

Transformer (cont') + Course Project

Wei Xu

(many slides from Greg Durrett)

This Lecture

- ▶ Wrap up with the Transformer model
- ▶ Other applications of Seq2Seq (beyond MT)
- ▶ Decoding in seq2seq models
- ▶ Frontiers in MT Research
- ▶ Final course project

Transformers

Recap: Self-Attention

- Want:

The ballerina is very excited that **she** will dance in the **show**.

A blue curved arrow originates from the word "she" and points to the word "show".

- LSTMs/CNNs: tend to look at local context

The ballerina is very excited that **she** will dance in the **show**.

Two blue curved arrows originate from the word "she" and point to the word "show". A red curved arrow originates from the word "will" and points to the word "show".

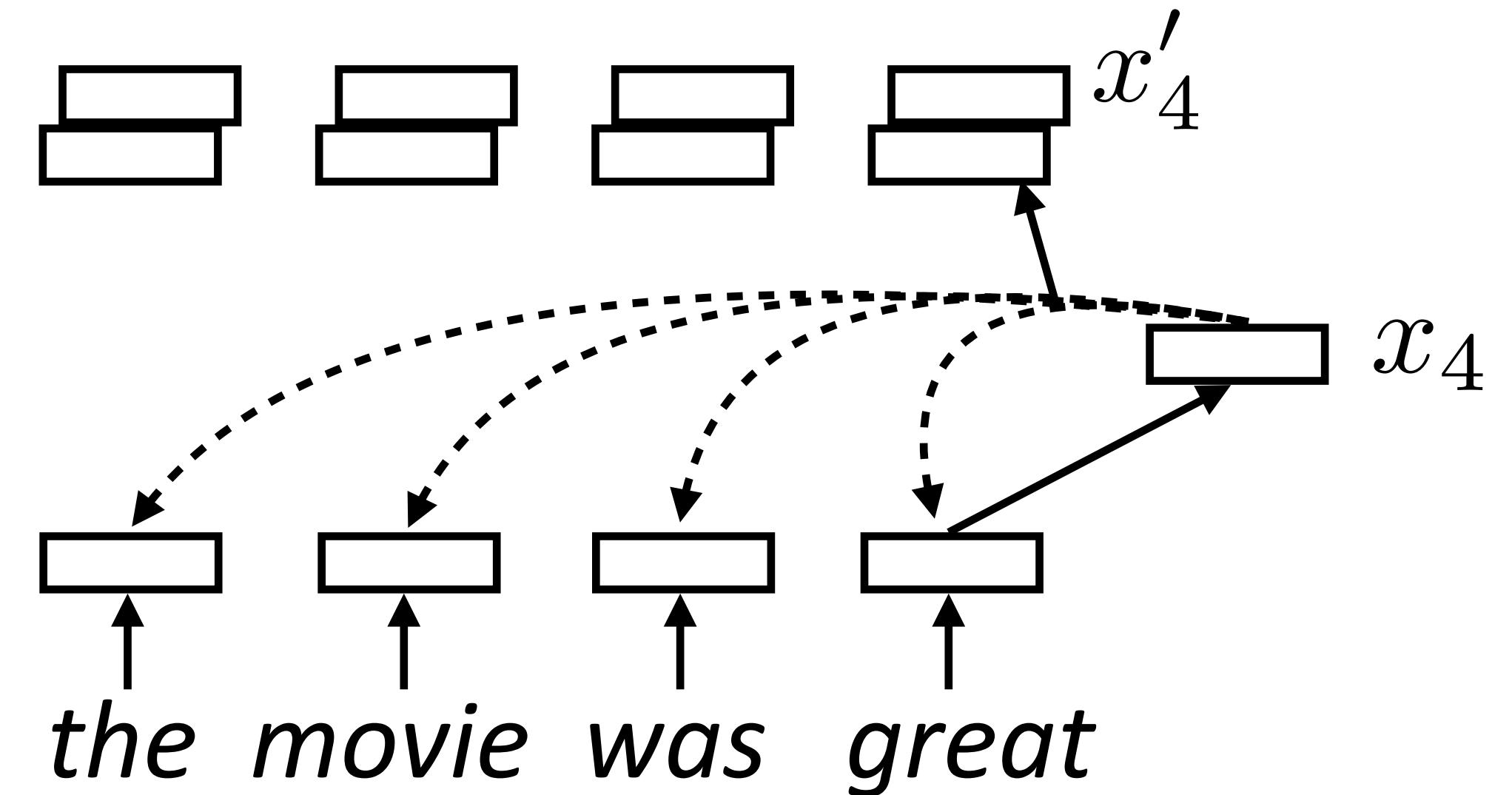
- To appropriately contextualize embeddings, we need to pass information over long distances dynamically for each word

Recap: Self-Attention

- Each word forms a “query” which then computes attention over each word

$$\alpha_{i,j} = \text{softmax}(x_i^\top x_j) \quad \text{scalar}$$

$$x'_i = \sum_{j=1}^n \alpha_{i,j} x_j \quad \text{vector} = \text{sum of scalar} * \text{vector}$$



- Multiple “heads” analogous to different convolutional filters. Use parameters W_k and V_k to get different attention values + transform vectors

$$\alpha_{k,i,j} = \text{softmax}(x_i^\top W_k x_j) \quad x'_{k,i} = \sum_{j=1}^n \alpha_{k,i,j} V_k x_j$$

Recap: What can self-attention do?



*The ballerina is very excited that **she** will dance in the **show**.*

0	0.5	0	0	0.1	0.1	0	0.1	0.2	0	0	0
---	-----	---	---	-----	-----	---	-----	-----	---	---	---

0	0.1	0	0	0	0	0	0	0.5	0	0.4	0
---	-----	---	---	---	---	---	---	-----	---	-----	---

- ▶ Attend nearby + to semantically related terms
- ▶ This is a demonstration, we will revisit what these models actually learn when we discuss BERT
- ▶ Why multiple heads? Softmaxes end up being peaked, single distribution cannot easily put weight on multiple things

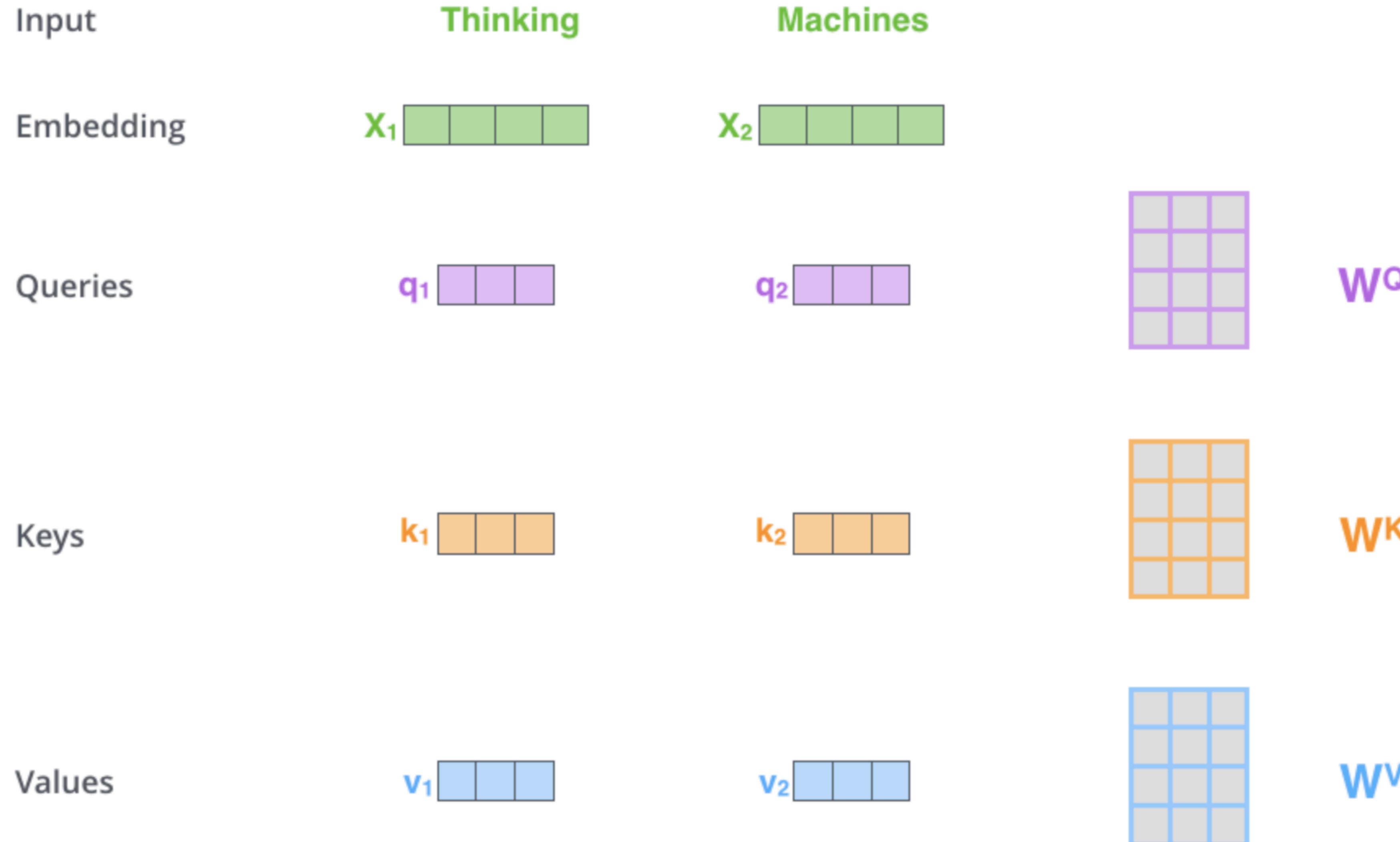
Recap: Multi-Head Self Attention

- ▶ Multiple “heads” analogous to different convolutional filters
- ▶ Let $X = [\text{sent len}, \text{embedding dim}]$ be the input sentence
- ▶ Query $Q = W^Q X$: these are like the **decoder hidden state** in attention
- ▶ Keys $K = W^K X$: these control what gets attended to, along with the query
- ▶ Values $V = W^V X$: these vectors get summed up to form the output

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

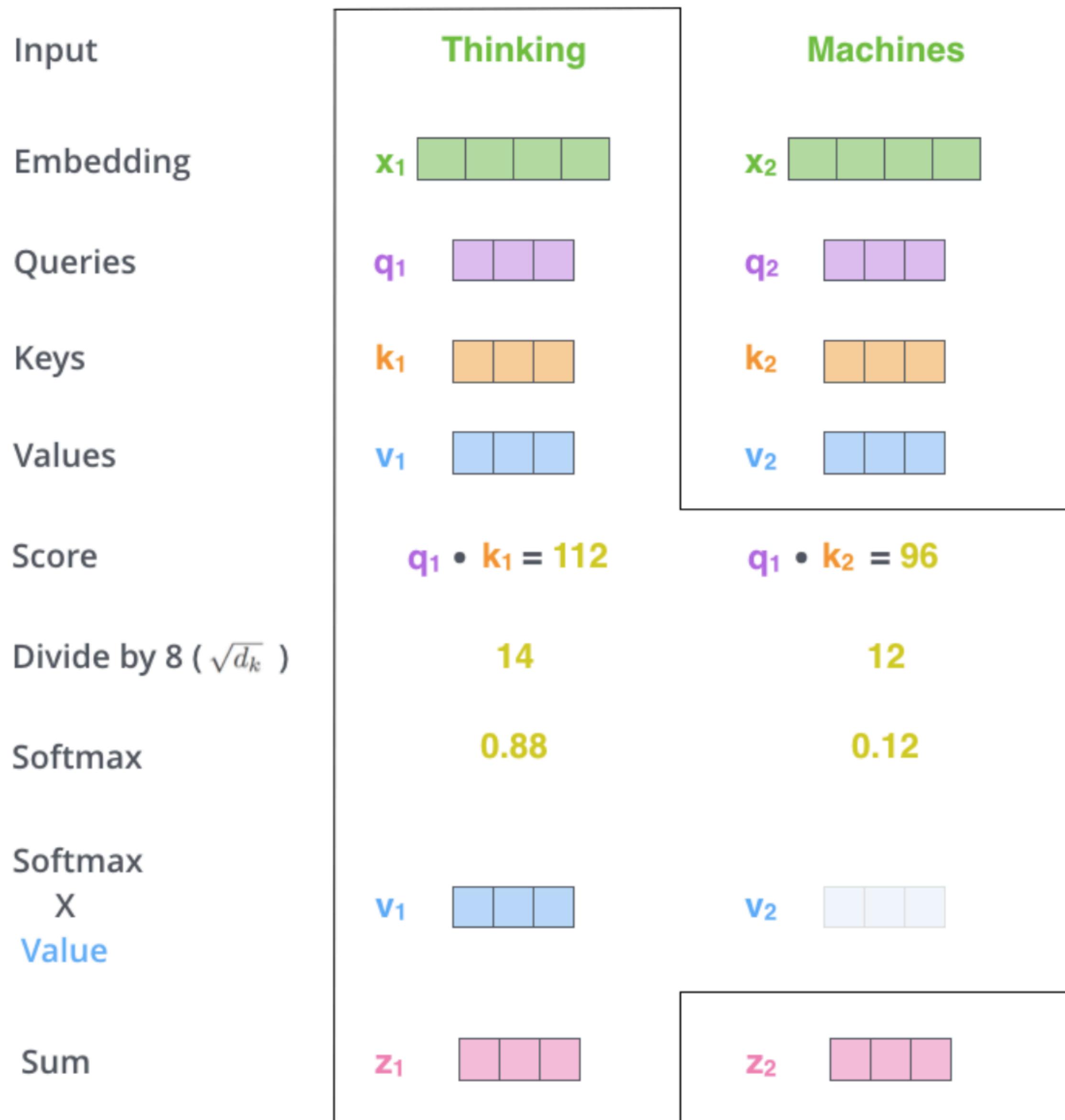
dim of keys

Recap: Single-Head Self Attention



Credit: Alammar, *The Illustrated Transformer*

Recap: Single-Head Self Attention



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

Credit: Alammar, *The Illustrated Transformer*

Recap: Single-Head Self Attention

every row in X is a word in input sent

$$\begin{matrix} \mathbf{x} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \mathbf{w}_Q \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \mathbf{Q} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} \mathbf{x} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \mathbf{w}_K \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \mathbf{K} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} \mathbf{x} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \mathbf{w}_V \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \mathbf{V} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} \end{matrix}$$

sent len x sent len (attn for each word to each other)

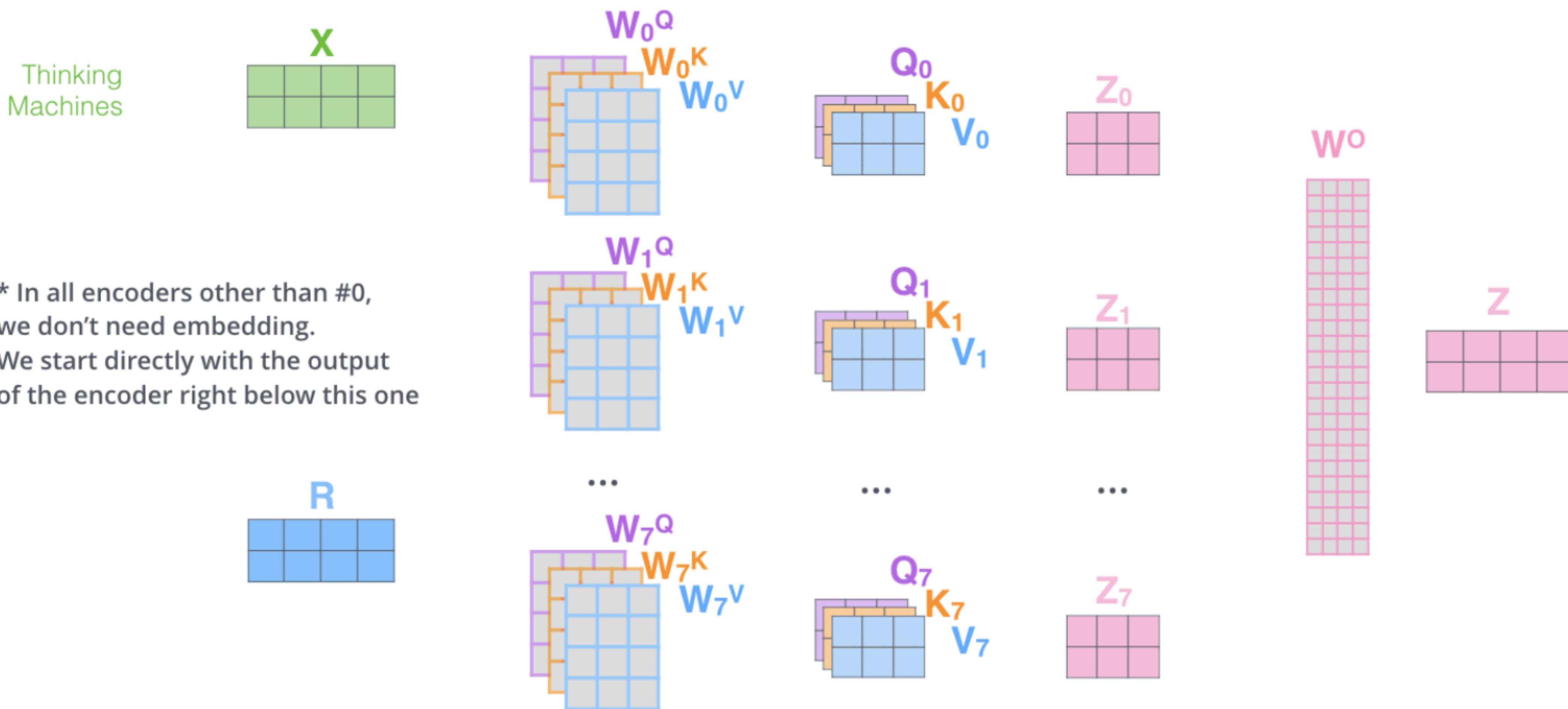
$$\text{softmax} \left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}} \right) = \mathbf{Z}$$

sent len x hidden dim

Z is a weighted combination of V rows

Recap: Multi-Head Self Attention

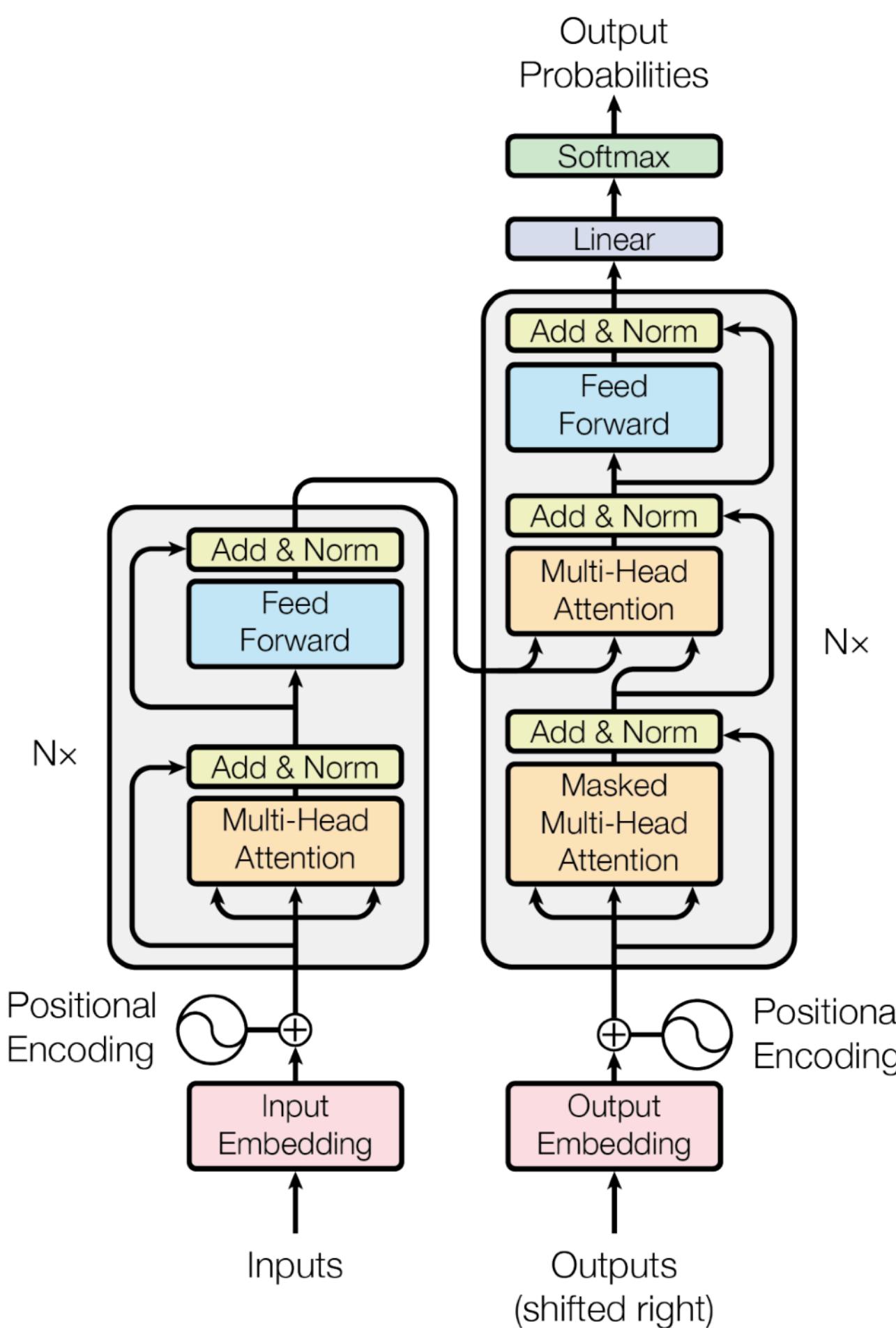
- 1) This is our input sentence* X
- 2) We embed each word* R
- 3) Split into 8 heads. We multiply X or R with weight matrices W_0^Q, W_0^K, W_0^V
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



Credit: Alammar, *The Illustrated Transformer*

Transformer Uses

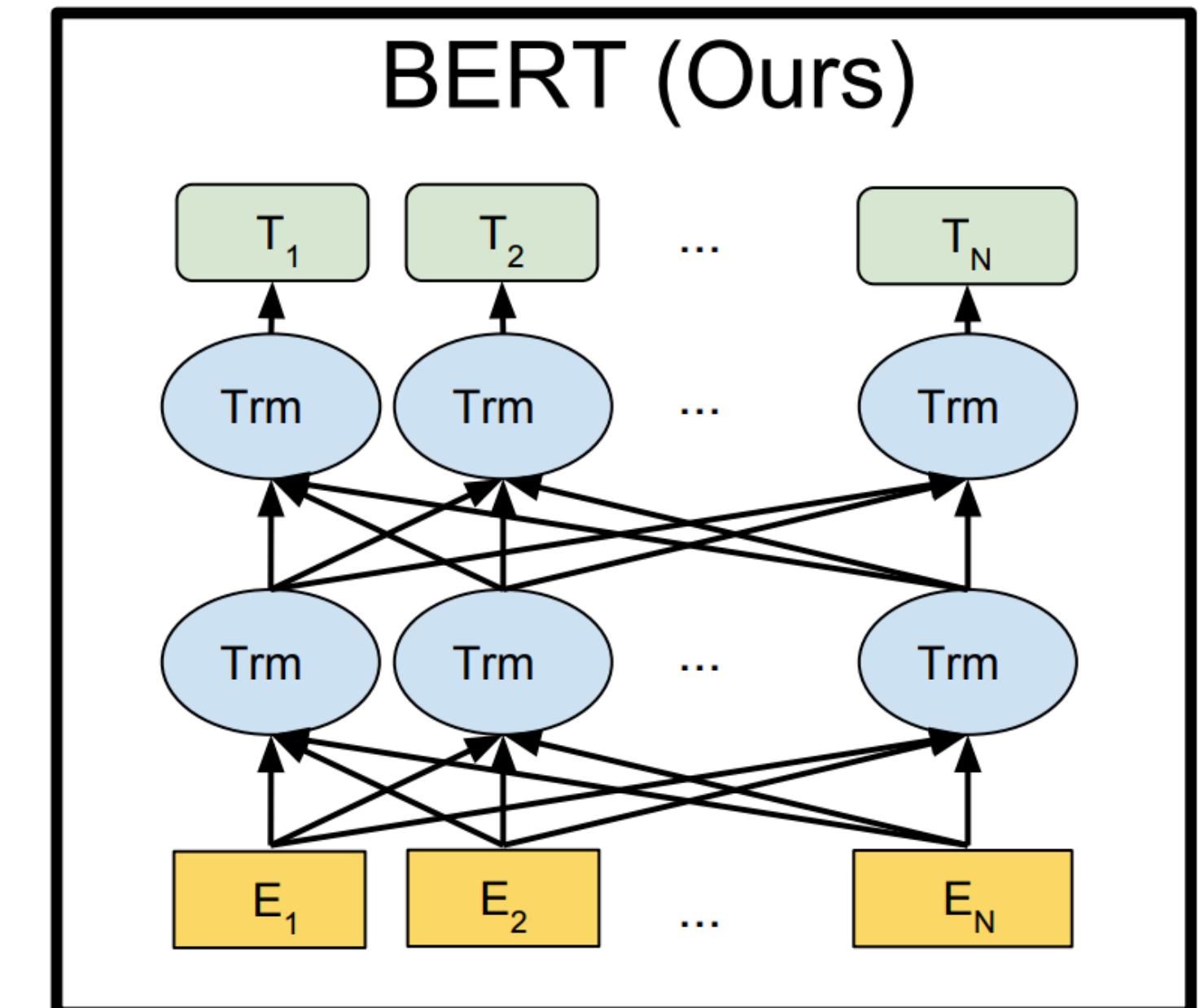
- ▶ Supervised: transformer can replace LSTM as encoder, decoder, or both; such as in machine translation and natural language generation tasks.



- ▶ Encoder and decoder are both transformers
- ▶ Decoder consumes the previous generated token (and attends to input), but has *no recurrent state*
- ▶ Many other details to get it to work: residual connections, layer normalization, positional encoding, optimizer with learning rate schedule, label smoothing

Transformer Uses

- ▶ Unsupervised: transformers work better than LSTM for unsupervised pre-training of embeddings — predict word given context words
- ▶ BERT (Bidirectional Encoder Representations from Transformers): pretraining transformer language models similar to ELMo (based on LSTM)
- ▶ Stronger than similar methods, SOTA on ~11 tasks (including NER — 92.8 F1)

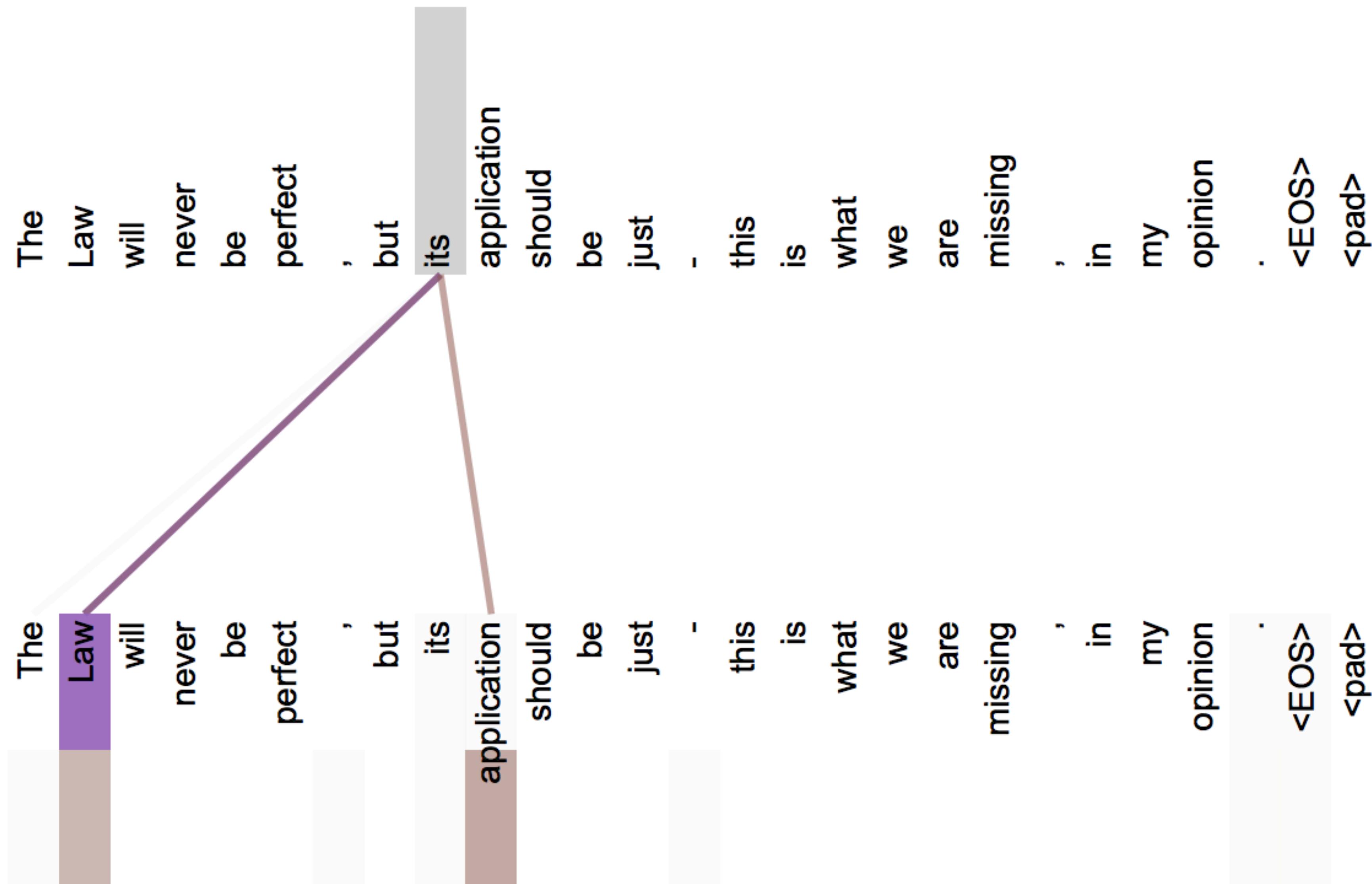


Visualization

It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS>

It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS>

Visualization



Visualization

The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>

The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>

Useful Resources

nn.Transformer:

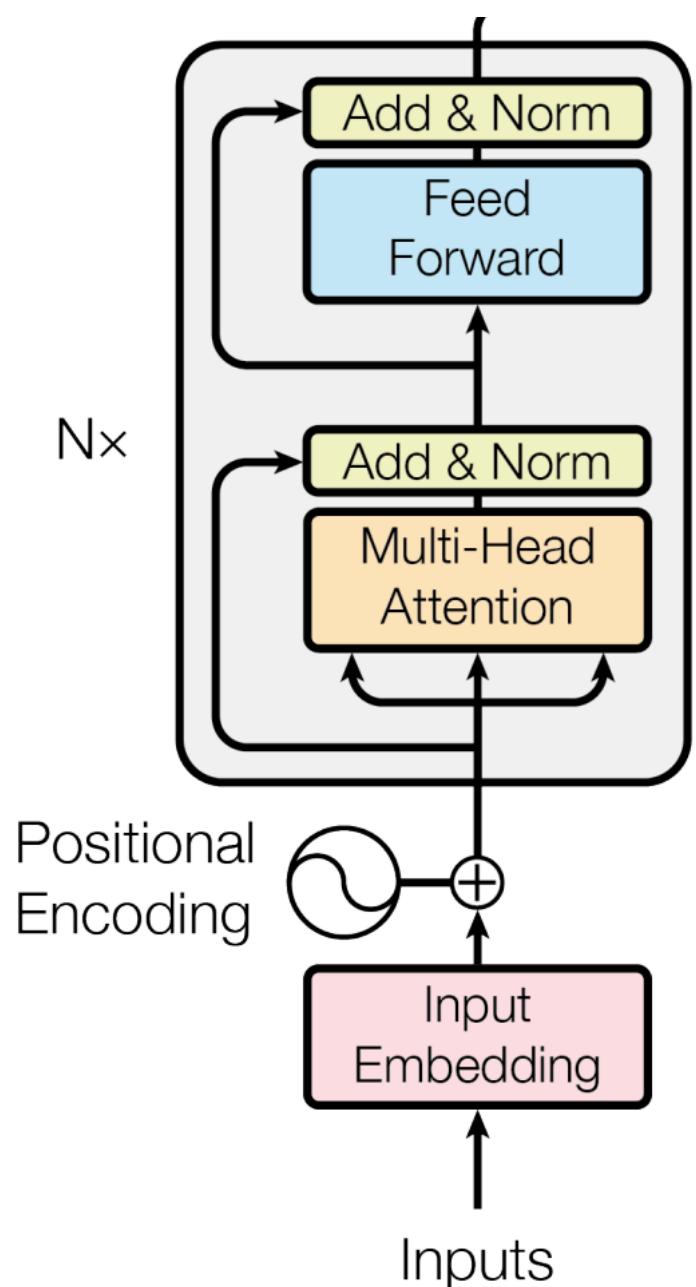
```
>>> transformer_model = nn.Transformer(nhead=16, num_encoder_layers=12)
>>> src = torch.rand((10, 32, 512))
>>> tgt = torch.rand((20, 32, 512))
>>> out = transformer_model(src, tgt)
```

nn.TransformerEncoder:

```
>>> encoder_layer = nn.TransformerEncoderLayer(d_model=512, nhead=8)
>>> transformer_encoder = nn.TransformerEncoder(encoder_layer, num_layers=6)
>>> src = torch.rand(10, 32, 512)
>>> out = transformer_encoder(src)
```

Other Transformer Variations

- ▶ Multilayer transformer networks consist of interleaved self-attention and feedforward sublayers.
- ▶ Could ordering the sublayers in a different pattern lead to better performance?



sfs

(a) Interleaved Transformer

sssssssfssfssfssfssfssffffffff

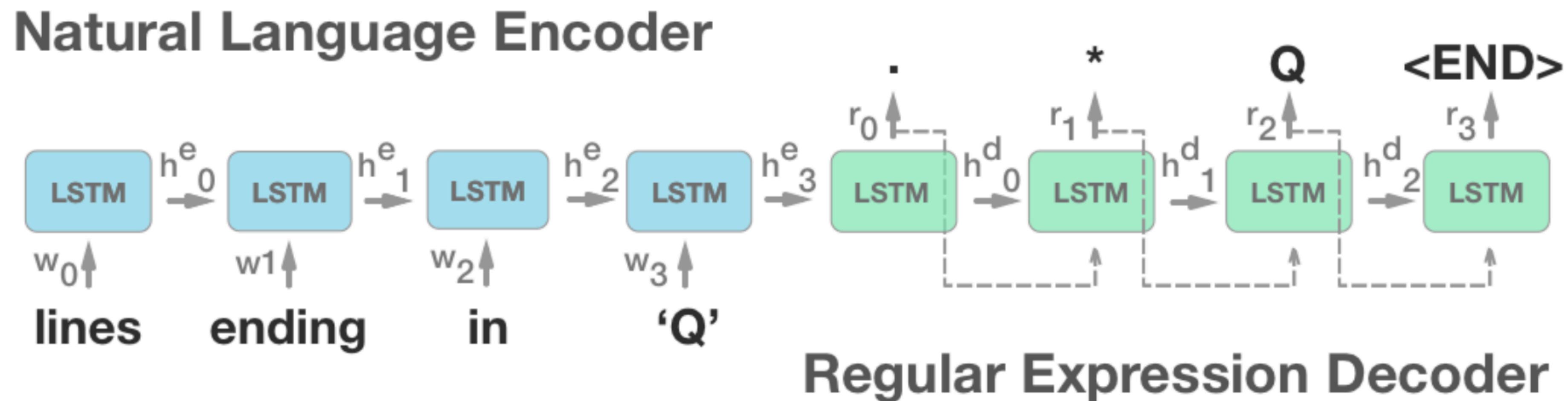
(b) Sandwich Transformer

Figure 1: A transformer model (a) is composed of interleaved self-attention (green) and feedforward (purple) sublayers. Our sandwich transformer (b), a reordering of the transformer sublayers, performs better on language modeling. Input flows from left to right.

Other Applications of Seq2Seq

Regex Prediction

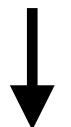
- ▶ Seq2seq models can be used for many other tasks!
- ▶ Predict regex from text



- ▶ Problem: requires a lot of data: 10,000 examples needed to get ~60% accuracy on pretty simple regexes

Semantic Parsing as Translation

“what states border Texas”



$\lambda \ x \ \text{state}(\ x \) \wedge \text{borders}(\ x \ , \ \text{e89} \)$

- ▶ Write down a linearized form of the semantic parse, train seq2seq models to directly translate into this representation
- ▶ No need to have an explicit grammar, simplifies algorithms
- ▶ Might not produce well-formed logical forms, might require lots of data

SQL Generation

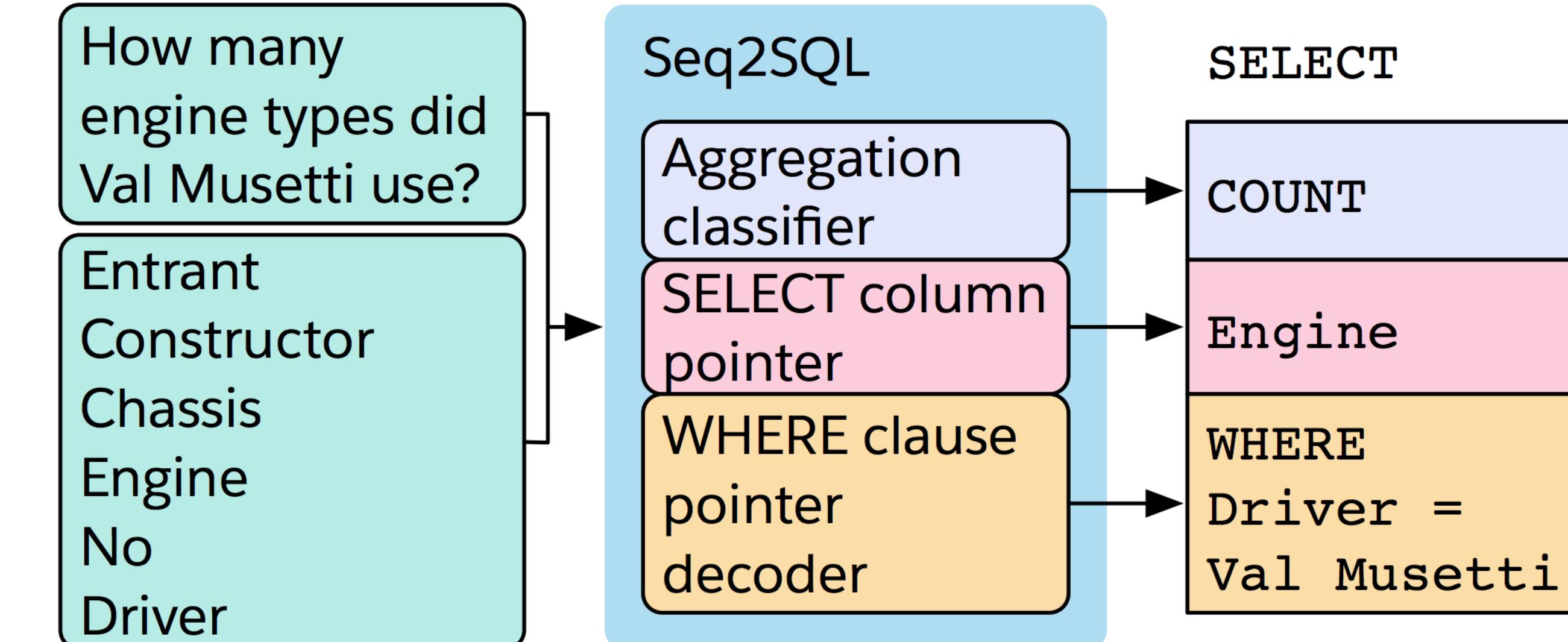
- ▶ Convert natural language description into a SQL query against some DB
- ▶ How to ensure that well-formed SQL is generated?
 - ▶ Three components
 - ▶ How to capture column names + constants?
 - ▶ Pointer mechanisms

Question:

How many CFL teams are from York College?

SQL:

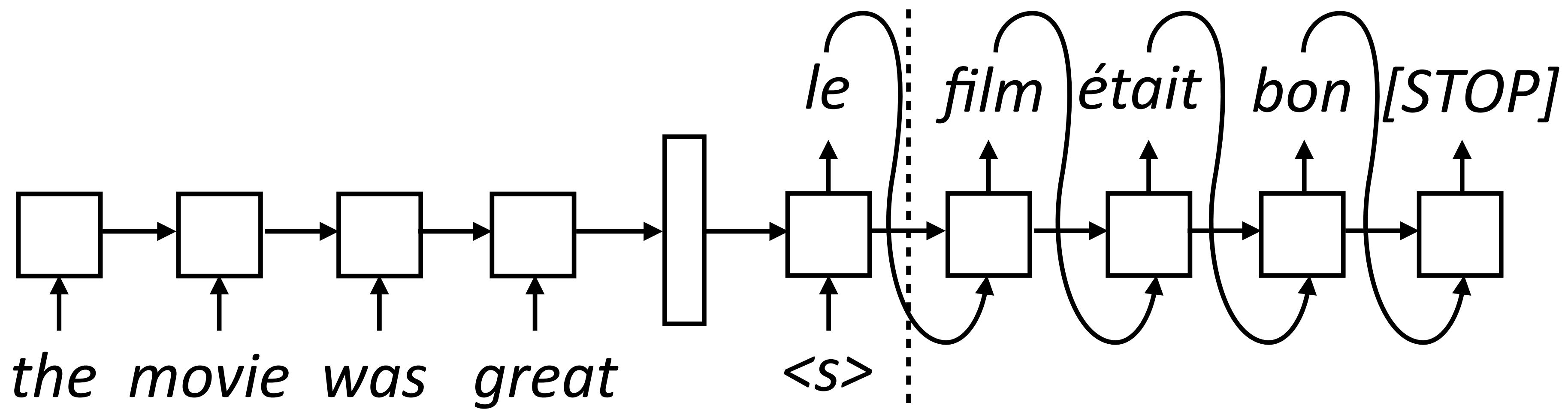
```
SELECT COUNT CFL Team FROM  
CFLDraft WHERE College = "York"
```



Decoding Strategies

Greedy Decoding

- ▶ Generate next word conditioned on previous word as well as hidden state



- ▶ During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state. This is **greedy decoding**

$$P(y_i | \mathbf{x}, y_1, \dots, y_{i-1}) = \text{softmax}(W\bar{h}) \quad (\text{or attention/copying/etc.})$$

$$y_{\text{pred}} = \text{argmax}_y P(y | \mathbf{x}, y_1, \dots, y_{i-1})$$

Problems with Greedy Decoding

- ▶ Only returns one solution, and it may not be optimal
- ▶ Can address this with **beam search**, which usually works better...but even beam search may not find the correct answer! (max probability sequence)

Model	Beam-10	
	BLEU	#Search err.
LSTM*	28.6	58.4%
SliceNet*	28.8	46.0%
Transformer-Base	30.3	57.7%
Transformer-Big*	31.7	32.1%

A sentence is classified as search error if the decoder does not find the global best model score.

“Problems” with Beam Decoding

- ▶ For machine translation, the highest probability sequence is often the empty string, i.e.. a single </s> token! (>50% of the time)

Search	BLEU	Ratio	#Search errors	#Empty
Greedy	29.3	1.02	73.6%	0.0%
Beam-10	30.3	1.00	57.7%	0.0%
Exact	2.1	0.06	0.0%	51.8%

- ▶ Beam search results in *fortuitous search errors* that avoid these bad solutions
- ▶ Exact inference uses depth-first search, but cut off branches that fall below a lower bound.

Sampling

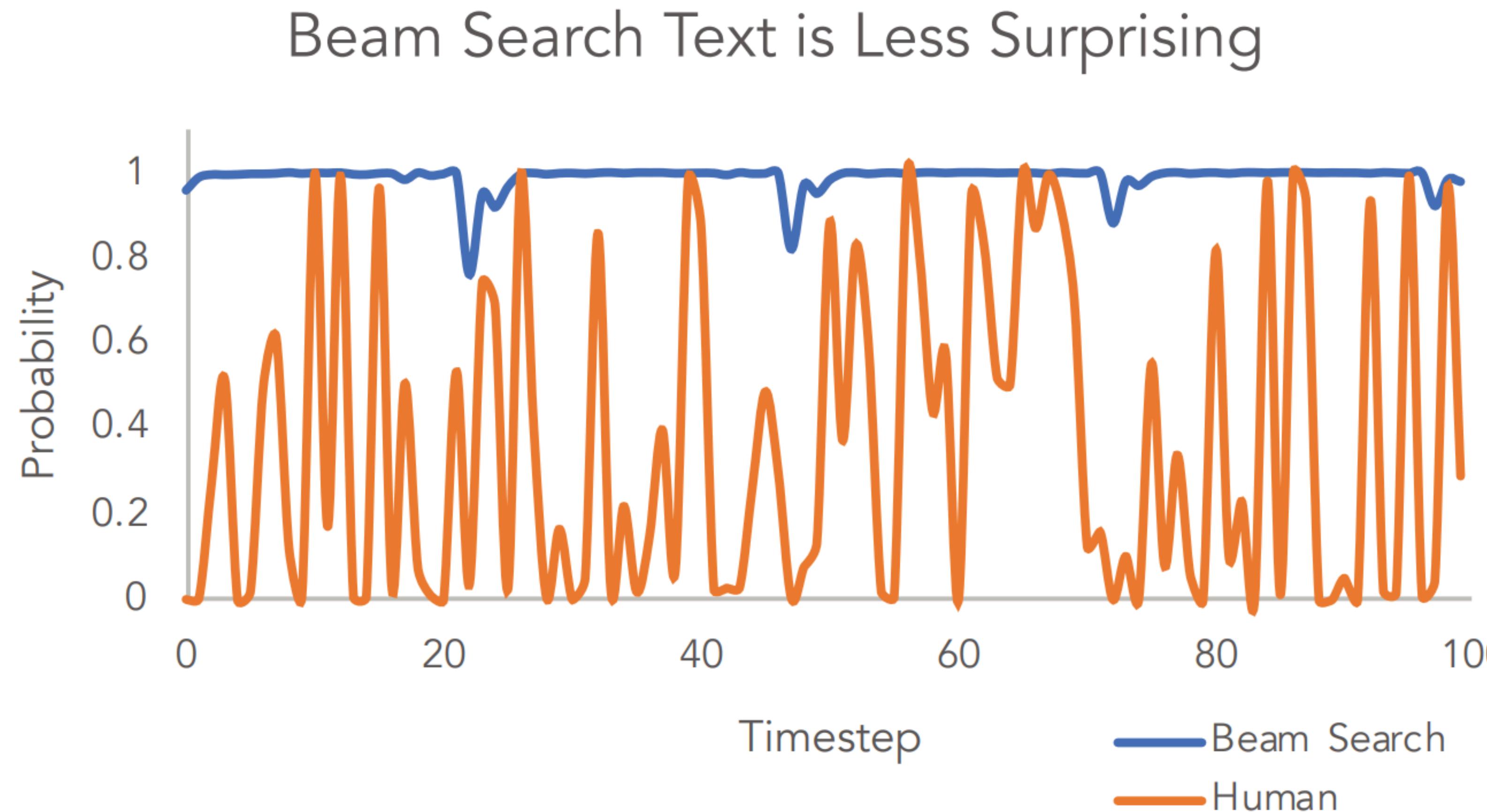
- ▶ Beam search may give many similar sequences, and these actually may be *too close* to the optimal. Can sample instead:

$$P(y_i | \mathbf{x}, y_1, \dots, y_{i-1}) = \text{softmax}(W\bar{h})$$

$$y_{\text{sampled}} \sim P(y | \mathbf{x}, y_1, \dots, y_{i-1})$$

- ▶ Text *degeneration*: greedy solution can be uninteresting / vacuous for various reasons. Sampling can help.

Beam Search vs. Sampling



Decoding Strategies

- ▶ Greedy
- ▶ Beam search
- ▶ Sampling (e.g., top-k or Nucleus sampling)
 - ▶ Top-k: take the top k most likely words ($k=5$), sample from those
 - ▶ Nucleus: take the top p% (95%) of the distribution, sample from within that

Beam Search vs. Sampling

- ▶ These are samples from an unconditioned language model (not seq2seq model)

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Beam Search, $b=32$:

"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ...")

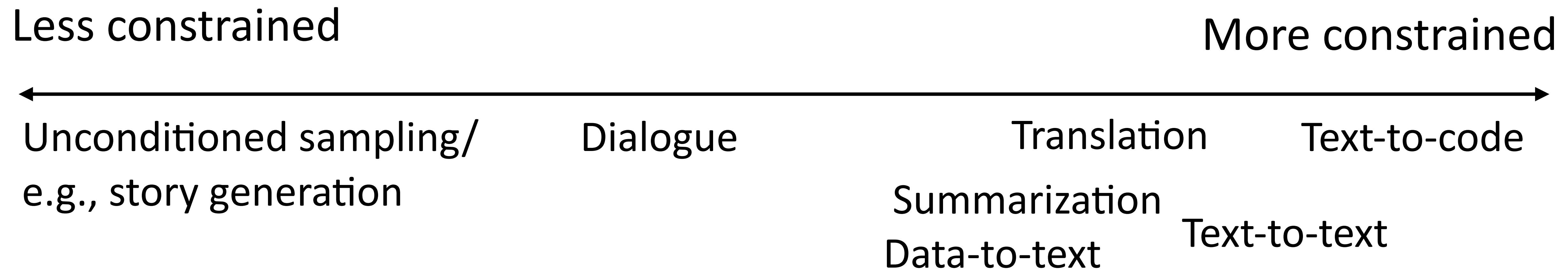
Pure Sampling:

They were cattle called **Bolivian Cavalleros**; they live in a remote desert **uninterrupted by town**, and they speak **huge, beautiful, paradisiacal Bolivian linguistic thing**. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

- ▶ Sampling is better but sometimes draws too far from the tail of the distribution

Generation Tasks

- ▶ There are a range of seq2seq modeling tasks we will address
- ▶ For more constrained problems: greedy/beam decoding are usually best
- ▶ For less constrained problems: nucleus sampling introduces favorable variation in the output



Final Project

Final Project

- ▶ **Groups Size:** 2-4 people; 1 is possible (email me for permission).
 - ▶ **Submission:** up to 4 page report (including everything) + final presentation.
 - ▶ **Prize:** We will give out 1-3 best project awards. 
-
- ▶ **Shared project** with other classes is allowed
 - ▶ project is expected to be accordingly bigger/better
 - ▶ clearly declare at the beginning of your report that you are sharing project (with which class)
 - ▶ **External collaborators** (non CS4650 students) are also allowed
 - ▶ clearly describe in the report which parts of the projects are your work

Finding Research Topics

- ▶ Two basic starting points, for all of science:
 - ▶ **Nails** — start with a (domain) problem of interest and try to find good/better ways to address it than are currently known/used
 - ▶ **Hammers** — start with a technical method/approach of interest, and work out good ways to extend or improve it or new ways to apply it

Typical Project Types

- ▶ This is not an exhaustive list —
- ▶ 1) Find an application/task of interest and explore how to approach/solve it effectively, often with an existing model
 - ▶ Could be task in the wild or some existing Kaggle competition or shared task (e.g.. WNUT or SemEval, etc.)
 - ▶ Or dialogue system (prepare for Amazon Alexa Challenges next year)
- ▶ 2) Analyze the behavior of models or existing datasets
 - ▶ how the model represents linguistic knowledge or what kinds of phenomena it can handle or errors that it makes.
 - ▶ what linguistic phenomena/errors exist in the dataset, how they affect model performance (see Idea #3 for an example).

Typical Project Types

- ▶ This is not an exhaustive list —
- ▶ 3) Create a new dataset, conduct some analysis, train a prediction model
 - ▶ for a new topic/task (see Idea #1 and #2 for an example), or for an existing task but better way to create higher quality dataset
 - ▶ may involve some manual annotation
 - ▶ conduct some quantitative and linguistic analyses
- ▶ 4) Implement a complex neural architecture and demonstrate its performance on some data, especially for non-English data
- ▶ 5) Come up with a new or variant neural network model and explore its empirical success (but this has become harder since 2020 —)

Place to start?

- ▶ Look at ACL Anthology for NLP papers:
 - ▶ <https://aclanthology.org/>
- ▶ Also look at the online proceedings of major ML/Web conferences
 - ▶ ICLR, NeurIPS , ICML
 - ▶ ICWSM (<https://www.icwsm.org/2021/>)
- ▶ Look at online preprint servers, especially:
 - ▶ <https://arxiv.org/>
- ▶ Look for an interesting problem in the world!
 - ▶ Psycholinguistics (e.g., Idea #1), computational social science, journalism, ...

Finding Data

- ▶ Some people collect their own data for a project — **we like that!**
 - ▶ You may have a project that uses “unsupervised” data
 - ▶ You can annotate a small amount of data
 - ▶ You can find a website that effectively provides annotations, such as likes, starts, rating, responses, etc.
 - ▶ Look at research papers to see what data they use, how they get it
- ▶ Many others make use of existing datasets built by other researchers
 - ▶ Shared task at WNUT, SemEval, etc.
 - ▶ Kaggle competition
 - ▶ Datasets used in other papers (e.g. <https://aclanthology.org/>)

An Example

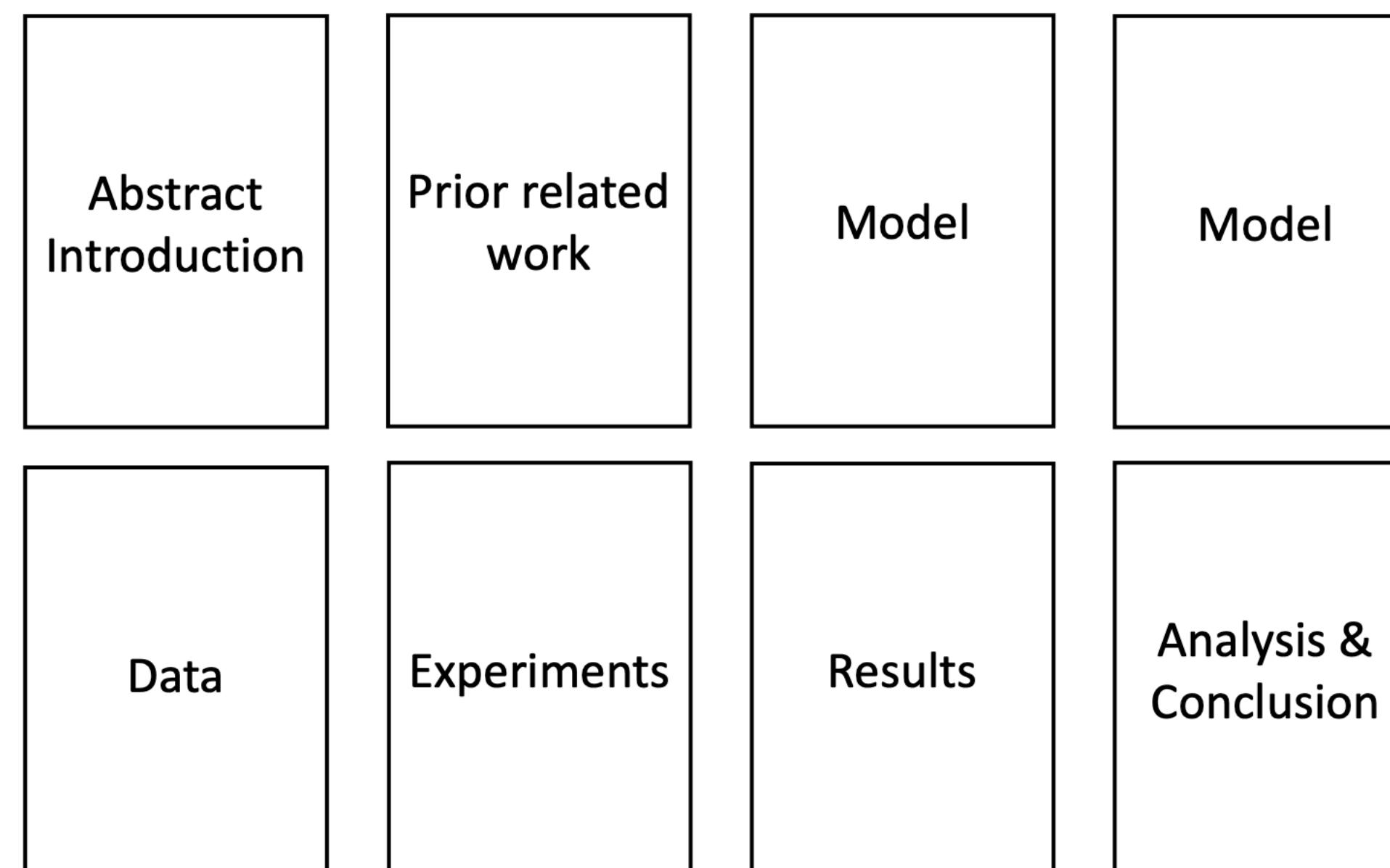
- ▶ Define Task
- ▶ Define Dataset
 - ▶ Provide basic data statistics
 - ▶ If your own data —
 - ▶ steps you take to collect/clean/annotate the data
 - ▶ provide some examples, quality control (this is important!)

An Example

- ▶ Experiments
 - ▶ right from the beginning, separate off train/dev/test splits
 - ▶ search online for well-established metrics on this task
 - ▶ establish some baselines
 - ▶ Implement existing neural network model
 - ▶ compute metrics on train & dev, not test set
 - ▶ analyze outputs and errors
- ▶ Going beyond – try out different models, increasing quality/quantitive of your dataset, data argumentation, and other “researchy” ideas!

Final Project Writeup/Presentation

- ▶ Up to **4-page writeup** due the day before final exam date (no late submission!)
- ▶ Use **LaTeX template** from ACL
- ▶ Include references; statement of each group members' contribution
- ▶ Writeup quality is important to your grade!
- ▶ X-minute **oral presentation at the final exam time** ($X \in [5, 10]$)



Frontiers in MT

Low-Resource MT

- ▶ Particular interest in deploying MT systems for languages with little or no parallel data
- ▶ BPE allows us to transfer models even without training on a specific language
- ▶ Pre-trained models can help further

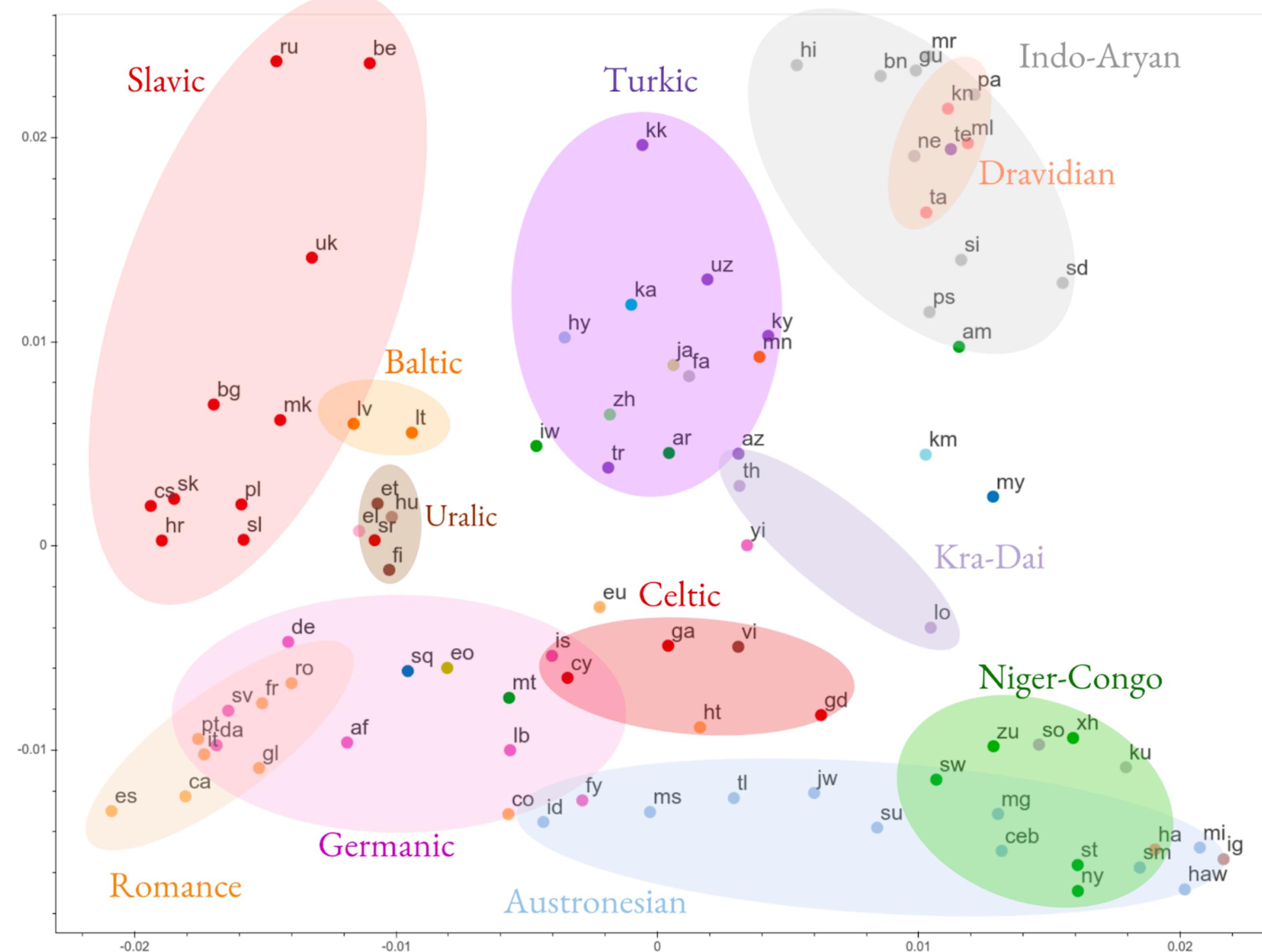
Burmese, Indonesian, Turkish
BLEU

Transfer	My→En	Id→En	Tr→En
baseline (no transfer)	4.0	20.6	19.0
transfer, train	17.8	27.4	20.3
transfer, train, reset emb, train	13.3	25.0	20.0
transfer, train, reset inner, train	3.6	18.0	19.1

Table 3: Investigating the model's capability to restore its quality if we reset the parameters. We use En→De as the parent.

Massively Multilingual MT

- ▶ For 103 languages



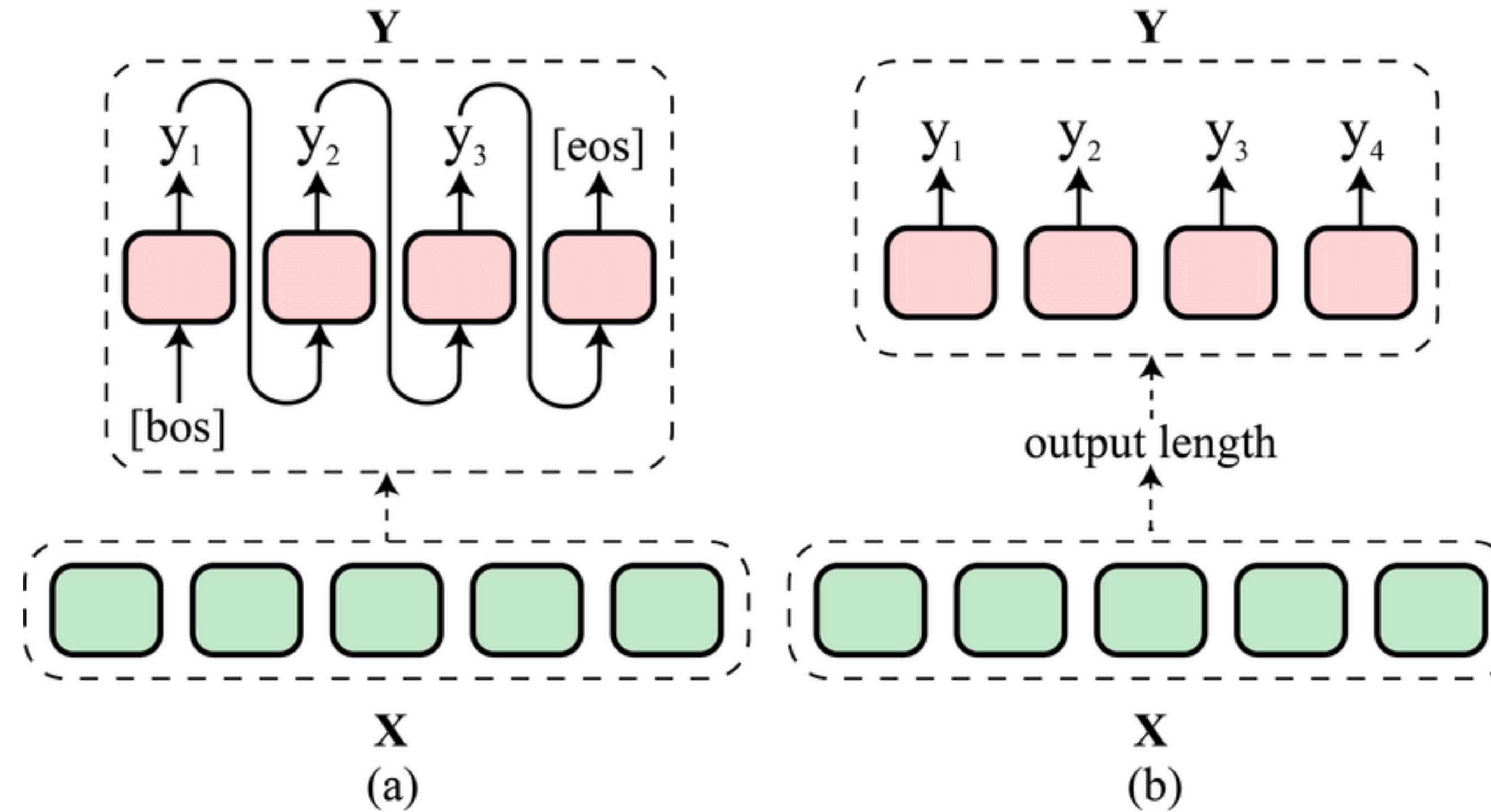
Arivazhagan et al. (2019), Kudugunta et al. (2019)

Unsupervised MT

Approach	Train/Val	Test	Loss
Supervised MT	L1-L2	L1-L2	$\mathcal{L}_{x \rightarrow y}^{MT} = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim (\mathcal{X}, \mathcal{Y})} [-\log p_{x \rightarrow y}(\mathbf{y} \mathbf{x})]$
Unsupervised MT	L1, L2	L1-L2	$\mathcal{L}_{x \leftrightarrow y}^{BT} = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} [-\log p_{y \rightarrow x}(\mathbf{x} g^*(\mathbf{x}))] + \mathbb{E}_{\mathbf{y} \sim \mathcal{Y}} [-\log p_{x \rightarrow y}(\mathbf{y} h^*(\mathbf{y}))]$ <p>g^*, h^*: sentence predictors</p>

- ▶ Common principles of unsupervised MT
 - ▶ Language models
 - ▶ (Iterative) Back-translation!

Non-Autoregressive NMT



- ▶ Q: why non-autoregressive? Pros and cons?

Efficiency of NMT

SIXTH CONFERENCE ON MACHINE TRANSLATION (WMT21)

November 10-11, 2021
Punta Cana (Dominican Republic) and Online

Shared Task: Efficiency

[[HOME](#)] [[SCHEDULE](#)] [[PAPERS](#)] [[AUTHORS](#)] [[RESULTS](#)]

TRANSLATION TASKS: [[NEWS](#)] [[SIMILAR LANGUAGES](#)] [[BIOMEDICAL](#)] [[EUROPEAN LOW RES MULTILINGUAL](#)] [[LARGE-SCALE MULTILINGUAL](#)]

[[TRIANGULAR MT](#)]

[[EFFICIENCY](#)] [[TERMINOLOGY](#)] [[UNSUP AND VERY LOW RES](#)] [[LIFELONG LEARNING](#)]

EVALUATION TASKS: [[QUALITY ESTIMATION](#)] [[METRICS](#)]

OTHER TASKS: [[AUTOMATIC POST-EDITING](#)]

Efficiency Task

The efficiency task measures latency, throughput, memory consumption, and size of machine translation on CPUs and GPUs. Participants provide their own code and models using standardized data and hardware. This is a continuation of the [WNLT 2020 Efficiency Shared Task](#).

Have fun with your project!