
Sequence Prediction and Representation

David I. Inouye

Dynamic Content · Last Updated: October 23, 2026

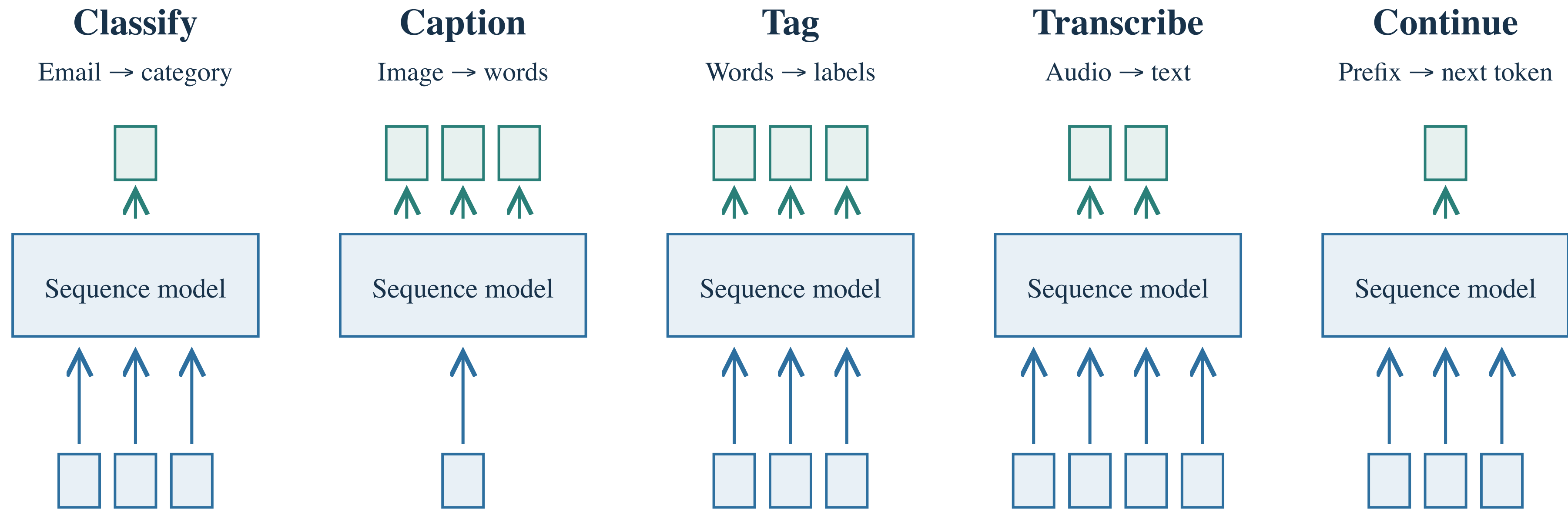
| A completion assistant needs a **task, examples and a way to use context**

Arc 6: Sequence-model architectures

1. **Define the task:** what is given, and what should come next?
2. **Build training examples:** tokens, learned features and shifted targets.
3. **Choose the context model:** fixed window, recurrent state, then attention.

We will follow one email-completion example through each choice.

| Sequence tasks differ in **what goes in and what comes out**



Inputs below → model → outputs above. Each small box is one item.

Our focus: next-token completion. Repeating it generates a continuation.

Original illustration informed by Karpathy's sequence-task examples.

| Email autocomplete suggests **the next few words while you write**

Typed prefix: “please send the”

Possible continuations: “report” · “slides” · “agenda”

- The writer supplies the message and accepts or ignores a suggestion.
- We predict the next token, then repeat to suggest a short continuation.
- Suggestions should fit the prefix and arrive quickly.

First: turn text into choices the model can predict.

Real application: [Gmail Smart Compose \(Chen et al., 2019\)](#). The email examples here are synthetic.

| Token IDs identify choices **without ordering their meanings**

Synthetic vocabulary for the worked examples:

Token	<BOS>	please	send	the	report	<EOS>
ID	0	1	2	3	4	5

`please send the report` becomes the ID sequence (1, 2, 3, 4).

- IDs select categories; subtracting two IDs does not measure similarity.
- A token can be a word, a word piece, punctuation, or a special marker.

Next: turn each ID into learned features.

| An embedding table learns **features** for each token

Synthetic table $E \in \mathbb{R}^{6 \times 2}$:

Token	Feature 1	Feature 2
<BOS>	0	0
please	1	0
send	0	1
the	1	1
report	1	-1
<EOS>	-1	0

For token ID $w_t = 2$ (**send**):

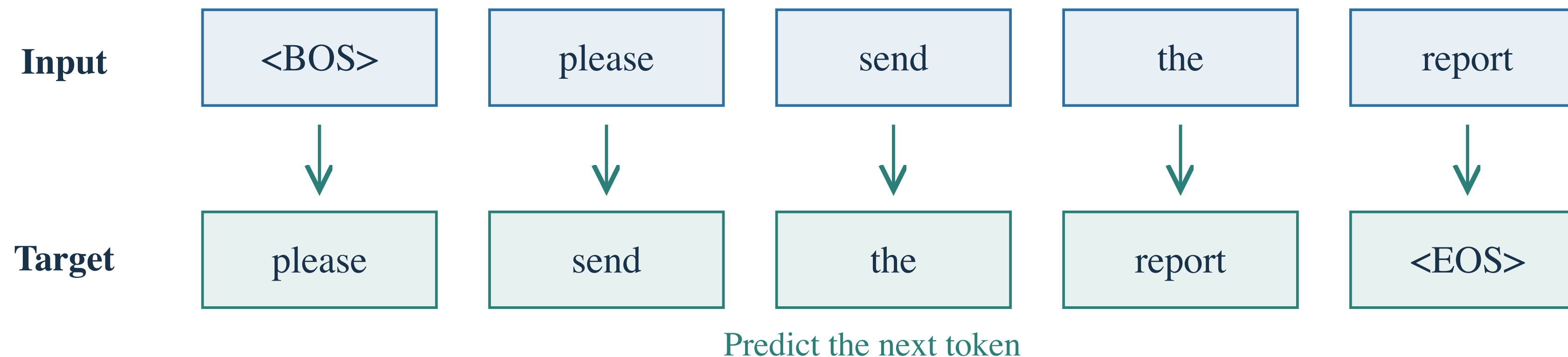
$$\mathbf{e}_t = E^\top \mathbf{o}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$\mathbf{o}_2 = [0, 0, 1, 0, 0, 0]^\top$ selects row 2.

- Lookup selects the same learned row wherever a token occurs.
- Training updates that row through the prediction loss.

Next: specify the target that teaches these features.

| Each input position predicts **the token immediately after it**



$$p_t = \text{softmax}(z_t), \quad \ell_t = -\log p_{t,w_{t+1}}$$

Allowed context for z_t : tokens through w_t only.

Putting it together: one next-token prediction at each position.

| The sequence model preserves positions while changing features

Stage	Tensor shape	Meaning
Input IDs	$[B, T]$	One integer ID at each position
Embedding lookup	$[B, T, d]$	d learned features per token
Context model	$[B, T, h]$	h context-dependent features per position
Vocabulary head	$[B, T, V]$	V logits per next-token prediction
Targets	$[B, T]$	One next-token ID per position

B : batch size. T : input length. V : vocabulary size.

Padding is absent data: exclude padded targets from the loss.

Before fitting: keep evaluation emails separate.

| Test completion on held-out email threads

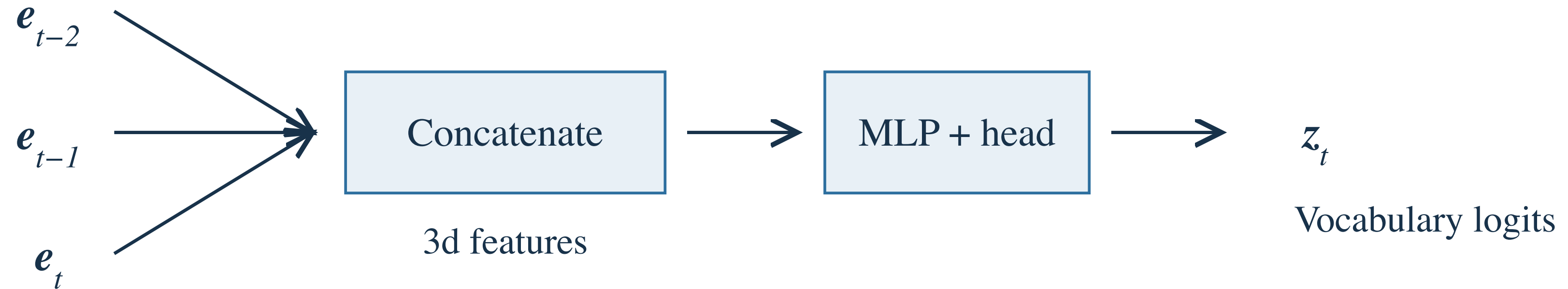
Group and split threads	Then create examples
Training threads	Training prefixes and next-token targets
Validation threads	Validation prefixes and next-token targets
Reserved test threads	Final held-out prefixes and targets

- Overlapping windows from one email share much of their text.
- Fit the vocabulary on **training emails only**; handle unseen tokens explicitly.

Now build a baseline: how much of the prefix should it read?

| A fixed window turns continuation into **familiar supervised prediction**

Last three token embeddings



Earlier tokens are outside this predictor's input.

$$z_t = f_{\theta} \left(\begin{bmatrix} e_{t-2} \\ e_{t-1} \\ e_t \end{bmatrix} \right) \in \mathbb{R}^v$$

Its limitation: anything before the window is unavailable to the model.

| Which change lets the model distinguish these contexts?

“Send the **report** today. Please attach the ...”

“Send the **slides** today. Please attach the ...”

Which change permits different predictions from a three-token model?

A. Add more hidden units

B. Include earlier tokens

C. Train on more emails

B. Identical input windows produce the same predicted distribution for fixed parameters.

| A recurrent state carries **earlier information into the next step**

Fixed-window predictor	Recurrent predictor
Reads the most recent k tokens	Reads the current token and previous state
Earlier tokens fall outside the input	Earlier tokens can affect the state
Input width depends on k	State width stays fixed

Next question: how can one learned update carry useful earlier information forward?