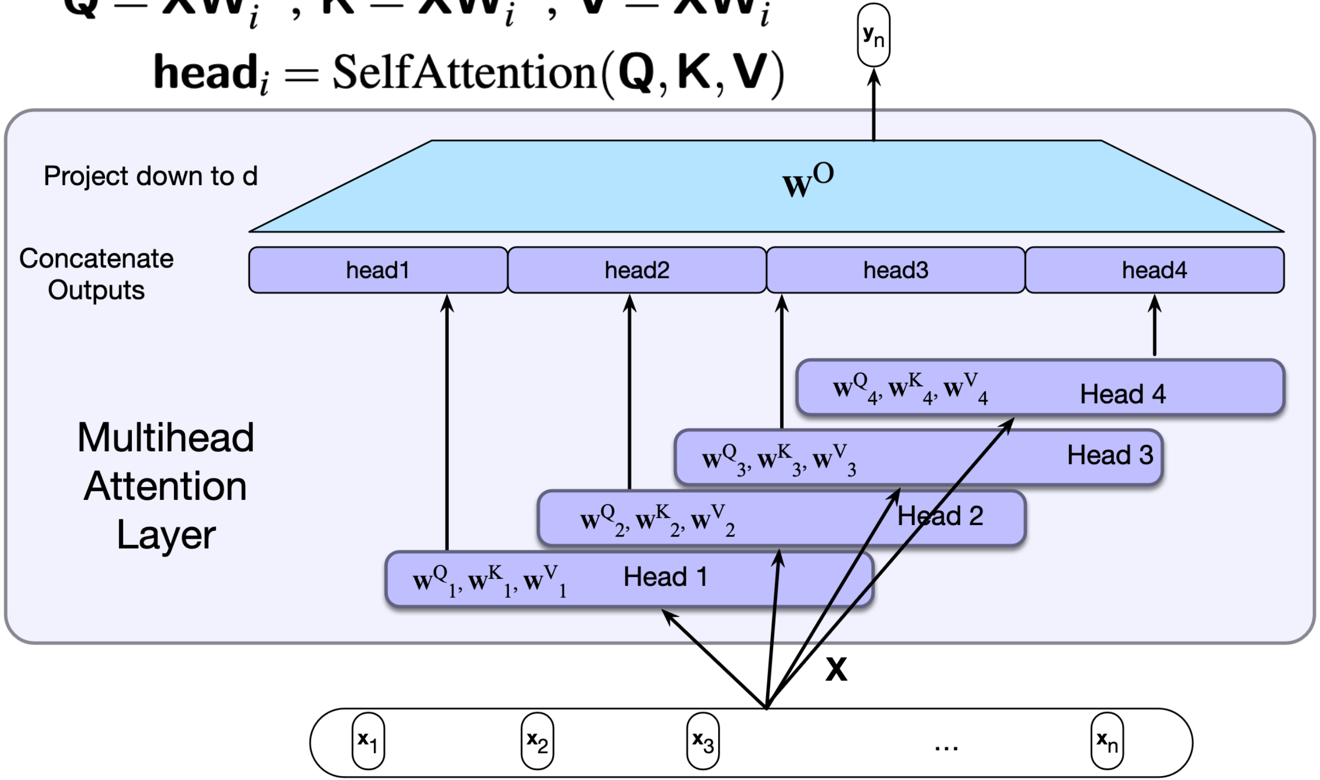


Multi-head attention

$$\mathrm{MultiHeadAttention}(\mathbf{X}) = (\mathbf{head}_1 \oplus \mathbf{head}_2 \ldots \oplus \mathbf{head}_h)\mathbf{W}^O$$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_i^Q \; ; \; \mathbf{K} = \mathbf{X}\mathbf{W}_i^K \; ; \; \mathbf{V} = \mathbf{X}\mathbf{W}_i^V$$

$$\mathbf{head}_i = \mathrm{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$$



Positional embeddings

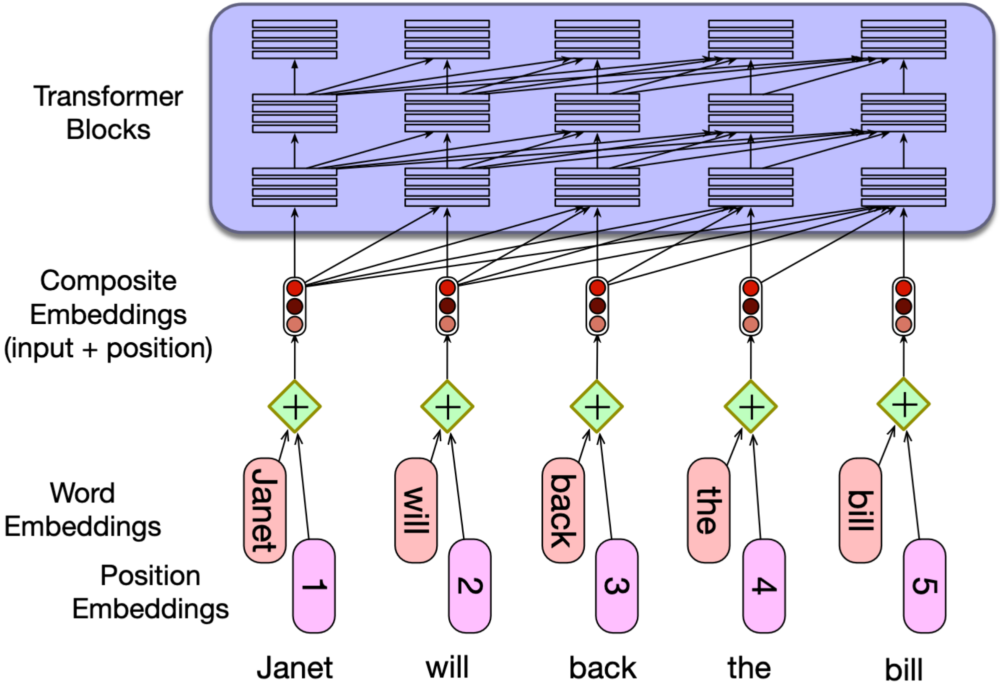
Transformers, as described so far, do not have any notion of the relative, or absolute, positions of the tokens in the input

- Let's scramble the rows in **X**, what happens in self-attention?
  - The same row permutation is in **Q**, **K**, and **V**
  - $\mathbf{QK}^T$ computes the same (permuted) row-column products
  - Self-attention gives the same result

Positional embeddings

Create an embedding for word positions

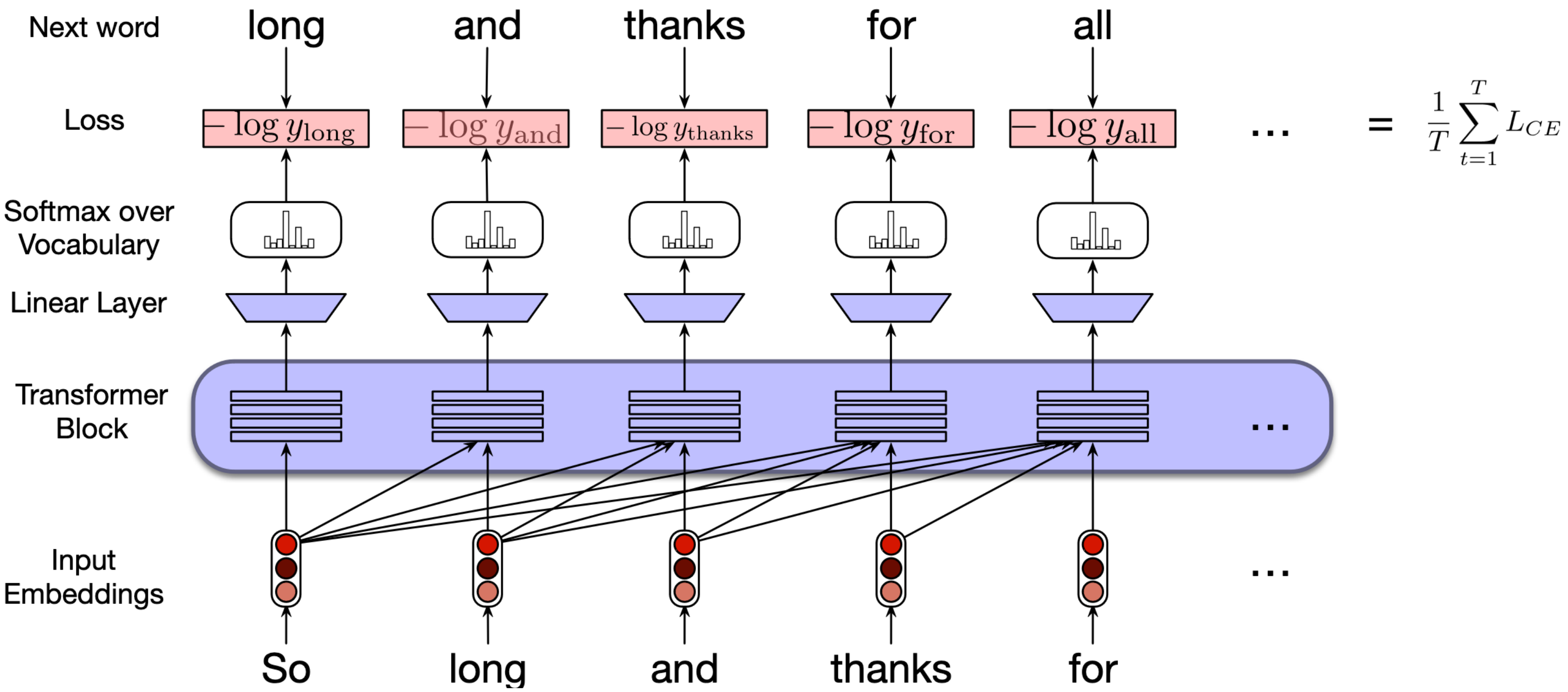
Sum the embedding with word embedding to maintain dimensionality



Transformers and Pretrained Language Models

Transformers as Language Models

Training with teacher forcing



$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

Greedy sampling for generation

Sample a word in the output from the softmax distribution that results from using the beginning of sentence marker, <s>, as the first input.

Use the word embedding for that first word as the input to the network at the next time step, and then sample the next word in the same fashion.

Continue generating until the end of sentence marker, </s>, is sampled or a fixed length limit is reached.

$$\hat{y}_t = \text{argmax}_{\mathrm{w}\in\mathrm{V}} P(w|y_1 \ldots y_{t-1})$$

Transformers and Pretrained Language Models

Sampling

Sampling strategies

Apart from the greedy approach there are various decoding strategies for Natural Language Generation (NLG) tasks

- Based on sampling strategies from the probabilities computed by softmax over the vocabulary

Sampling strategies

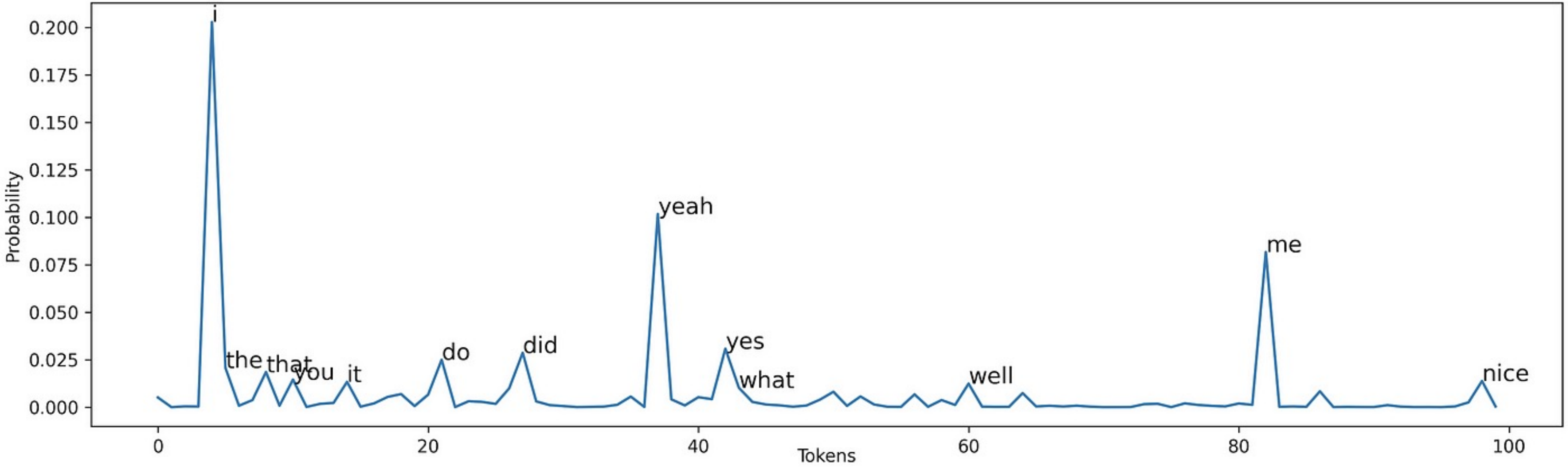
Let’s follow an example by comparing the different strategies on the same context

- Simple encoder-decoder model
- Context: *I love watching movies*
- More details at https://bit.ly/49cgmNI

Random Sampling

The Shannon game for unigrams

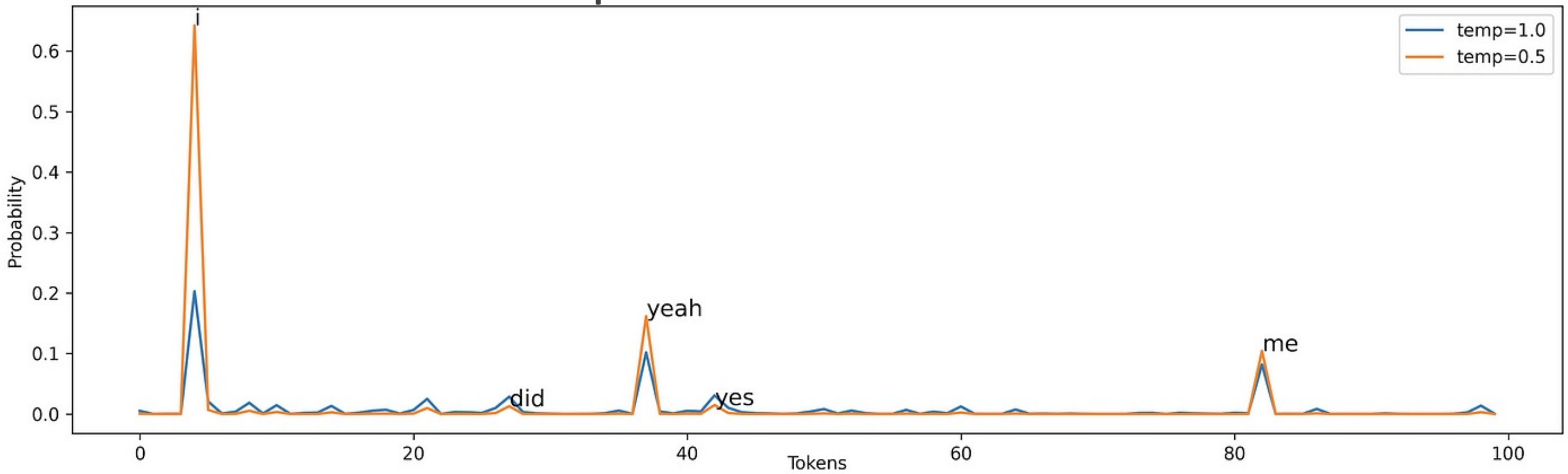
- *I*, *yeah* and *me* are the most probable tokens



Random Sampling with temperature

Temperature t is in the [0, 1] range $P(x_i|x_{1:i-1}) = \frac{\exp(u_i/t)}{\sum_j \exp(u_j/t)}$

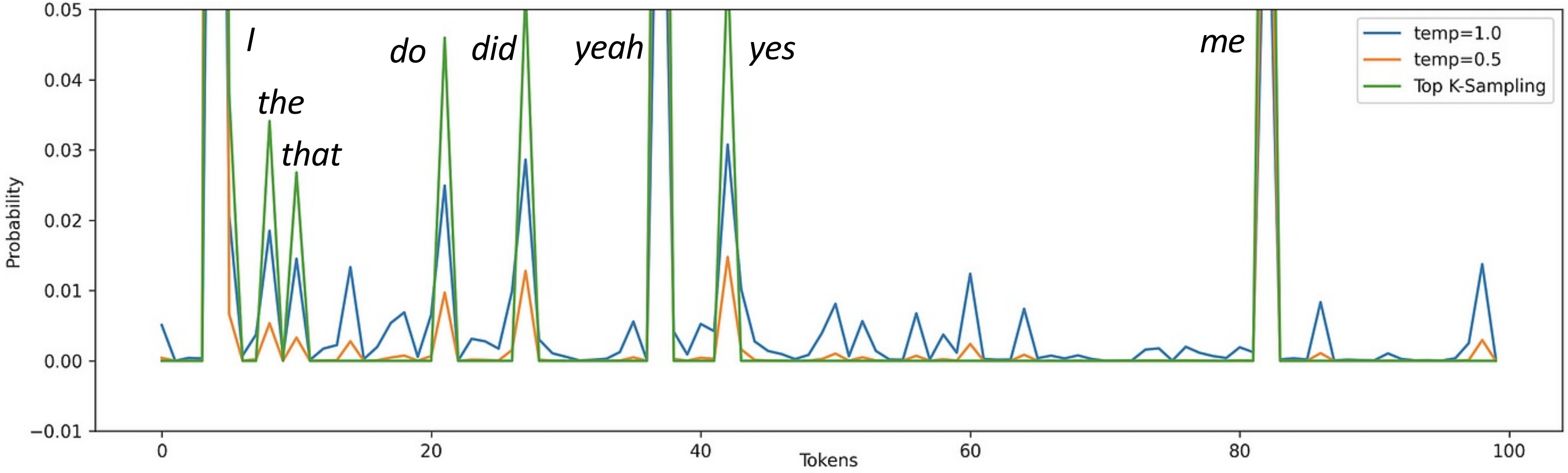
- Probability of the most likely tokens is increased, while the other probabilities are reduced



Top-*K* Sampling

Only the *K* highest probabilities tokens are sampled

- The top *K* probabilities have to be rescaled to sum up to 1



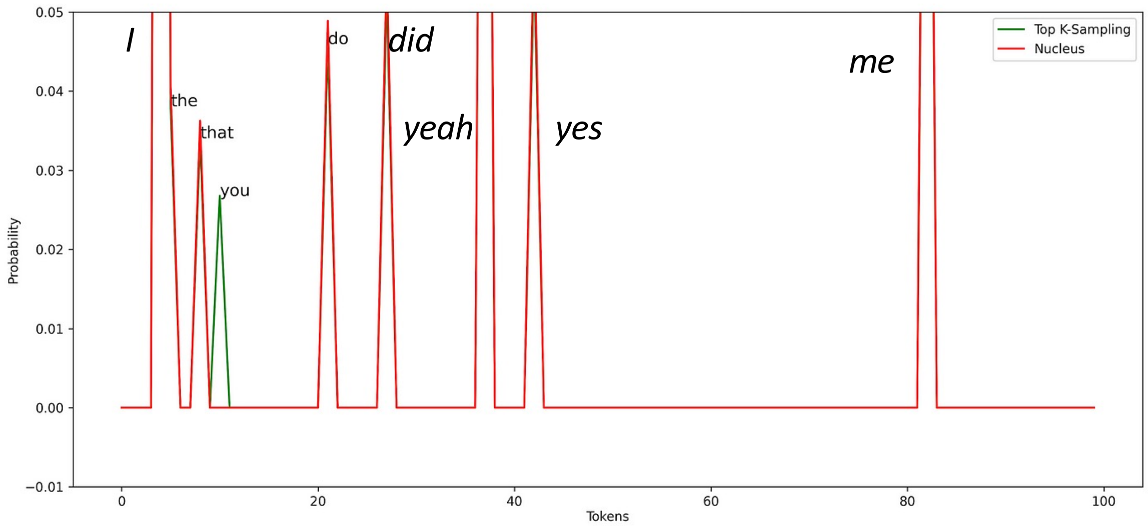
Nucleus (also top-*p*) Sampling

Only the «nucleus» most probable tokens, whose sum of probabilities is enough to exceed *p*, are sampled

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p$$

sort of adaptive

K selection



Transformers and Pretrained Language Models

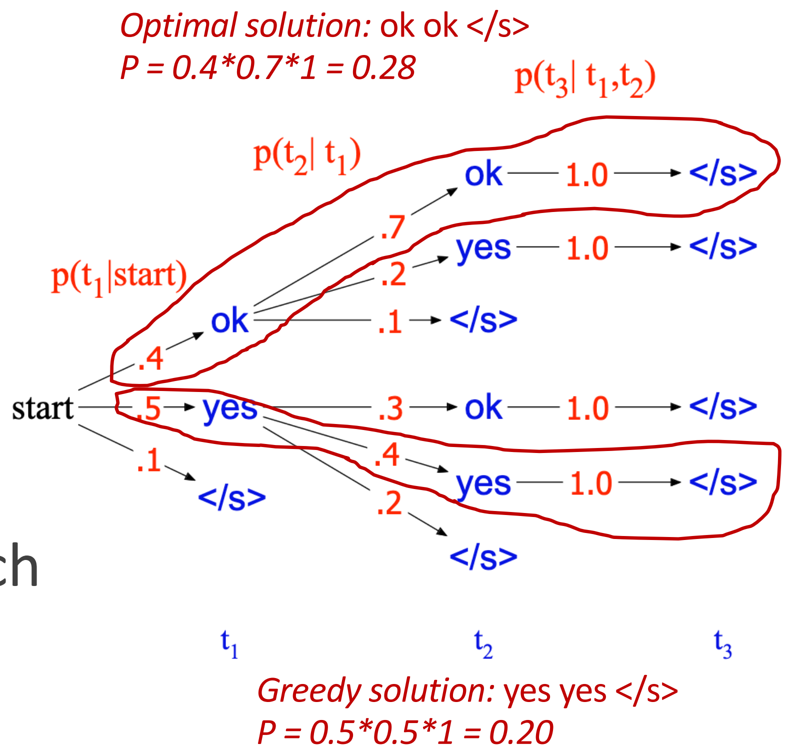
Beam Search

Search tree

Greedy search in the space of solutions is *locally optimal*

We need to implement search strategies that are optimal in a global way

The only algorithm that guarantees optimal solution is exahustive search that is infeasible (V^T possibilities)



Beam Search

Instead of choosing the best token to generate at each timestep, we keep *k* possible tokens at each step

- At each step t_i in the sequence till the END token:

1. For each of the *k* best hypotheses so far (i.e. the *k* best generated sequences)
    a. Score the entire vocabulary starting from the current hypothesis by $P(y_i|x, y_{<i})$
2. Prune the $k*V$ scores maintaining the top *k* new hypotheses

1/2

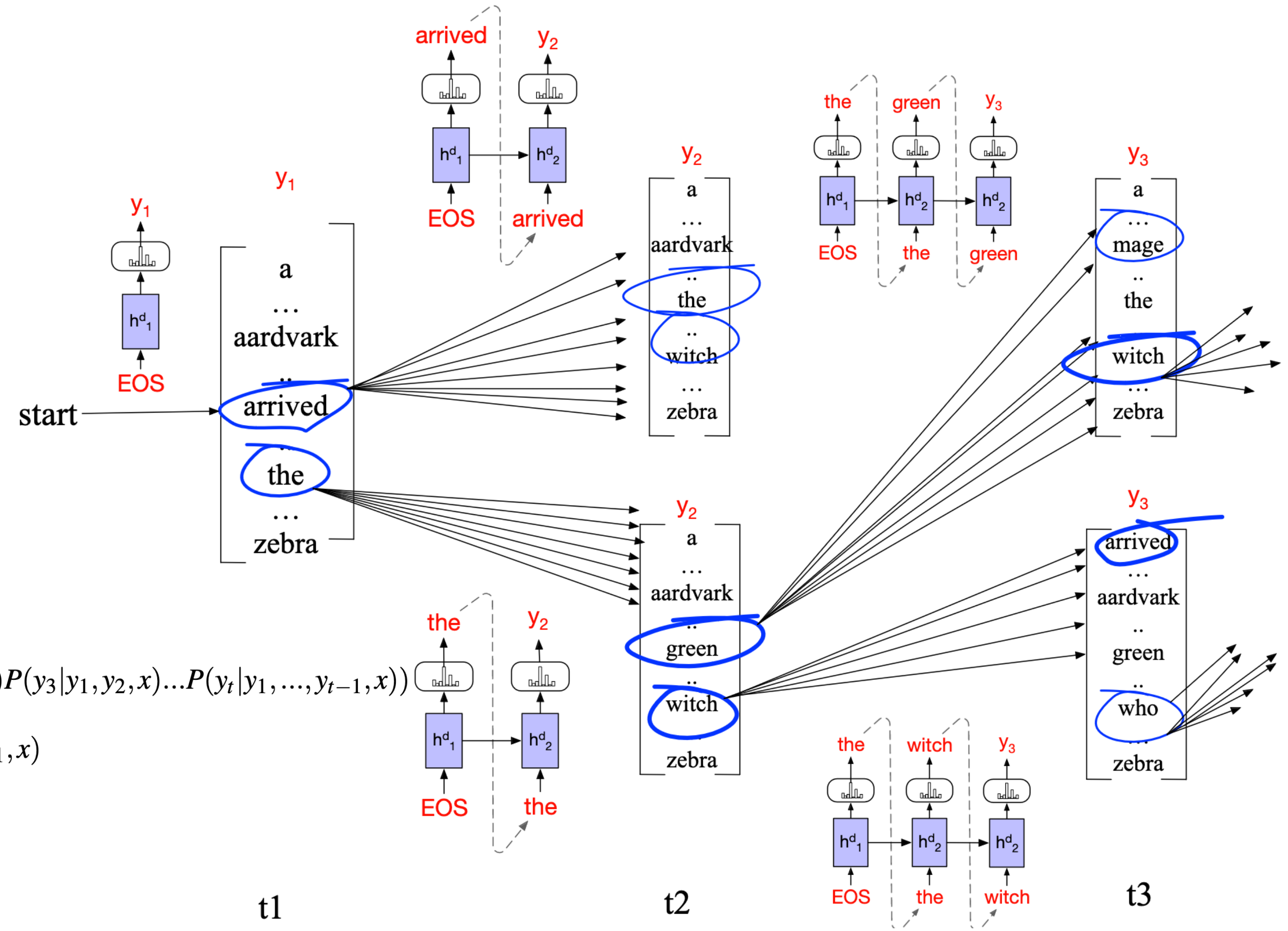
Beam Search

- if the END token is reached:

1. Remove the hypothesis from the beam
2. Reduce $k \rightarrow k - 1$
3. Restart the search with the new beam width k

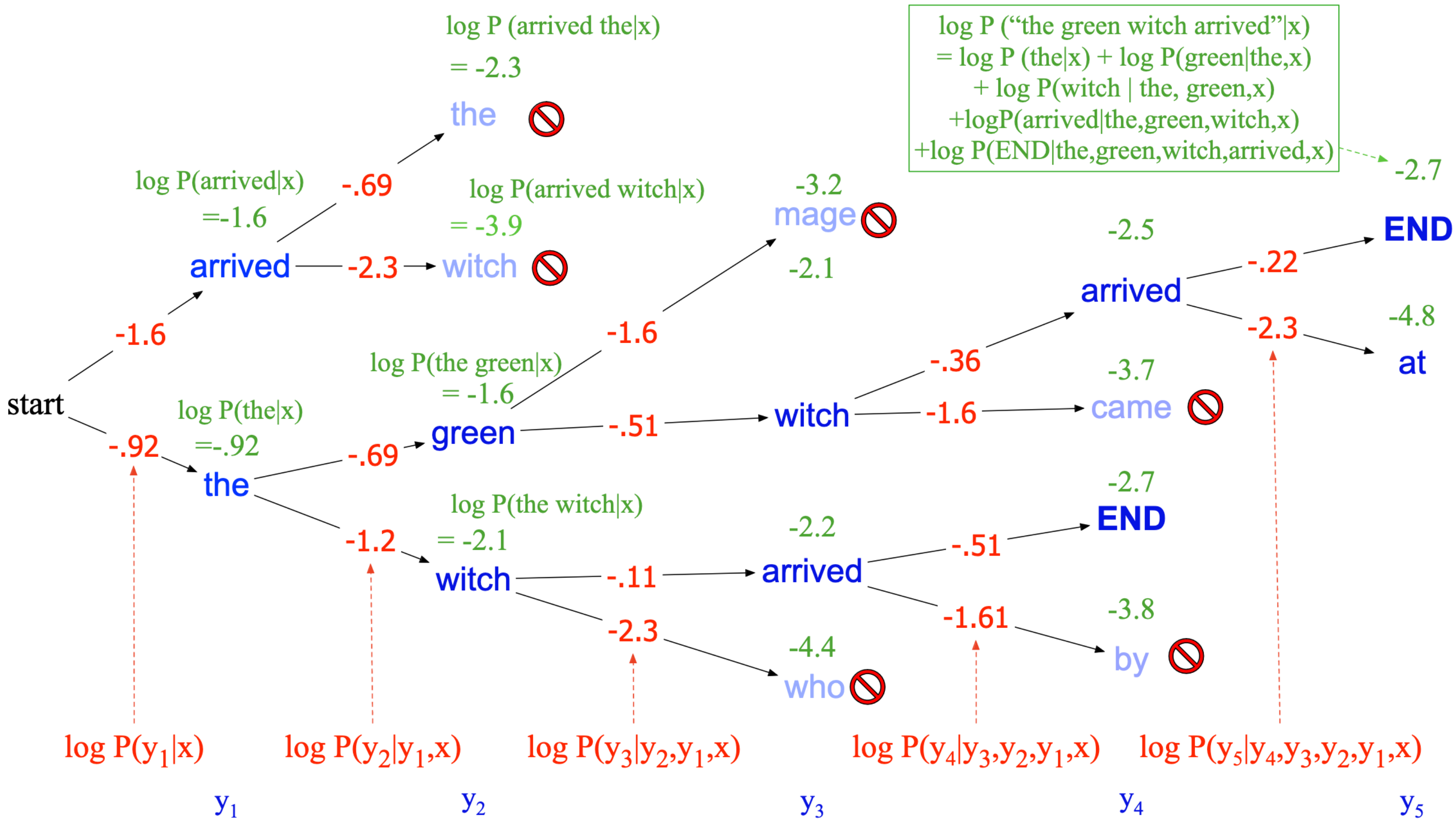
- Continue until $k == 0$
- Return the k hypotheses

Beam Search



$$
\begin{aligned}
score(y) &= \log P(y|x) \\
&= \log\left(P(y_1|x)P(y_2|y_1,x)P(y_3|y_1,y_2,x)\ldots P(y_t|y_1,\ldots,y_{t-1},x)\right) \\
&= \sum_{i=1}^{t} \log P(y_i|y_1,\ldots,y_{i-1},x)
\end{aligned}
$$

Beam Search



log P (arrived the|x)
= -2.3
the
log P(arrived|x)
=-1.6
-.69
log P(arrived witch|x)
= -3.9
arrived
-2.3
witch
-1.6
start
-.92
log P(the|x)
=-.92
the
-.69
green
log P(the green|x)
= -1.6
-1.6
-3.2
mage
-2.1
-.51
witch
-1.2
log P(the witch|x)
= -2.1
witch
-.11
-2.2
arrived
-2.3
-4.4
who
-.36
arrived
-1.6
-3.7
came
-.51
-2.7
END
-1.61
-3.8
by
-2.5
-.22
END
-2.3
-4.8
at
log P ("the green witch arrived"|x)
= log P (the|x) + log P(green|the,x)
+ log P(witch | the, green,x)
+logP(arrived|the,green,witch,x)
+log P(END|the,green,witch,arrived,x)
-2.7
log P(y1|x)
log P(y2|y1,x)
log P(y3|y2,y1,x)
log P(y4|y3,y2,y1,x)
log P(y5|y4,y3,y2,y1,x)
y1
y2
y3
y4
y5

Beam Search

The score has to be normalized for sequences with different length

- The additive nature of the score penalizes long sequences

$$score(y) = -\log P(y|x) \; = \; \frac{1}{T} \sum_{i=1}^{t} -\log P(y_i|y_1, \ldots, y_{i-1}, x)$$

Transformers and Pretrained Language Models

Pretraining and Zero-Shot Learning

Pretraining

Pretraining in AI refers to *training a model with one task to help it form parameters that can be used in other tasks*.

- Inspired by humans
- Previous knowledge is reused and tranferred to perform new tasks
  - Parameters learned before are used to initialize the model to learn a new task
- Read more details at https://bit.ly/45Qc9g2

Pretraining

Pretraining in AI refers to *training a model with one task to help it form parameters that can be used in other tasks*.

- Very huge sata sets are required for pretraining a Large Language Model
- An interesting portal is https://bit.ly/46W58f2

Zero-shot Learning

Zero-shot learning (ZSL) is defined as *learning a classifier on one set of labels and then evaluate on a different set of labels that the classifier has never seen before.*

Recently, especially in NLP, it's been used much more broadly to mean *get a model to do something that it wasn't explicitly trained to do.*

Zero-shot Learning

Large Language Models (LLM) are evaluated on downstream tasks like machine translation *without fine-tuning on these tasks directly*

- Each task is reformulated as a word/sentence prediction
- The key for this reformulation is *prompting*

Prompting

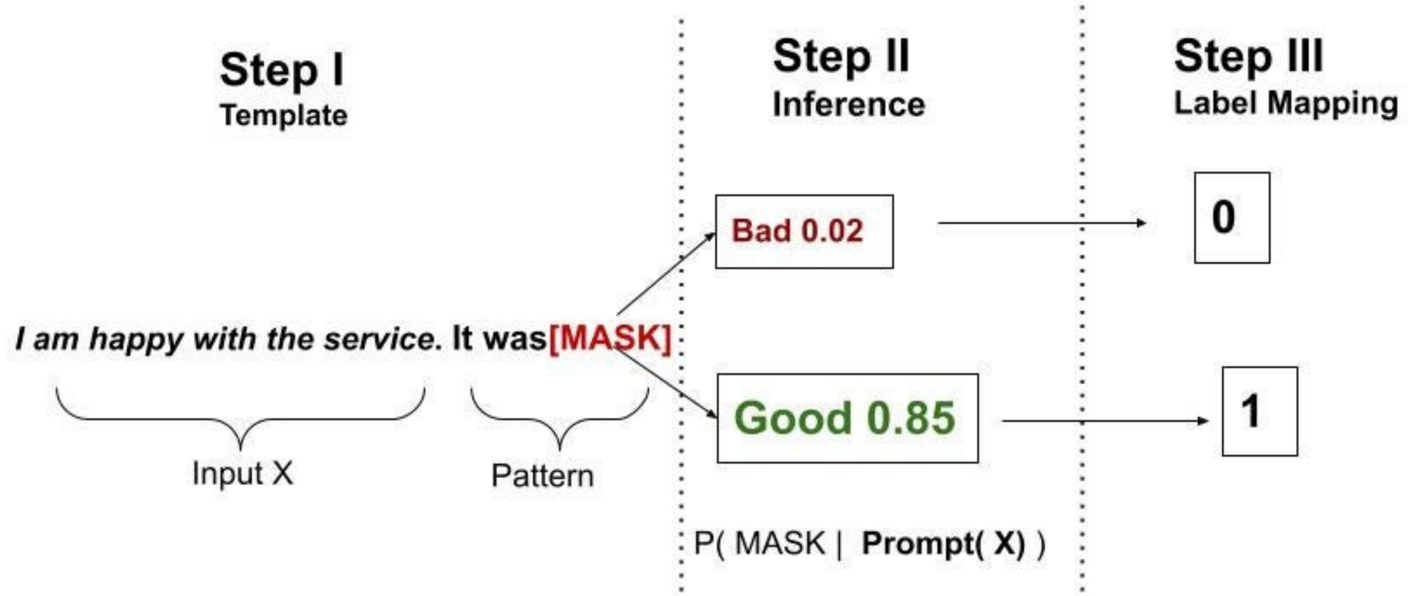
Prompting consists in adding task-specific tokens to the input sequence for conditioning the model

- During its forward pass the model is guideed with interventions to the input rather than passing as it is
- No supervised learning
- Prompts often contain a special `[MASK]` token
  - Typical for Masked Language Models (MLM) like BERT

More details at: https://bit.ly/45Pf7kJ

Prompting

A very simple example for sentiment classification



More details at: https://bit.ly/45Pf7kJ