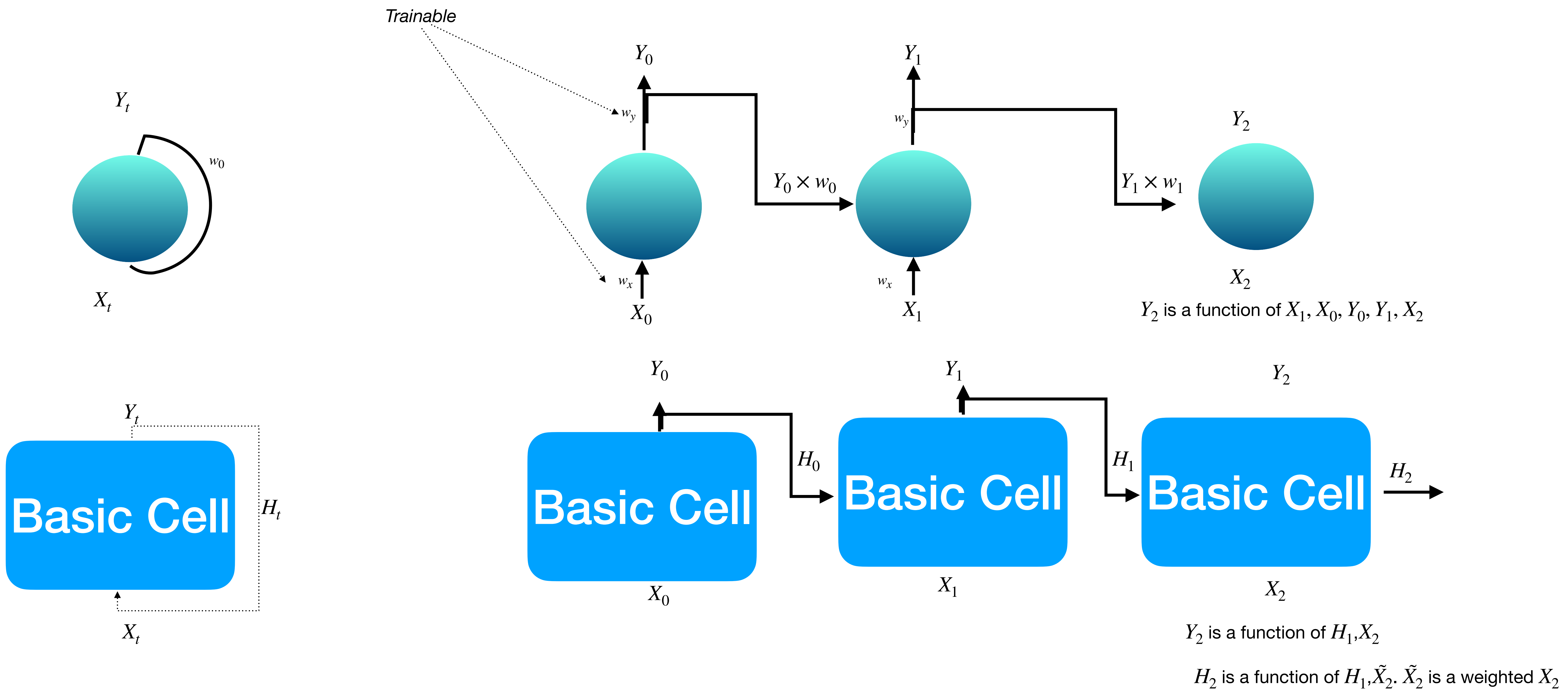
Stateless

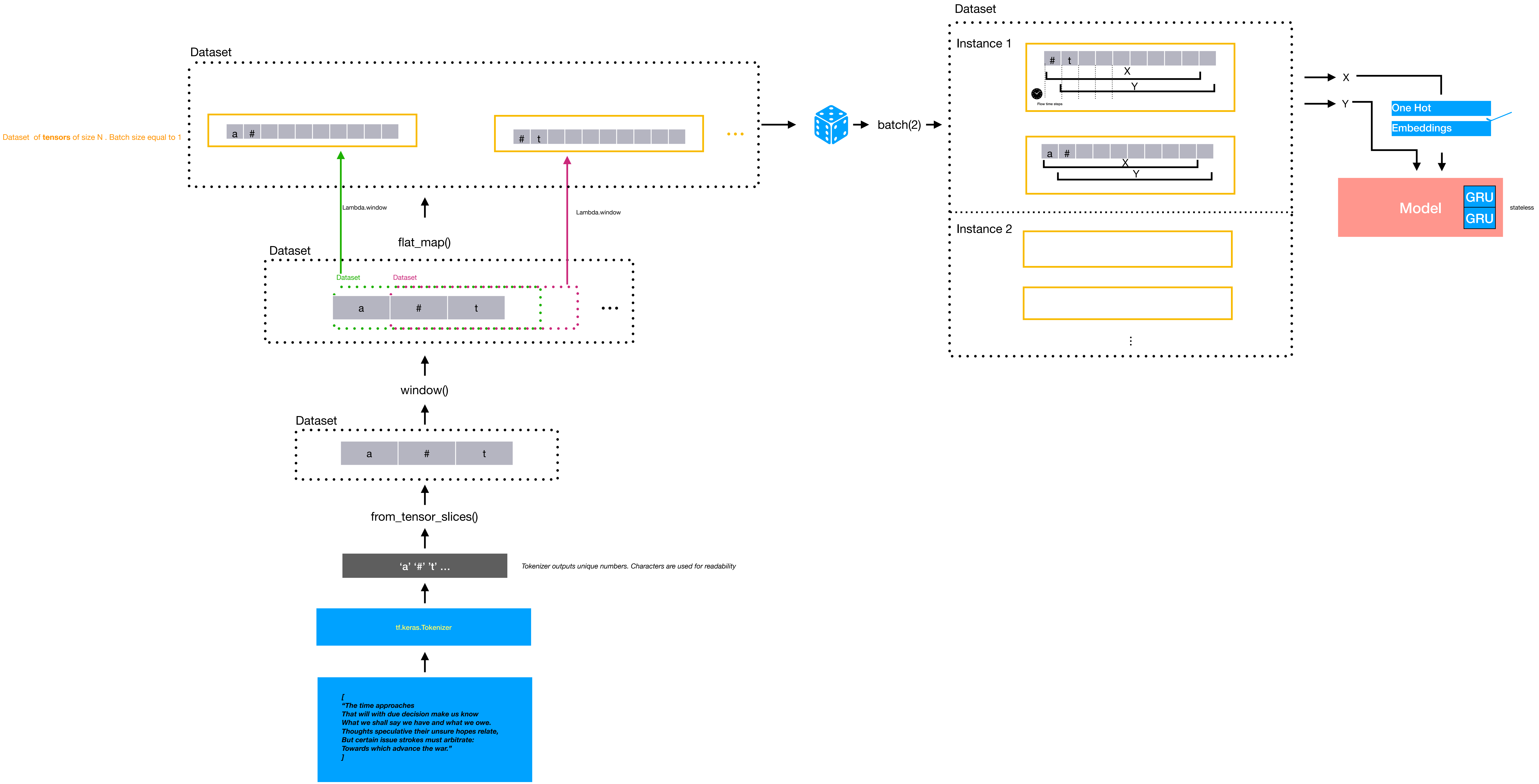
Learns on random portions of text, without any information about the text

Stateful

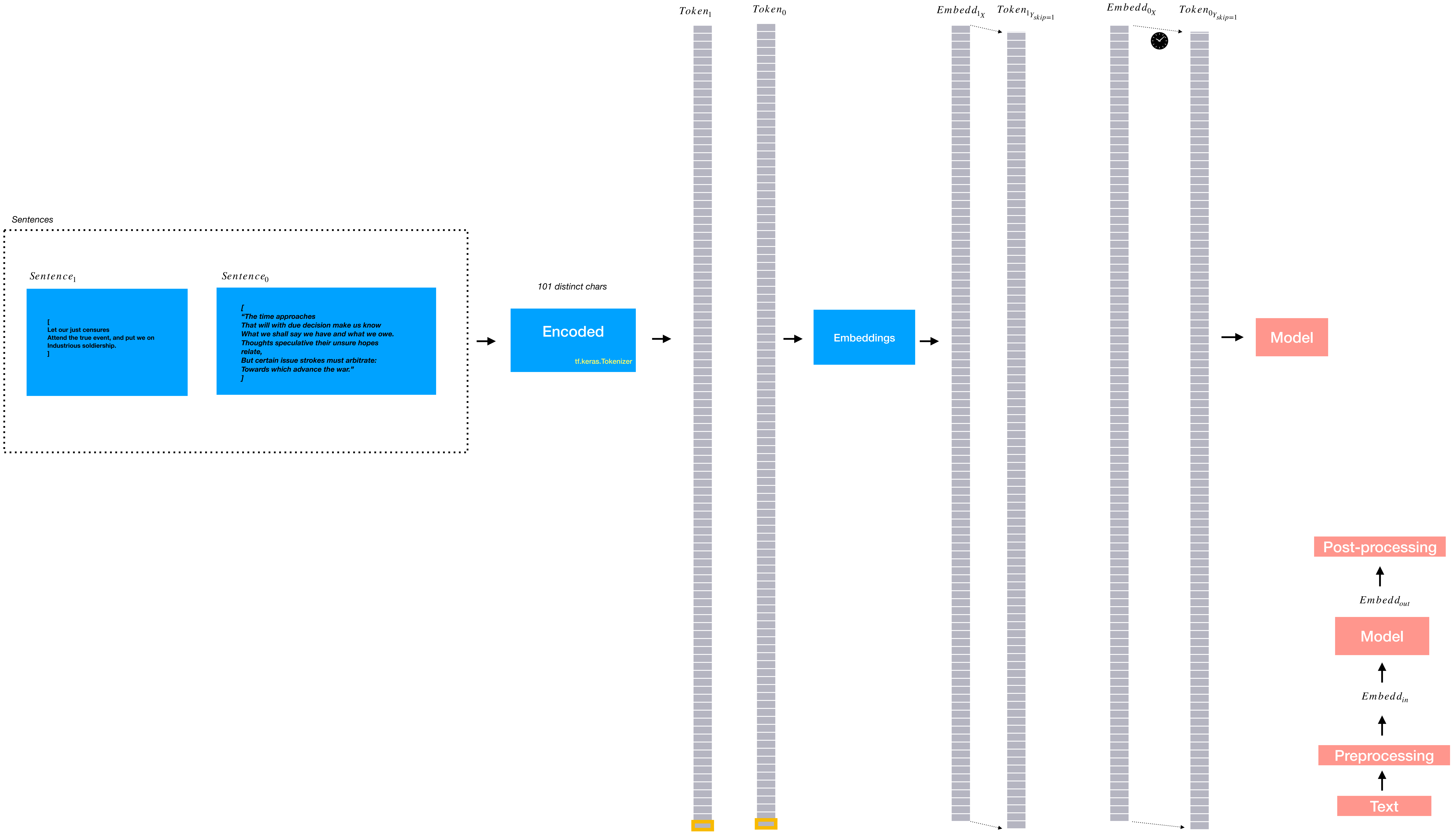
Learned hidden state preserved , allowing the model to learn longer patterns



Trainable
Y_t
w_0
X_t
Y_0
w_y
w_x
X_0
$Y_0 \times w_0$
Y_1
w_y
w_x
X_1
$Y_1 \times w_1$
Y_2
X_2
Y_2 is a function of X_1, X_0, Y_0, Y_1, X_2
Y_t
Basic Cell
H_t
X_t
Y_0
Basic Cell
X_0
H_0
Y_1
Basic Cell
X_1
H_1
Y_2
Basic Cell
X_2
H_2
Y_2 is a function of H_1,X_2
H_2 is a function of H_1,$\tilde{X}_2$. $\tilde{X}_2$ is a weighted X_2



Dataset of tensors of size N . Batch size equal to 1
Dataset
a #
t
...
Lambda.window
Lambda.window
flat_map()
Dataset
Dataset
Dataset
a # t
...
window()
Dataset
a # t
from_tensor_slices()
'a' '#' 't' ...
Tokenizer outputs unique numbers. Characters are used for readability
tf.keras.Tokenizer
[
"The time approaches
That will with due decision make us know
What we shall say we have and what we owe.
Thoughts speculative their unsure hopes relate,
But certain issue strokes must arbitrate:
Towards which advance the war."
]
batch(2)
Dataset
Instance 1
t
X
Y
a #
X
Y
Instance 2
X
Y
One Hot
Embeddings
Model
GRU
GRU
stateless

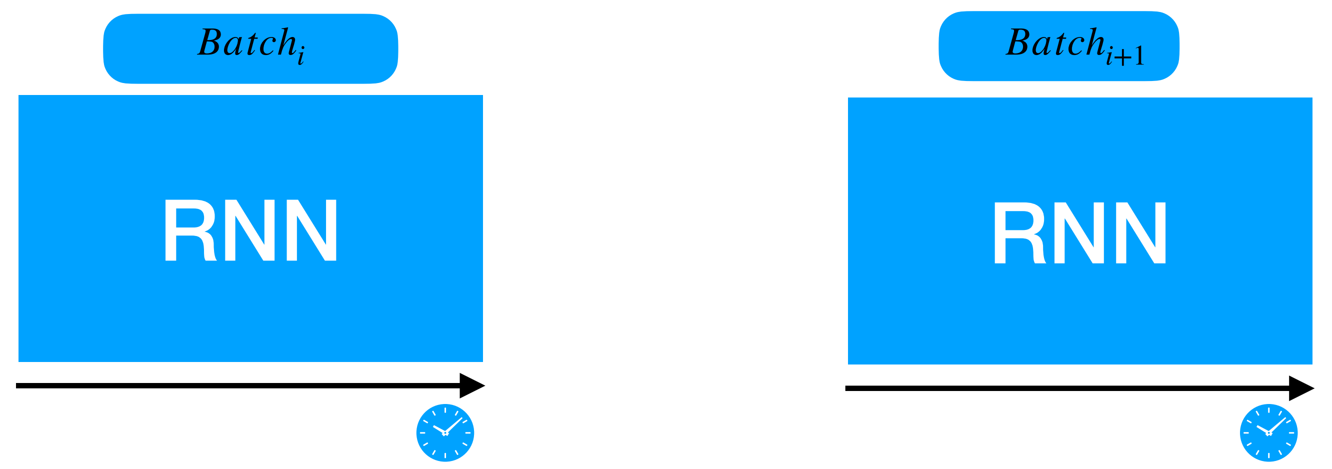


Model train gray data to predict single sample.
Models goal is to train sentence and predict next character

Batch = Training Iteration

Stateless

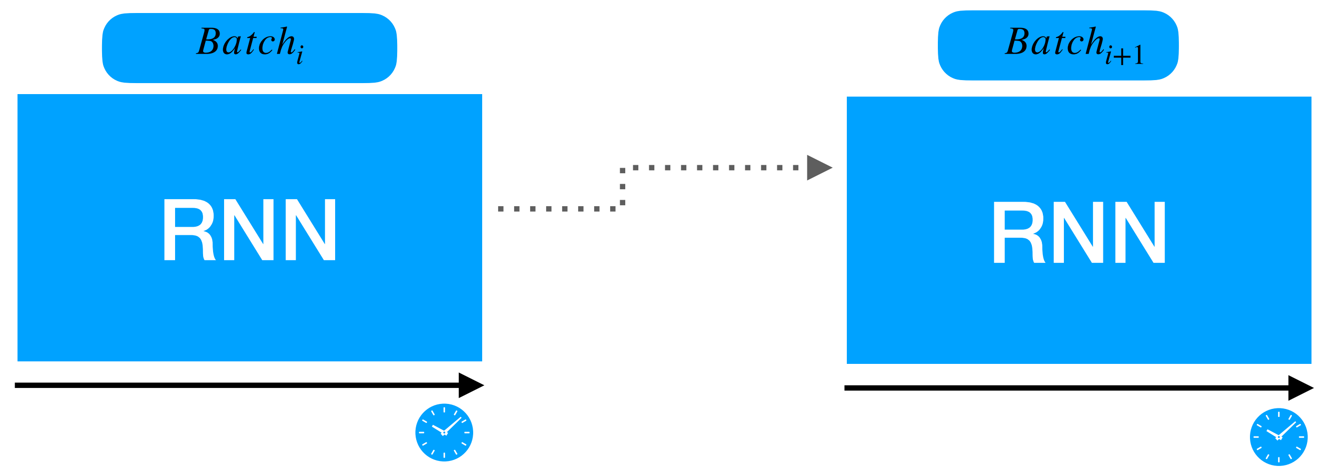
Learns on random portions of text, without any information about the text



Hidden state lost after each training iteration.

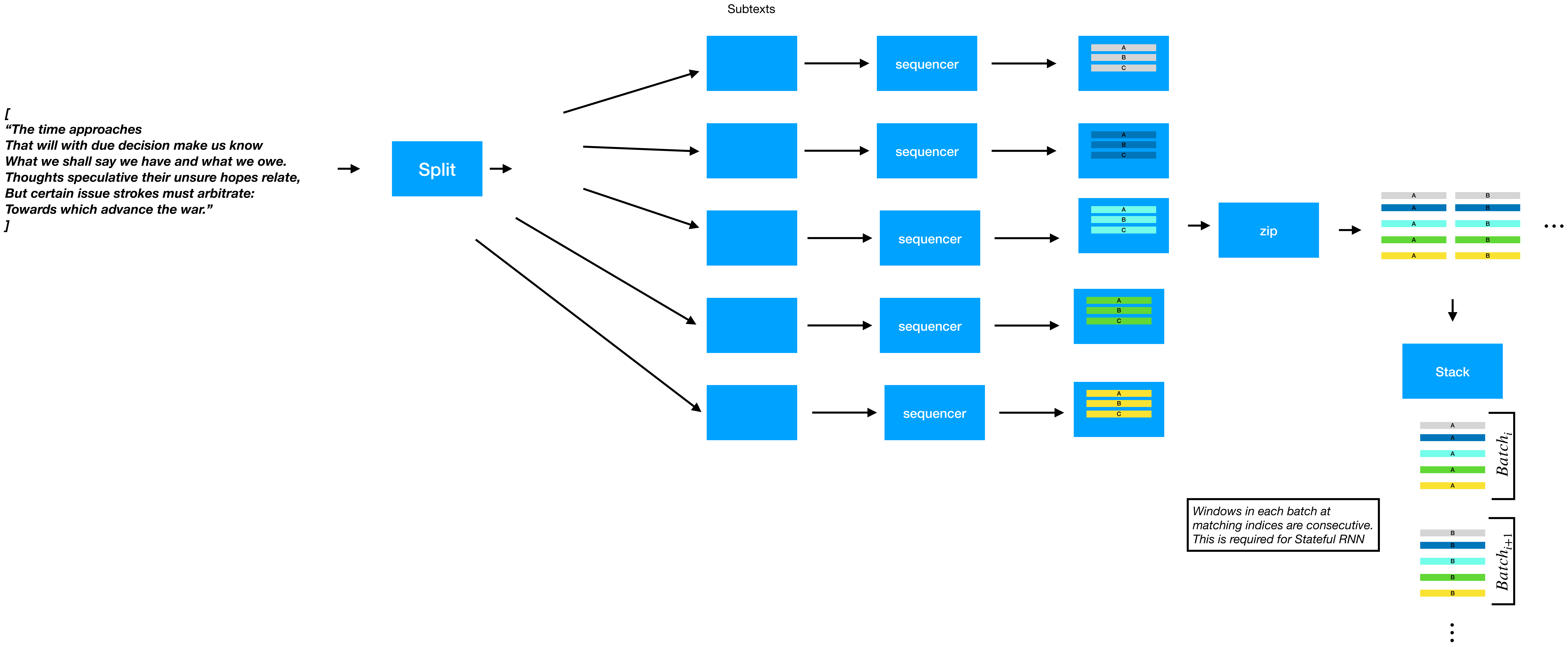
Stateful

Learned hidden state preserved , allowing the model to learn longer patterns



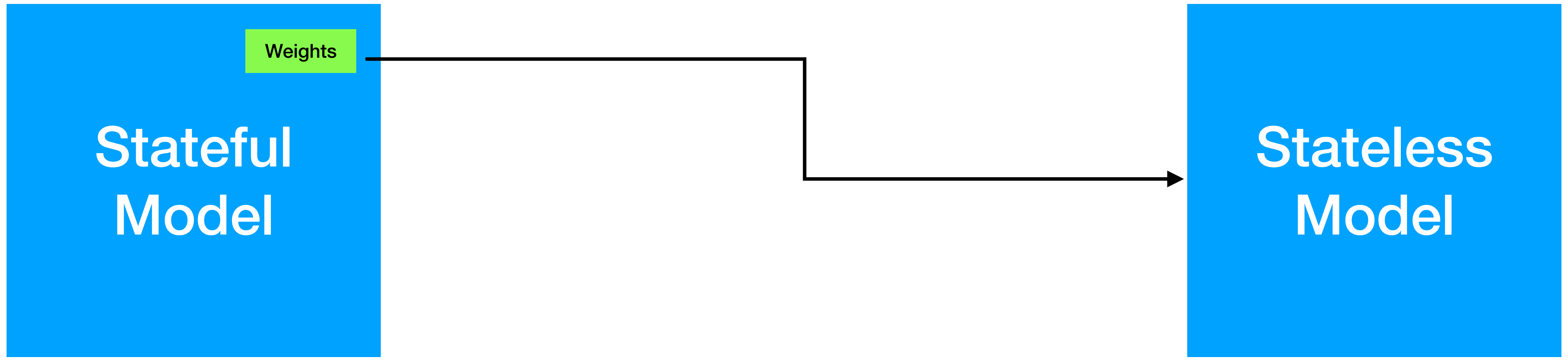
Preserve hidden state and using to train next batch

Batching: Stateful RNN



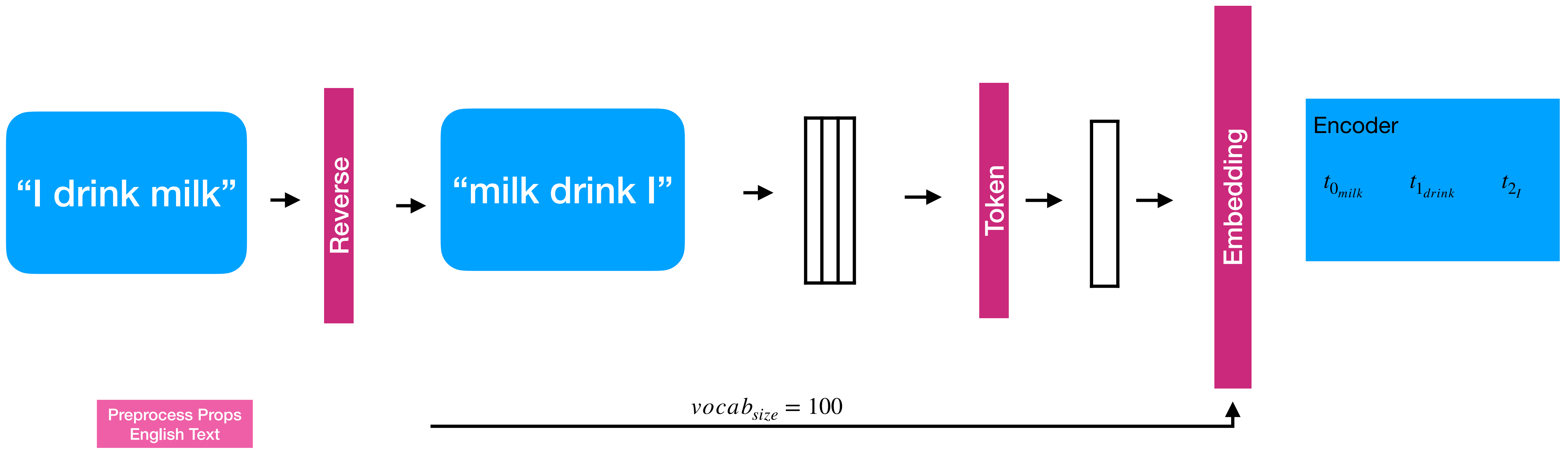
Batching: Stateful RNN

Batch size restricted to the training batch_size

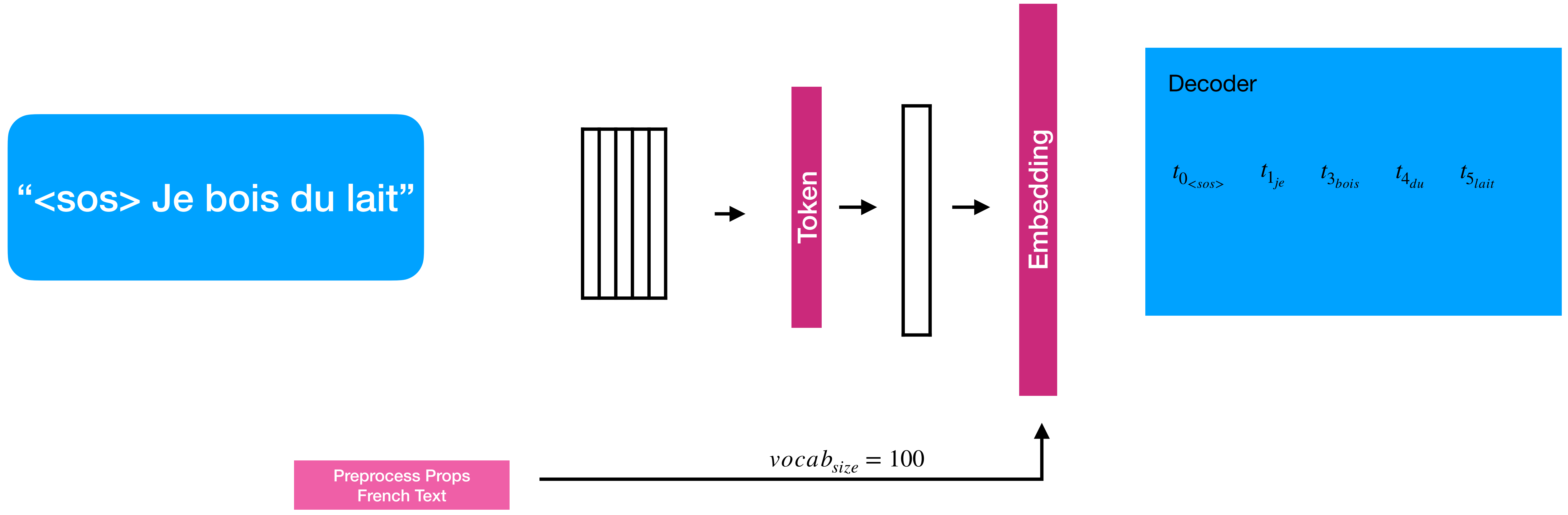


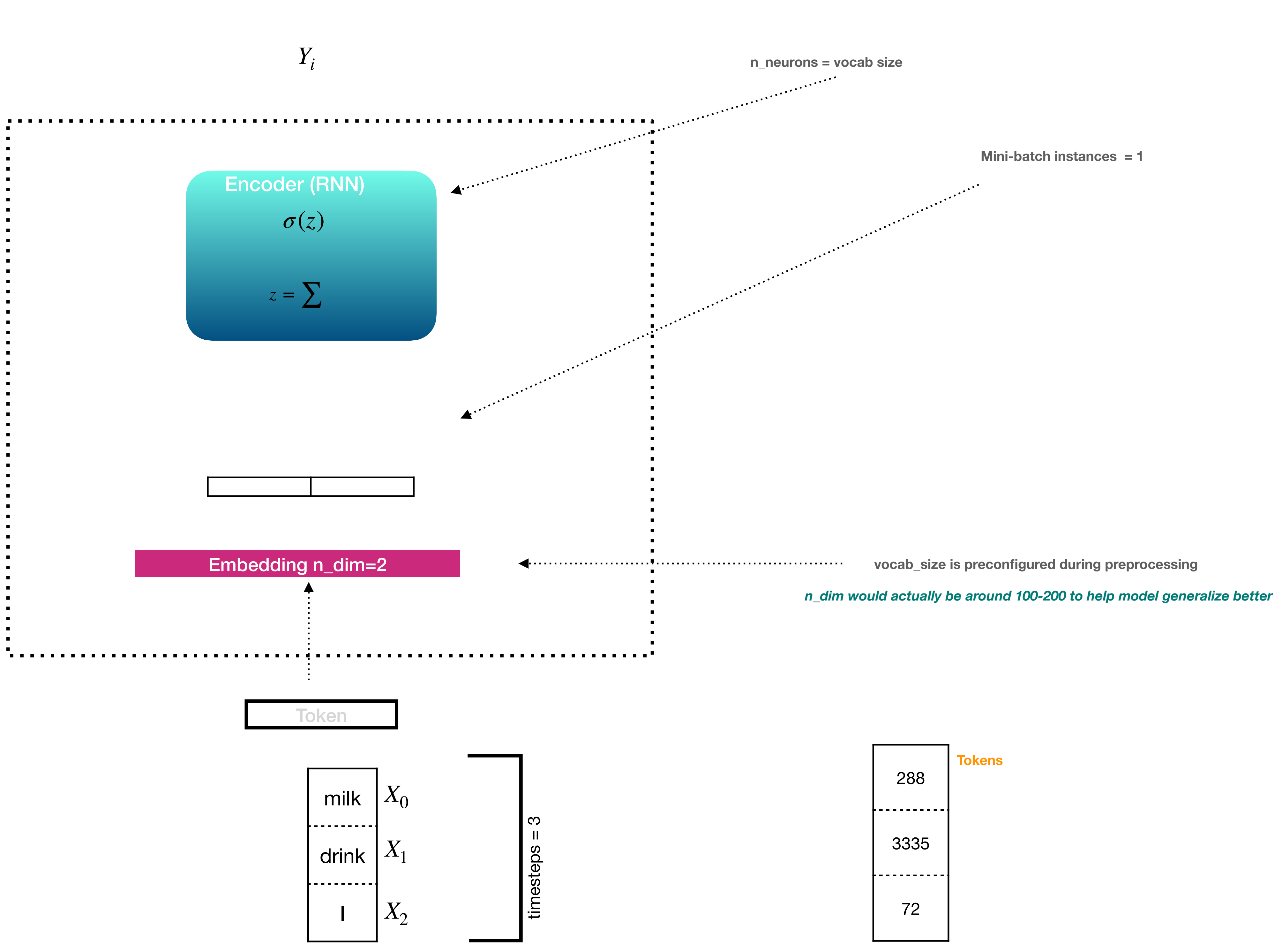
Copy model structure and weights to stateless model and enjoy inference

Neural Machine Translation (NMT)

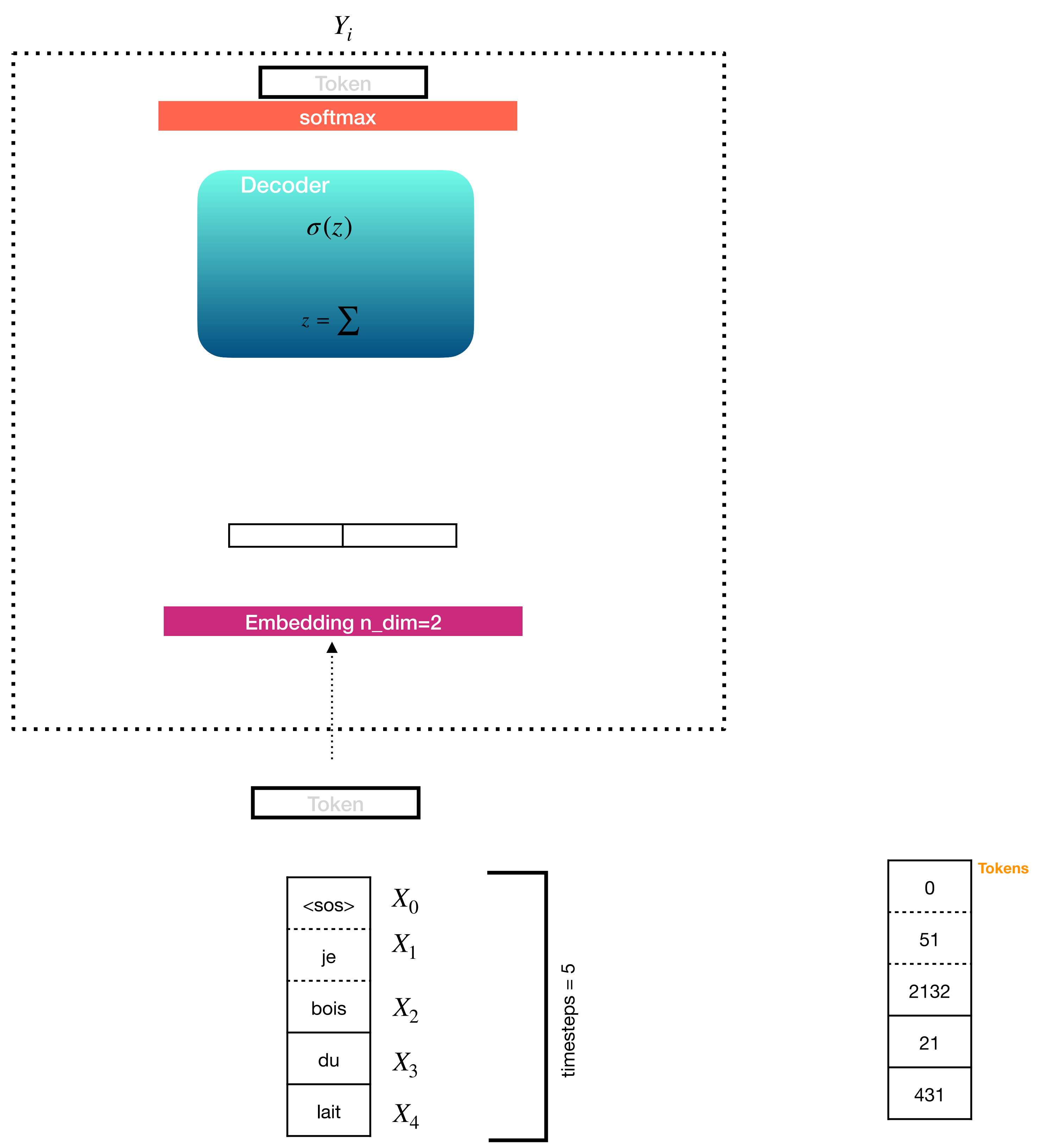


encoder learns to correlate sentence with encodings embedding

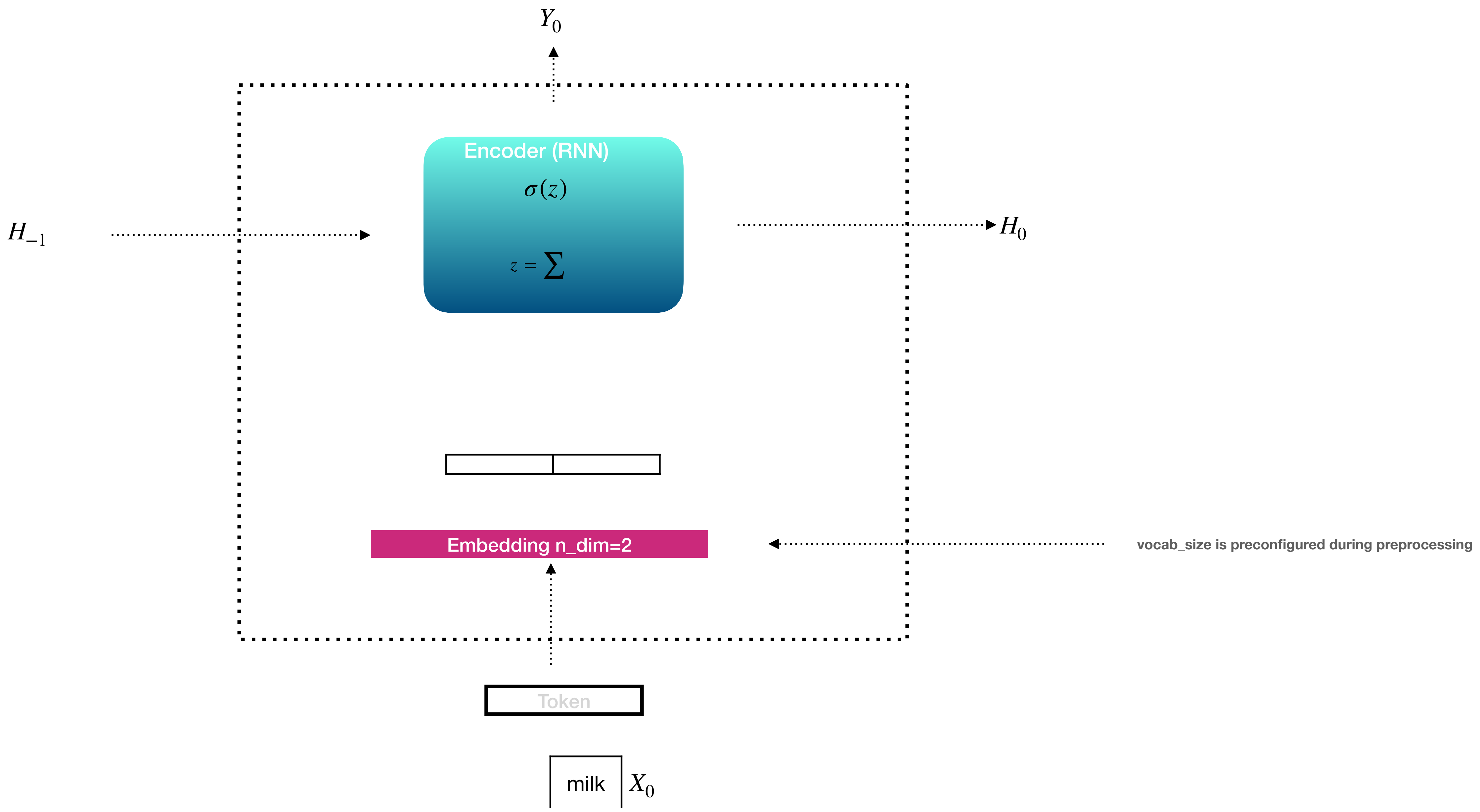




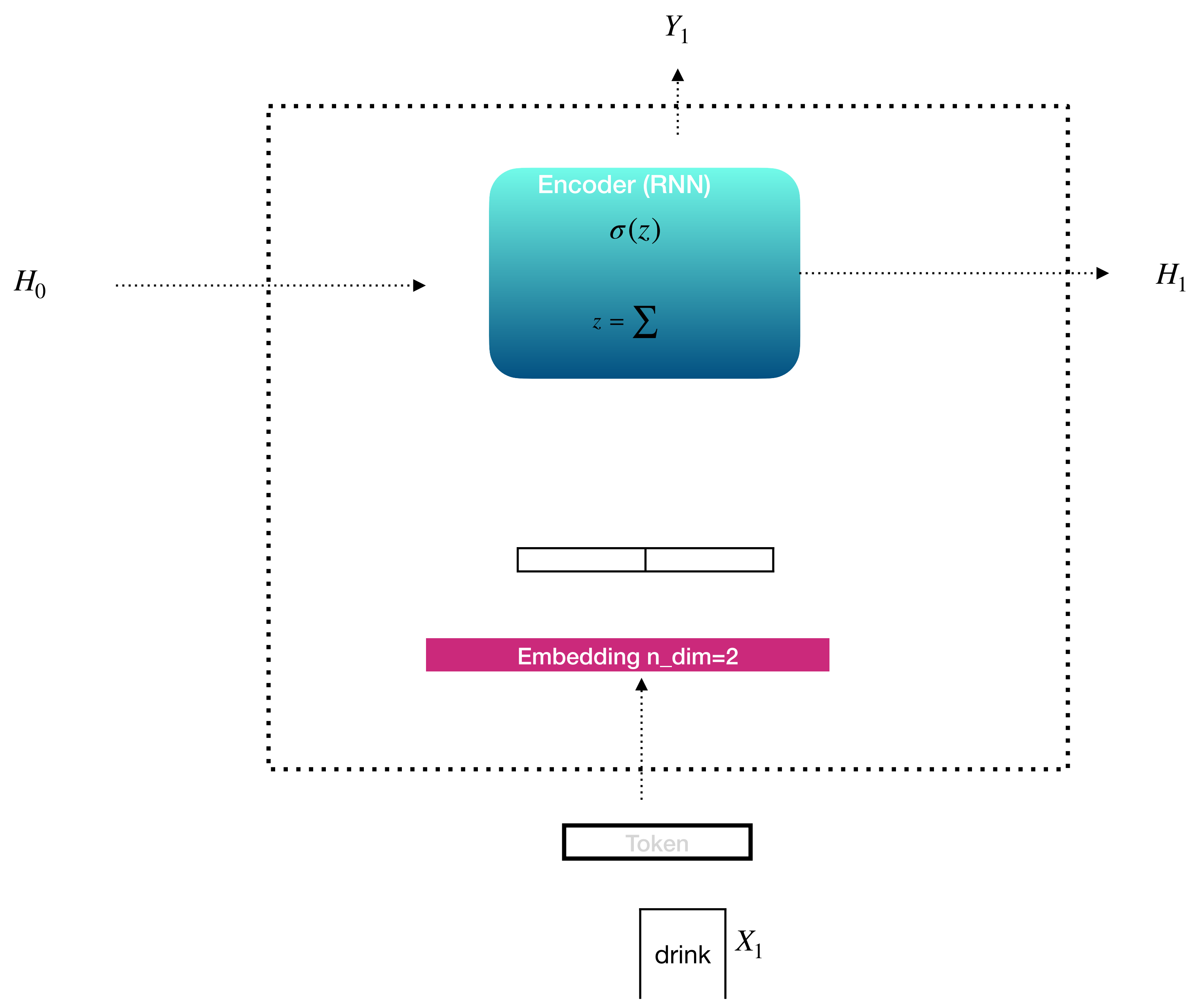
Y_i
n_neurons = vocab size
Mini-batch instances = 1
Encoder (RNN)
$\sigma(z)$
$z = \sum$
Embedding n_dim=2
vocab_size is preconfigured during preprocessing
n_dim would actually be around 100-200 to help model generalize better
Token
milk X_0
drink X_1
I X_2
timesteps = 3
288
3335
72
Tokens



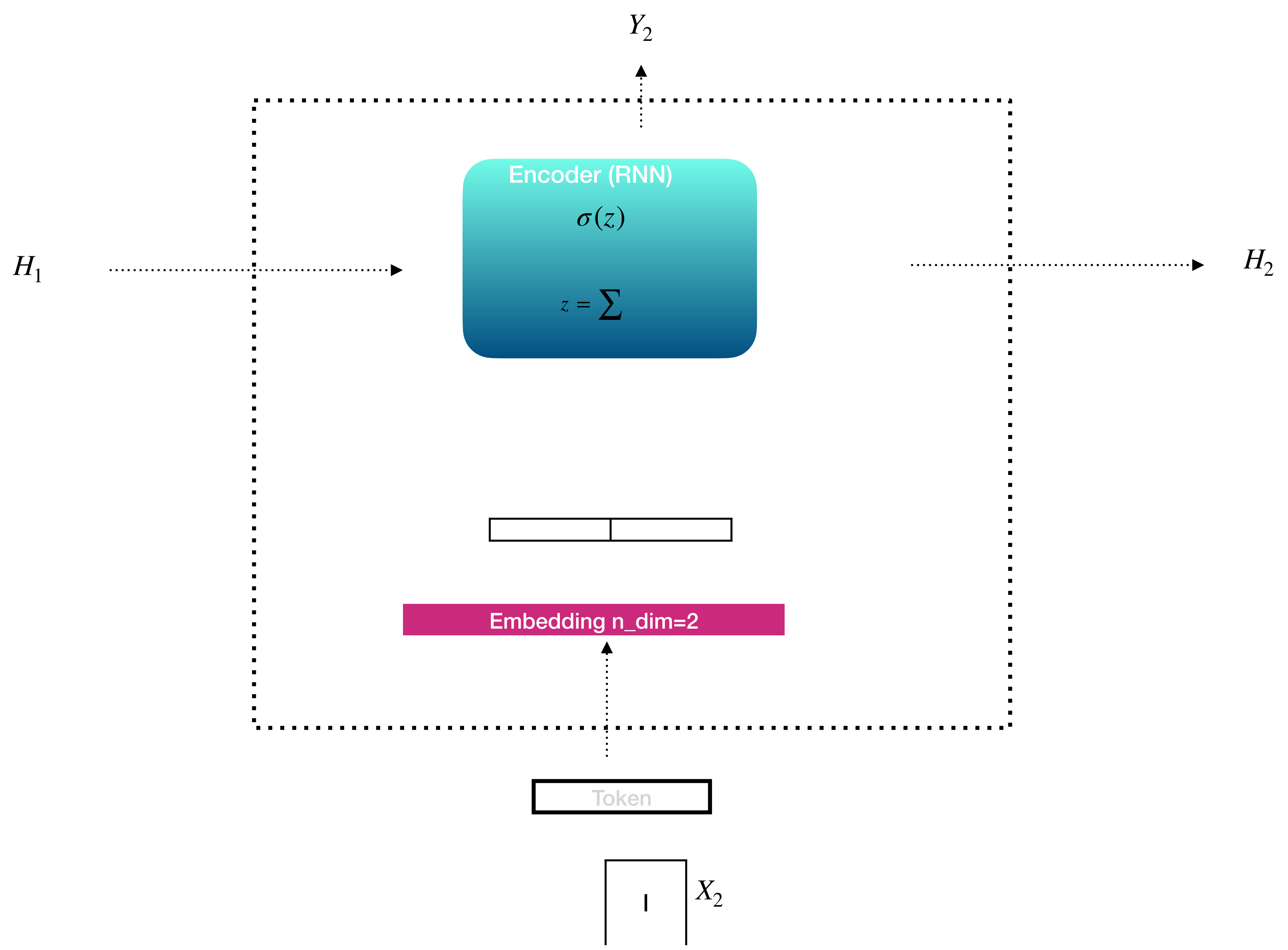
Y_i
Token
softmax
Decoder
$\sigma(z)$
$z = \sum$
Embedding n_dim=2
Token
<sos> X_0
je X_1
bois X_2
du X_3
lait X_4
timesteps = 5
0
51
2132
21
431
Tokens



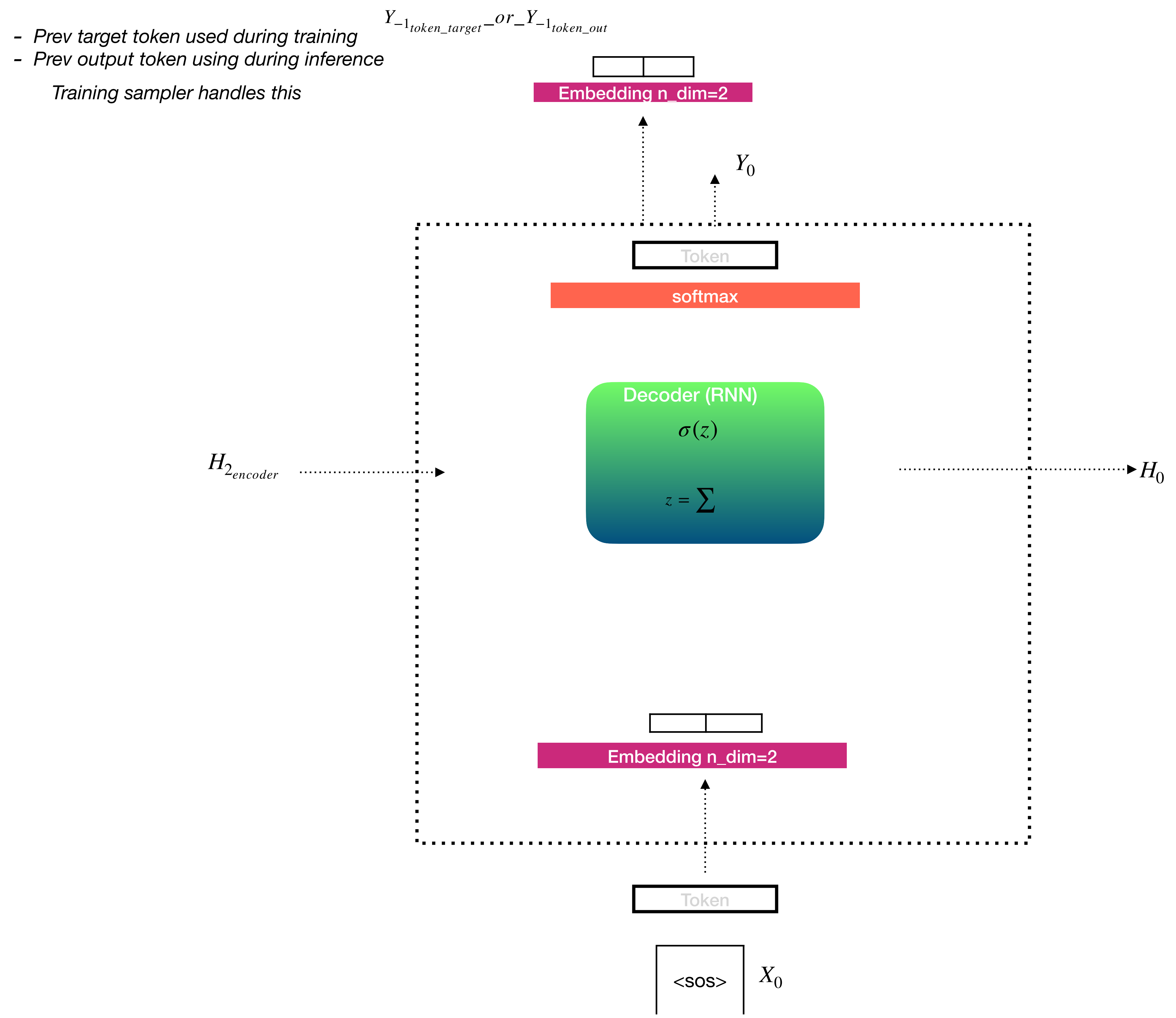
Y_0
Encoder (RNN)
$\sigma(z)$
$z = \sum$
H_{-1}
H_0
Embedding n_dim=2
vocab_size is preconfigured during preprocessing
Token
milk
X_0

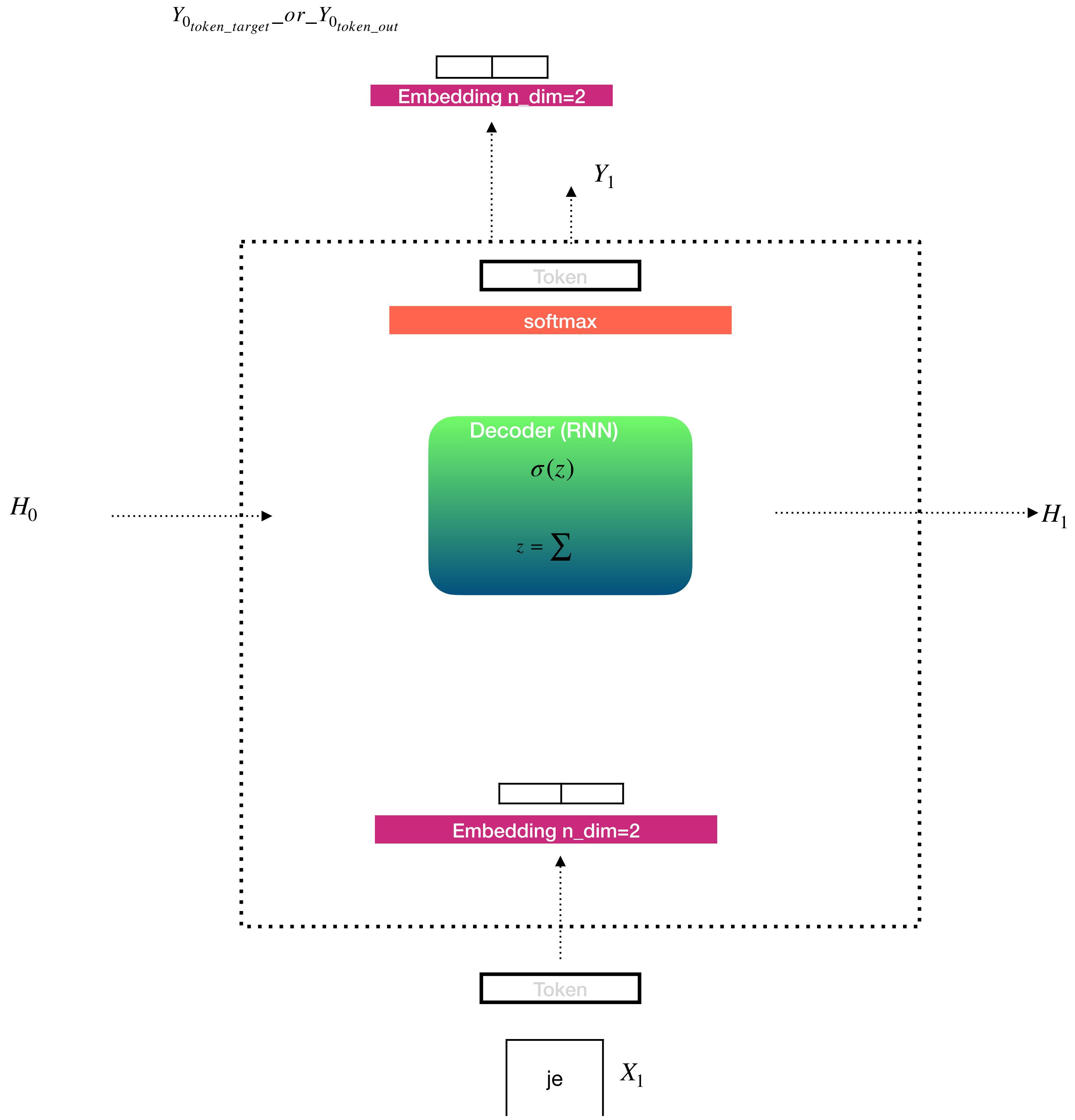


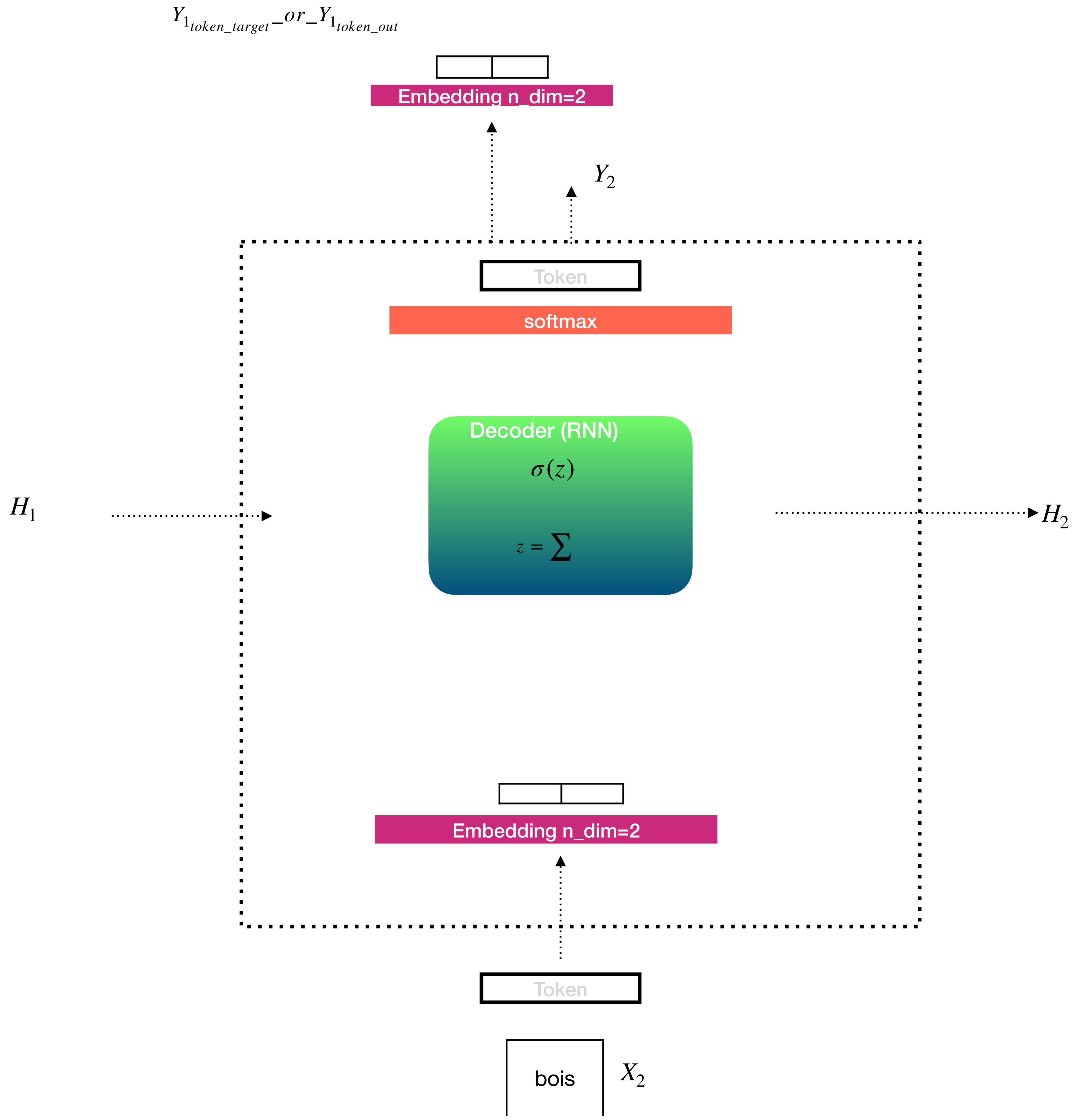
Y_1
Encoder (RNN)
$\sigma(z)$
$z = \sum$
H_0
H_1
Embedding n_dim=2
Token
drink
X_1



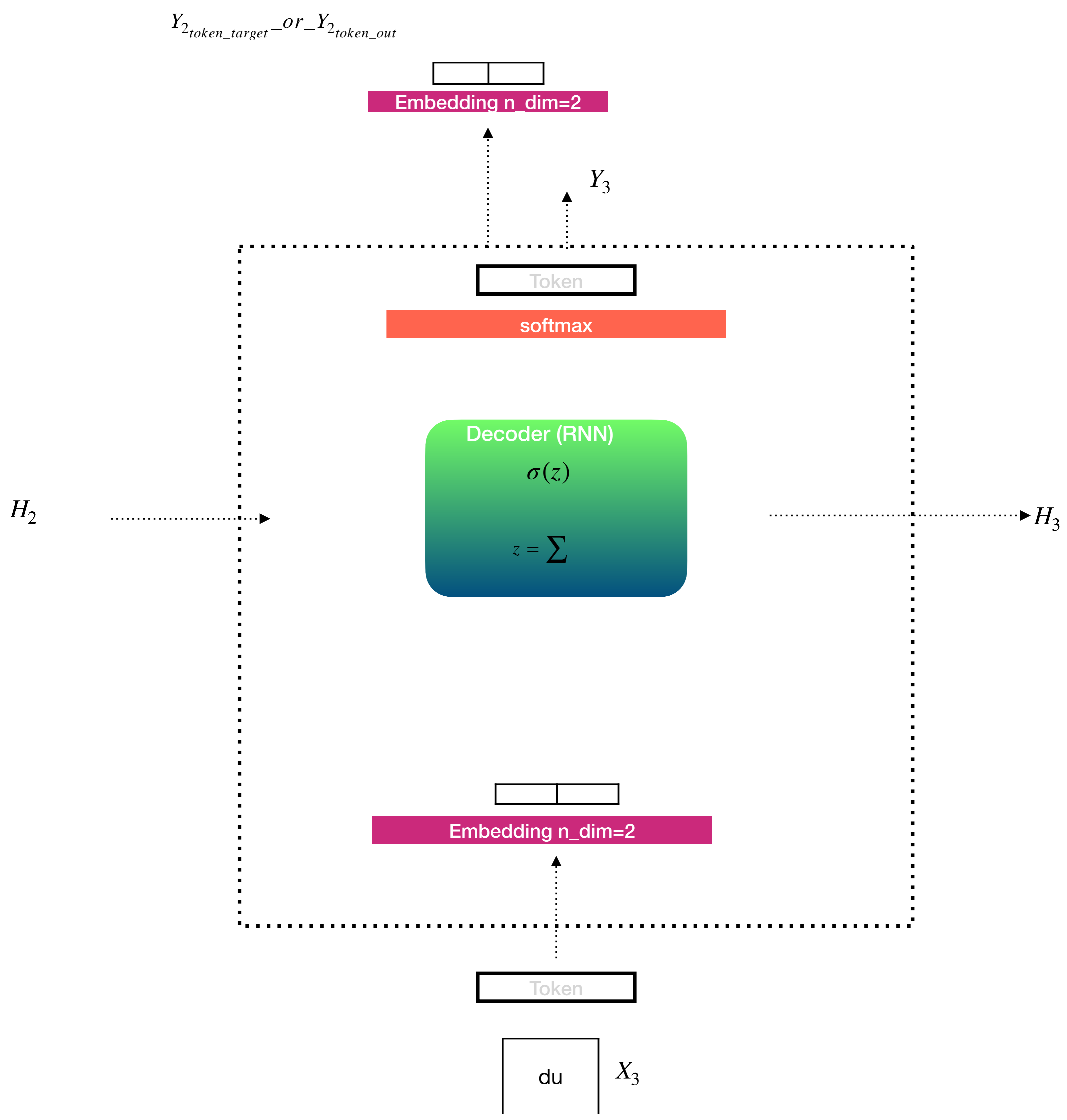
Y_2
Encoder (RNN)
$\sigma(z)$
$z = \sum$
H_1
H_2
Embedding n_dim=2
Token
I
X_2

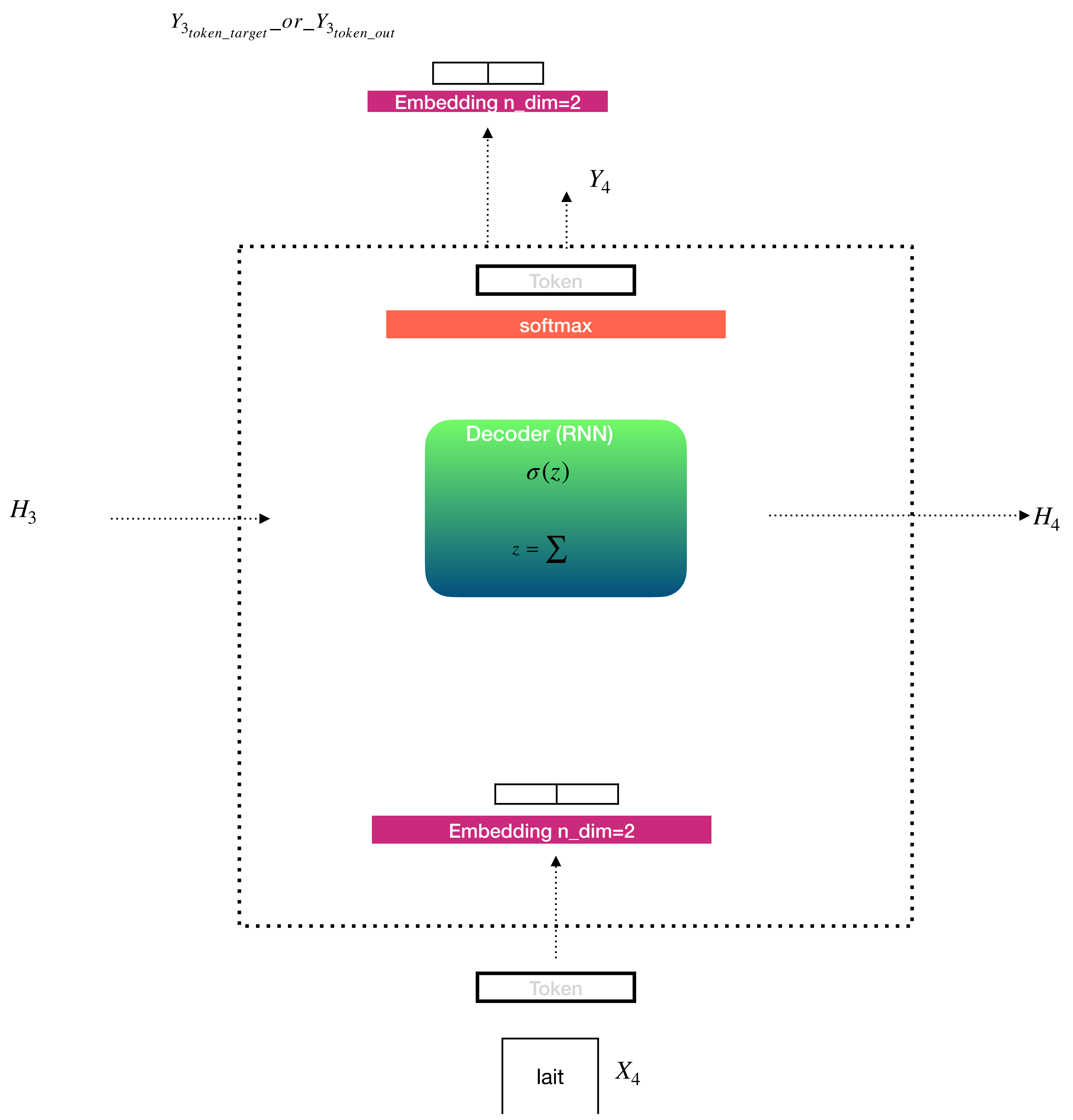

- Prev target token used during training
- Prev output token using during inference
Training sampler handles this
$Y_{-1_{token\_target}}\_or\_Y_{-1_{token\_out}}$
Embedding n_dim=2
Y_0
Token
softmax
Decoder (RNN)
$\sigma(z)$
$z = \sum$
$H_{2_{encoder}}$
H_0
Embedding n_dim=2
Token
<sos>
X_0


$Y_{0_{token\_target}}\_or\_Y_{0_{token\_out}}$
Embedding n_dim=2
Y_1
Token
softmax
Decoder (RNN)
$\sigma(z)$
$z = \sum$
H_0
H_1
Embedding n_dim=2
Token
je
X_1

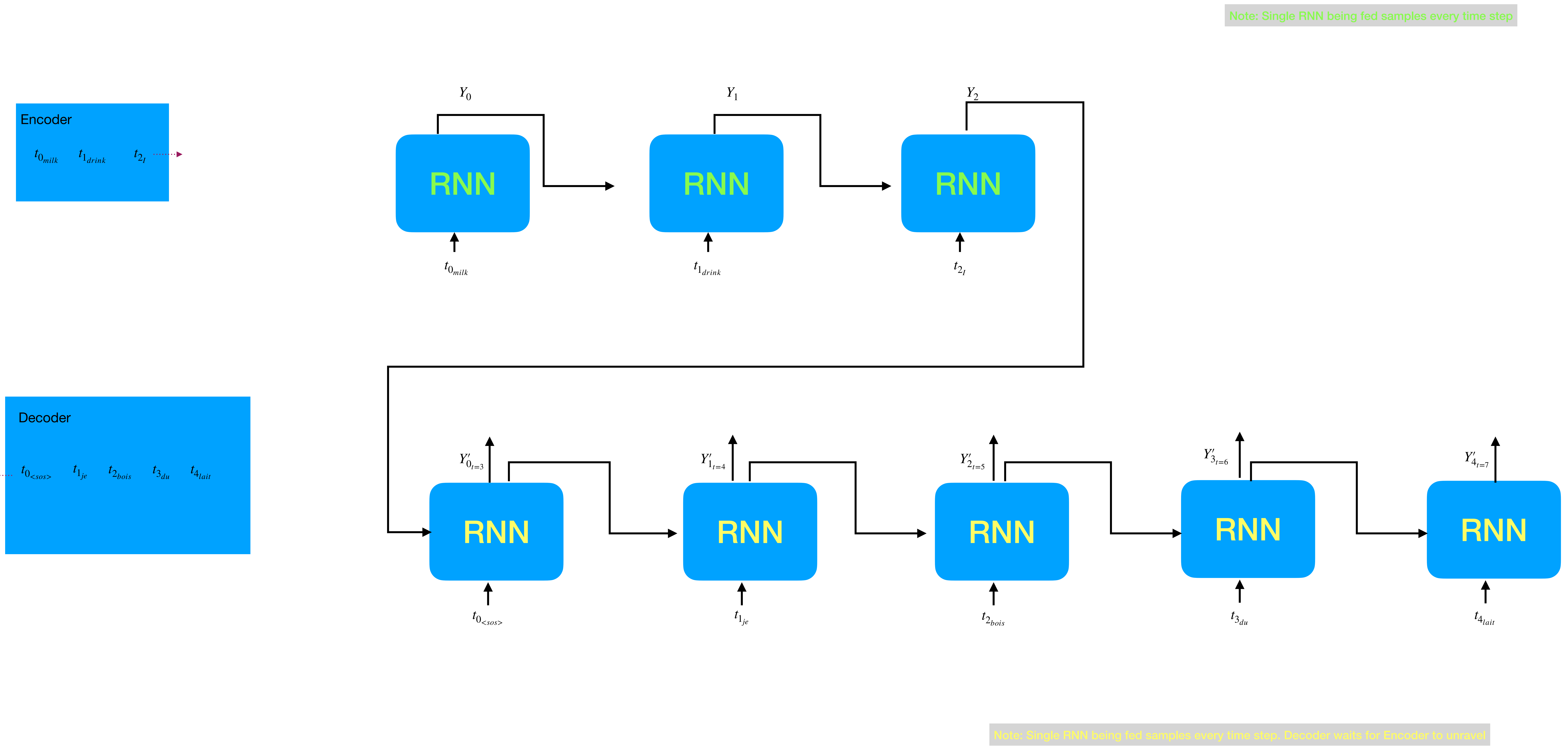


$Y_{1_{token\_target}}\_or\_Y_{1_{token\_out}}$
Embedding n_dim=2
Y_2
Token
softmax
Decoder (RNN)
$\sigma(z)$
$z = \sum$
H_1
H_2
Embedding n_dim=2
Token
bois
X_2

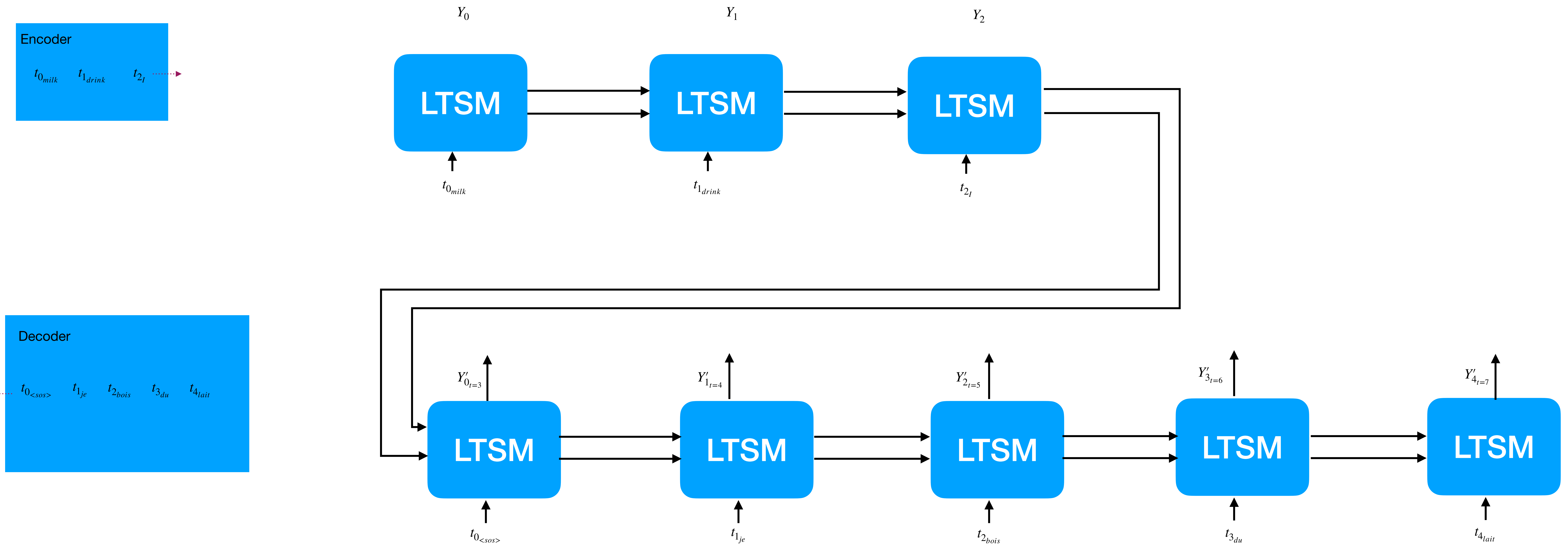

$Y_{2_{token\_target}}\_or\_Y_{2_{token\_out}}$
Embedding n_dim=2
Y_3
Token
softmax
Decoder (RNN)
$\sigma(z)$
$z = \sum$
H_2
H_3
Embedding n_dim=2
Token
du
X_3



$Y_{3_{token\_target}}\_or\_Y_{3_{token\_out}}$
Embedding n_dim=2
Y_4
Token
softmax
Decoder (RNN)
$\sigma(z)$
$z = \sum$
H_3
H_4
Embedding n_dim=2
Token
lait
X_4

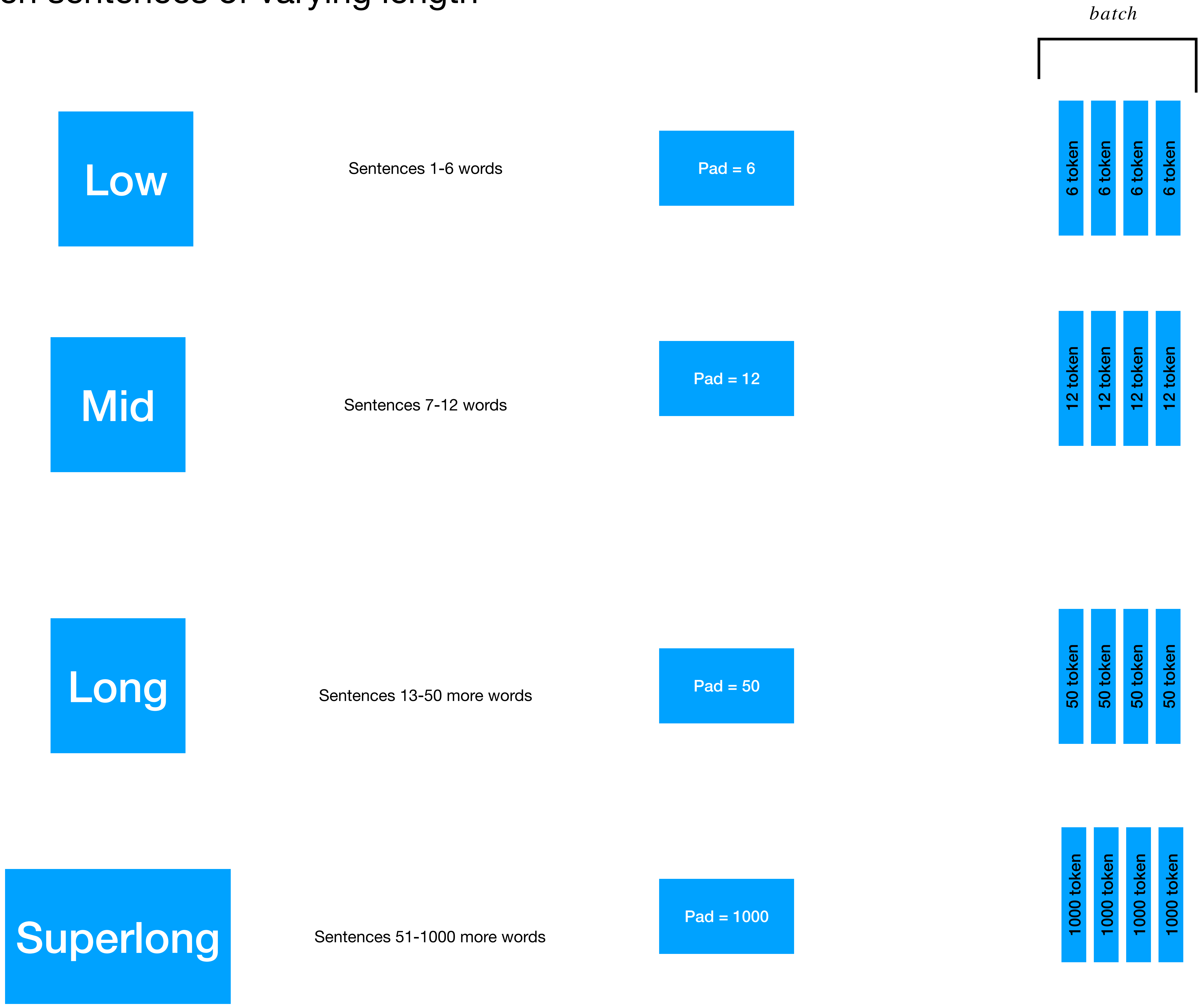


Model learns to translate English sentence into French

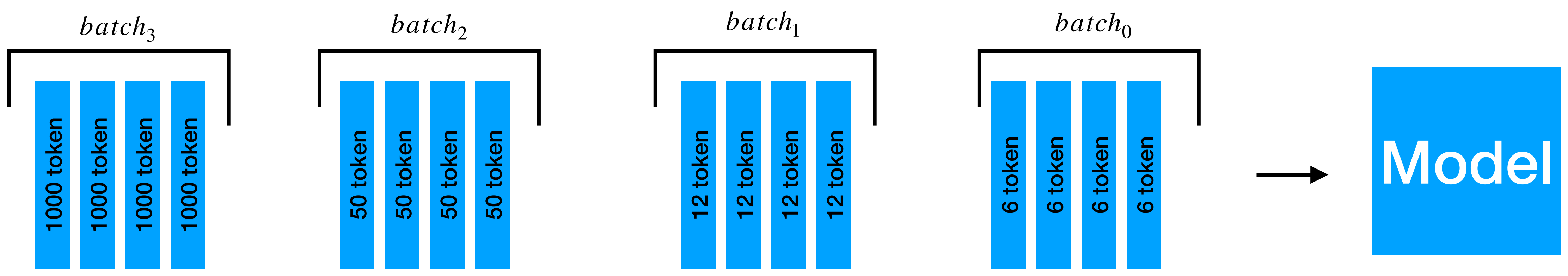


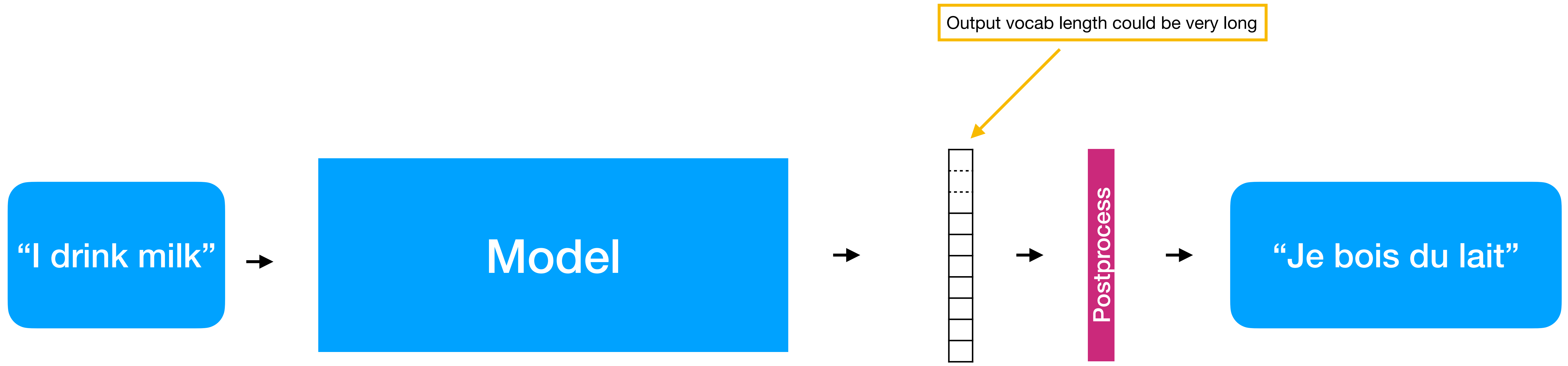
Model learns to translate English sentence into French

Batch sentences of varying length

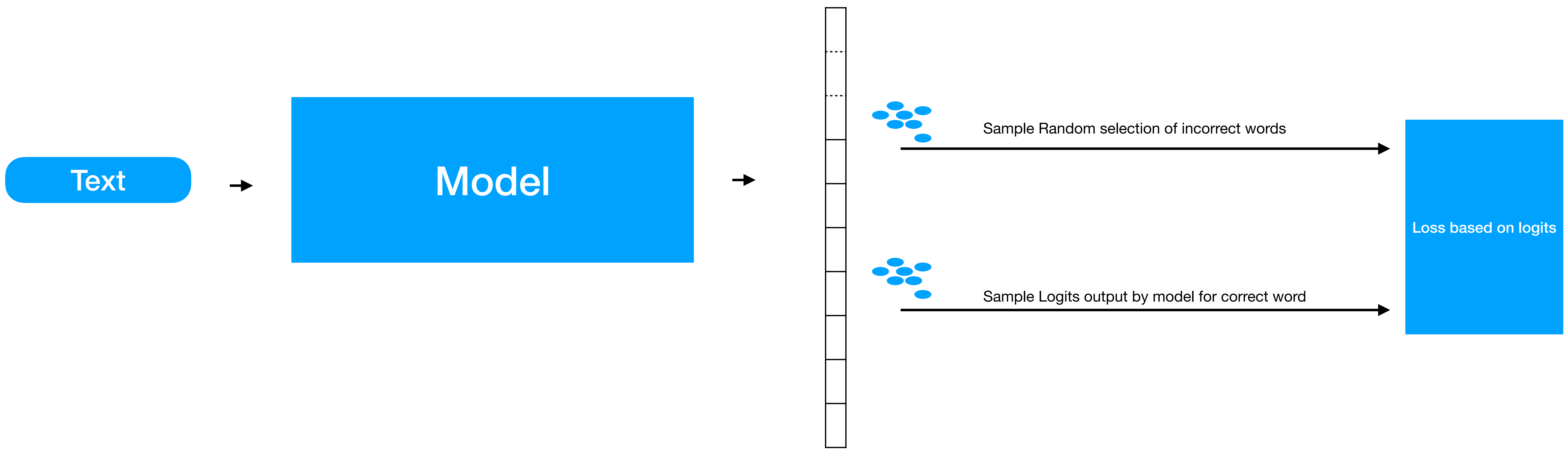


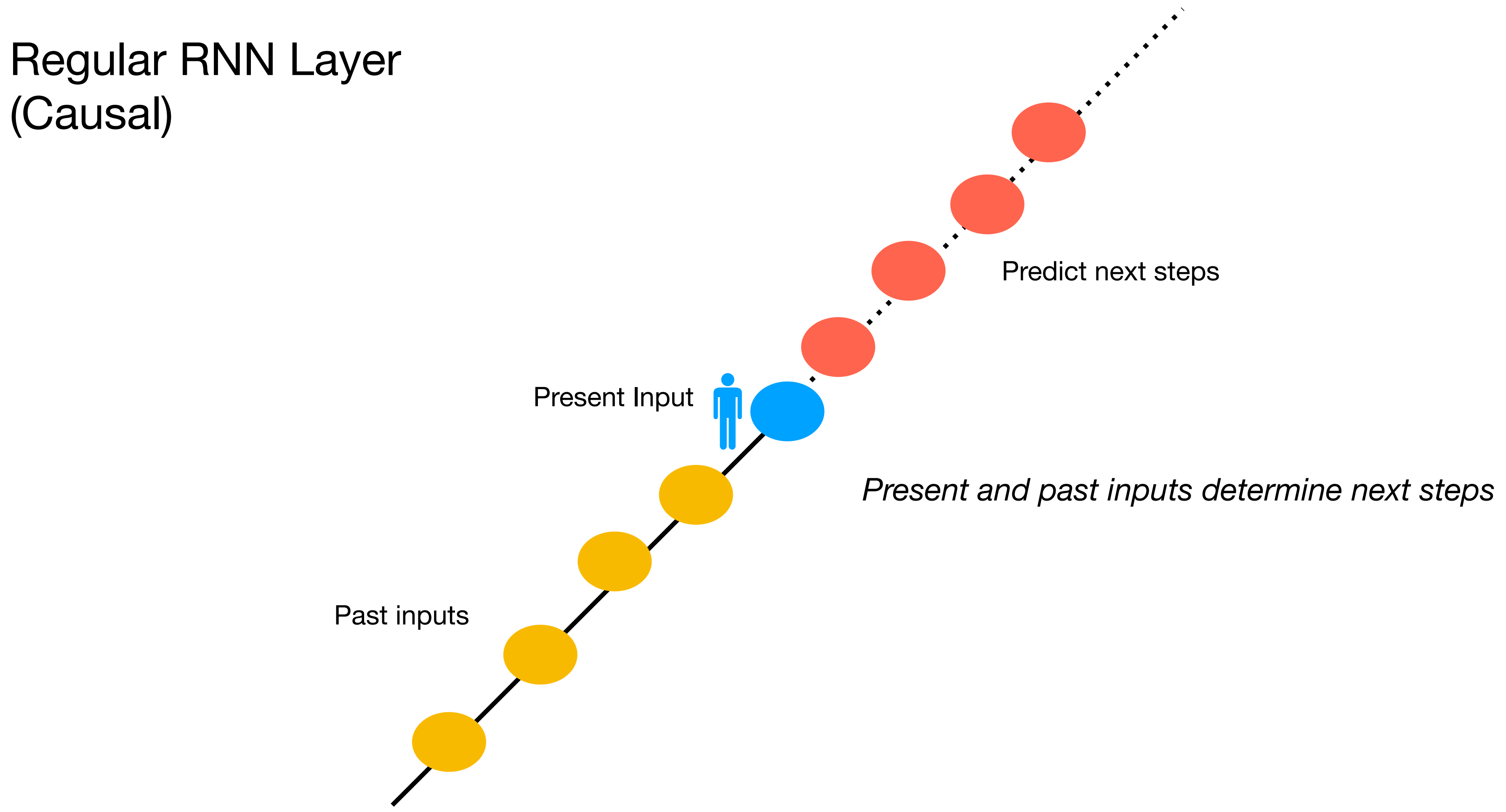
Batch of equal length Tensors are accepted as inputs to model

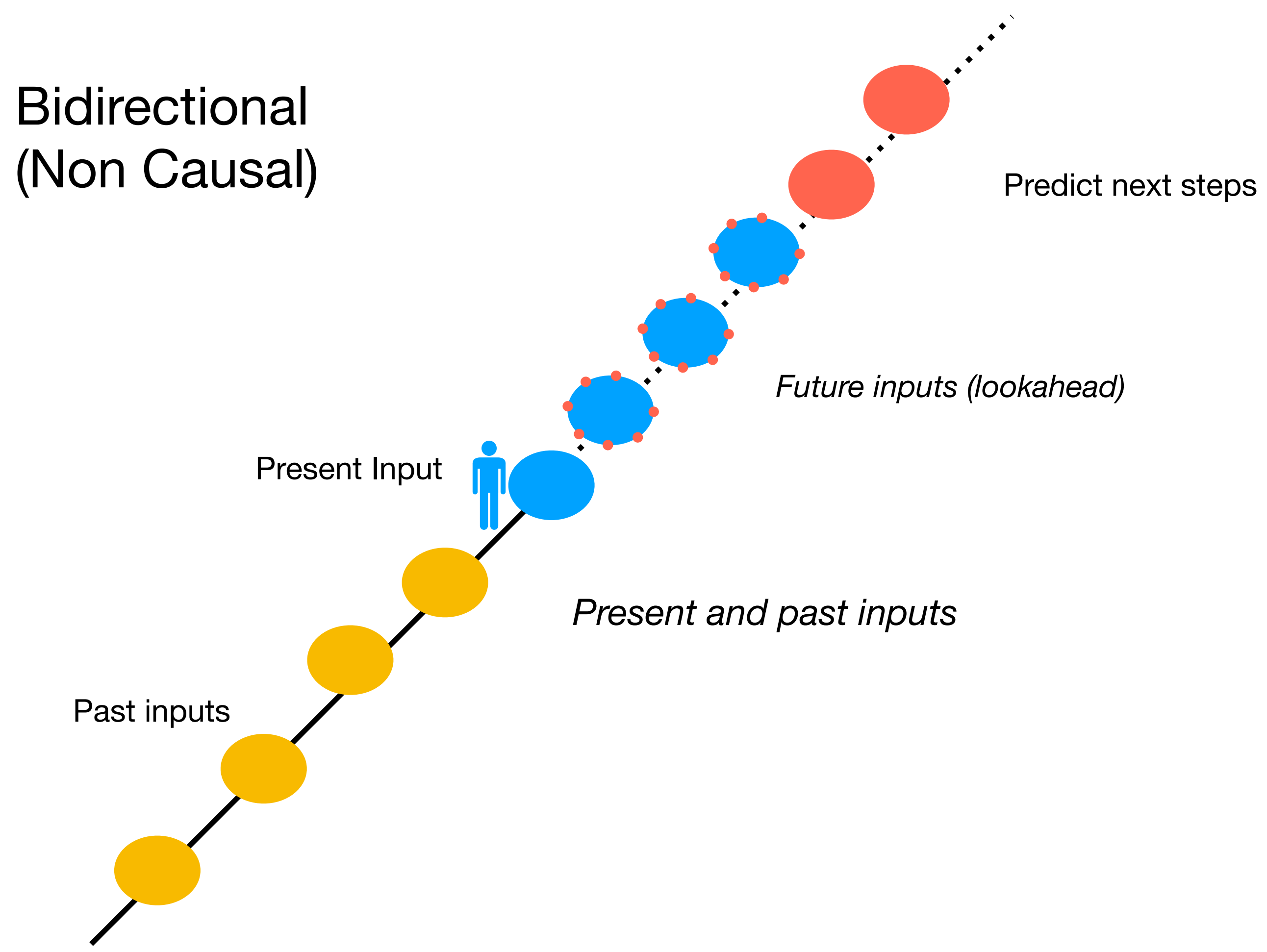



Output vocab length could be very long
“I drink milk”
Model
Postprocess
“Je bois du lait”

Handling large output vocab

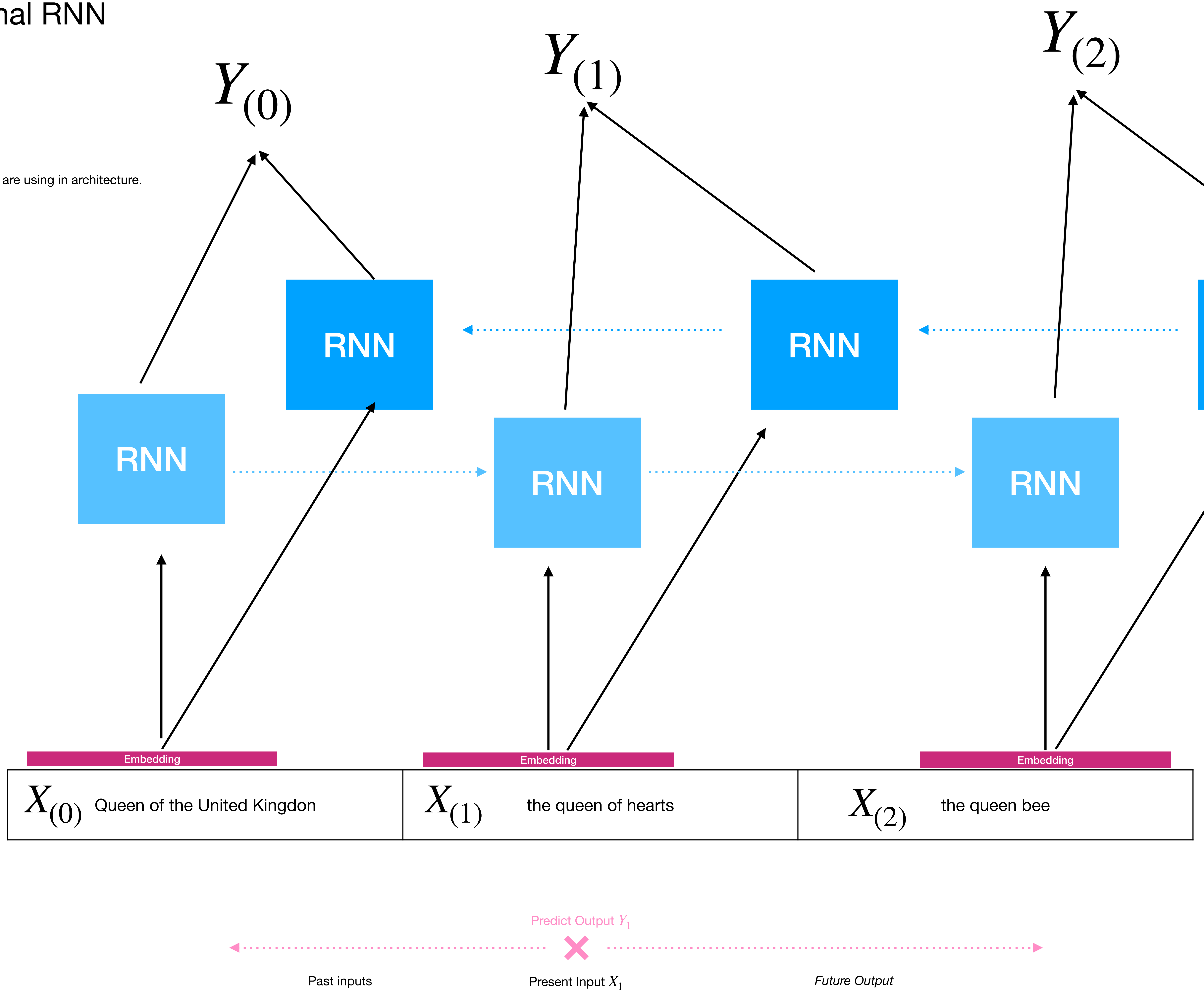



Regular RNN Layer
(Causal)
Predict next steps
Present Input
Present and past inputs determine next steps
Past inputs

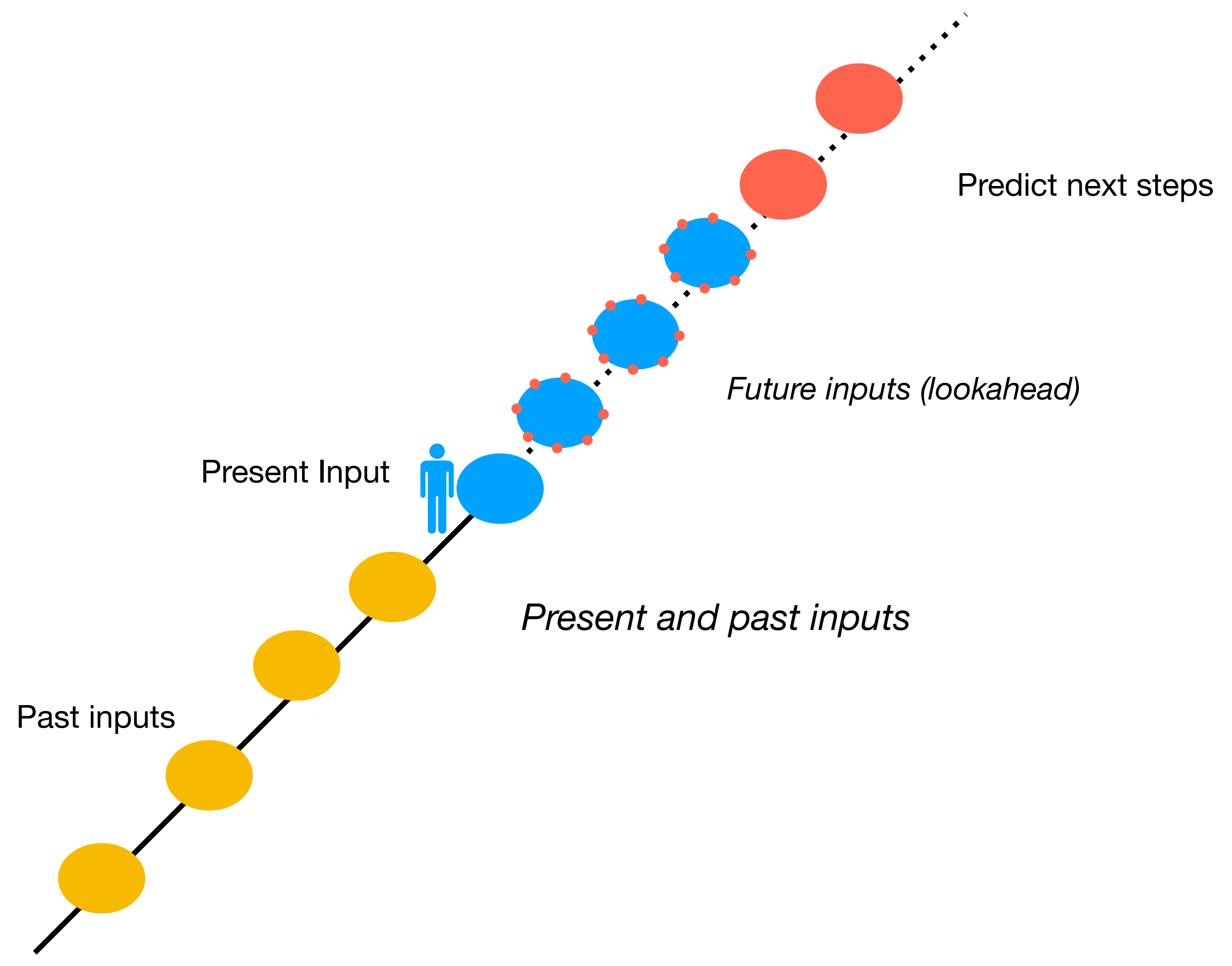

Bidirectional
(Non Causal)
Predict next steps
Future inputs (lookahead)
Present Input
Present and past inputs
Past inputs

Bidirectional RNN

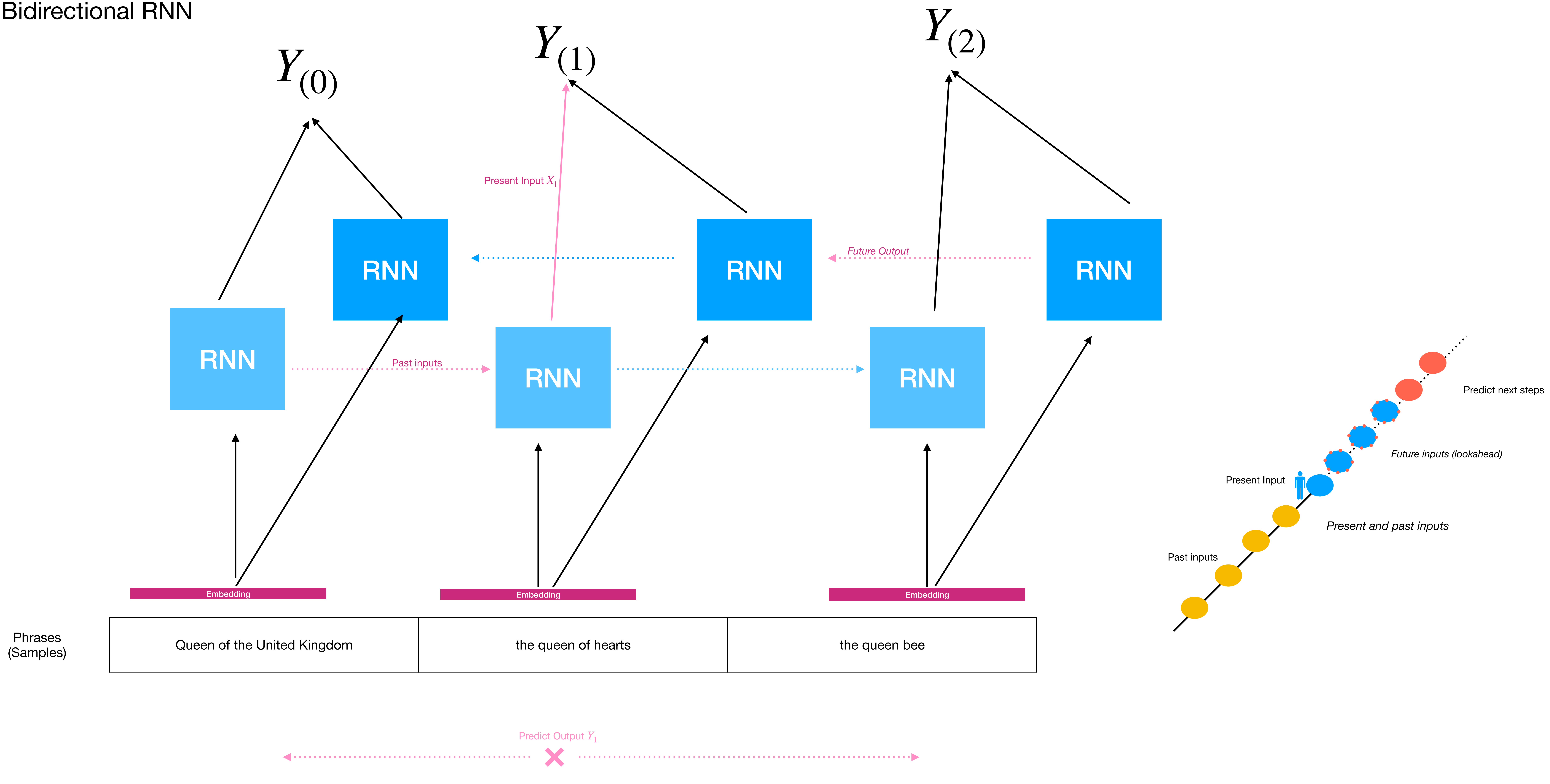
Note: Only two RNN are using in architecture.
Recursive unit!



Cloned RNN in reverse direction

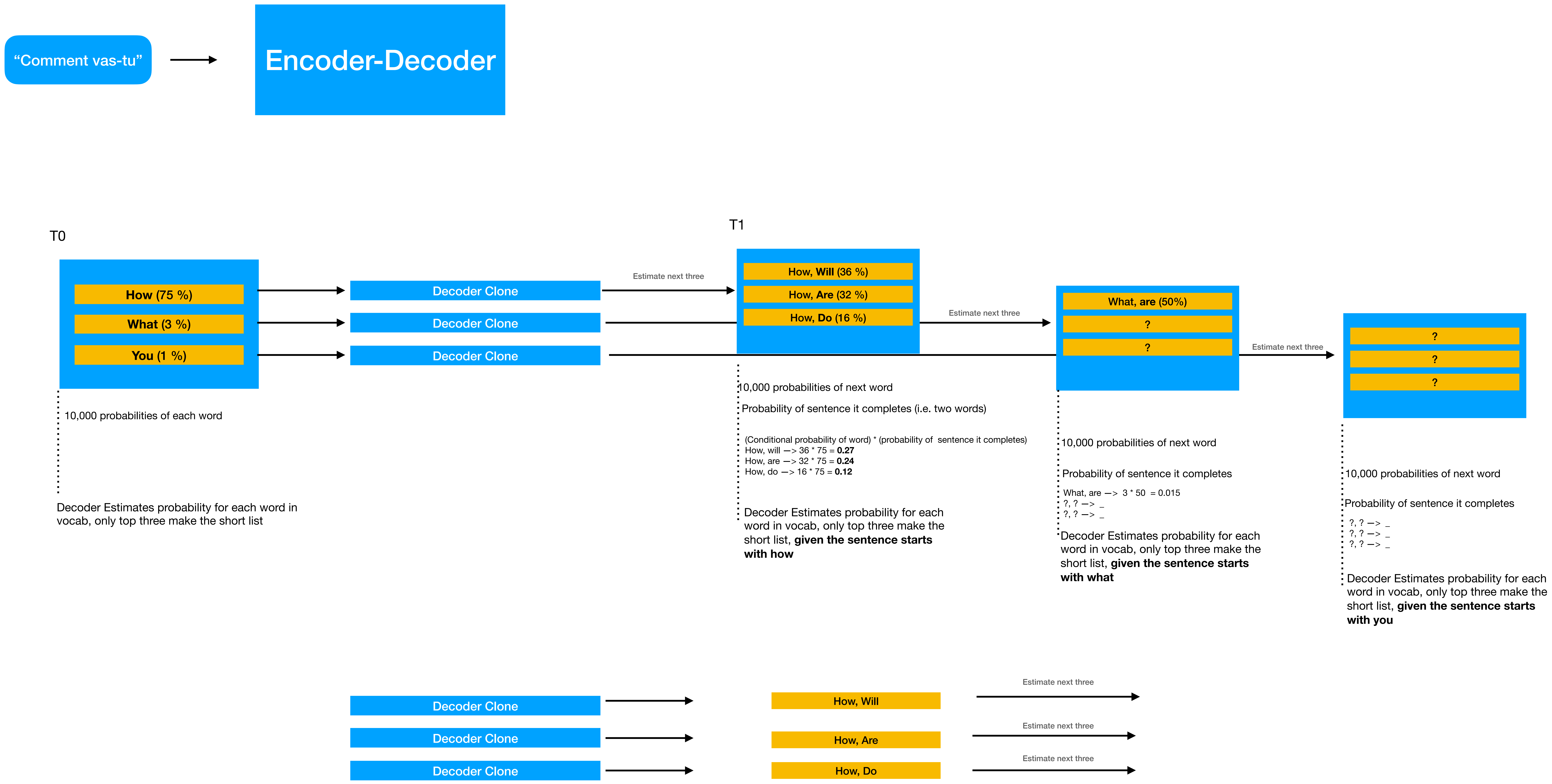


Bidirectional RNN

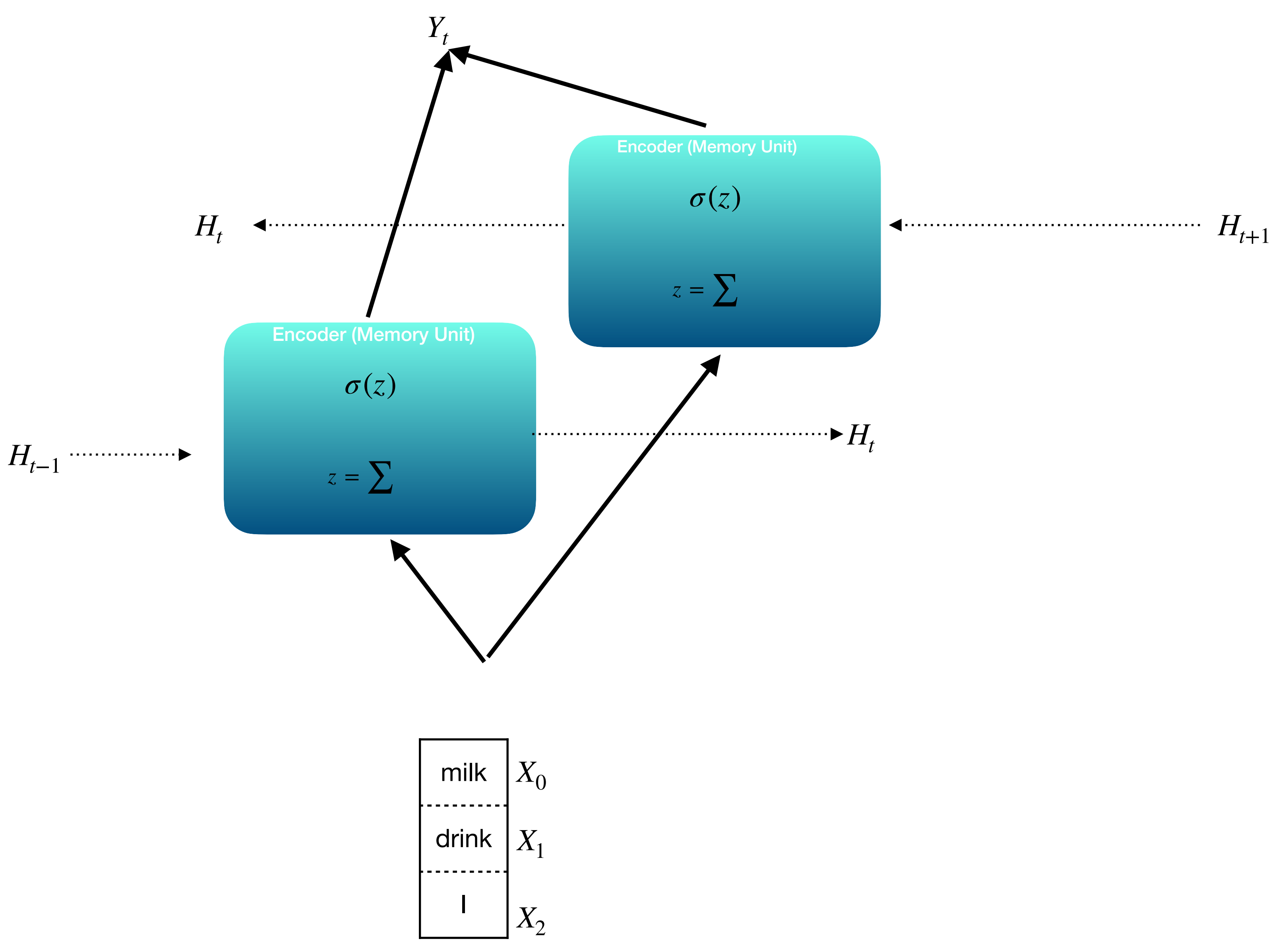


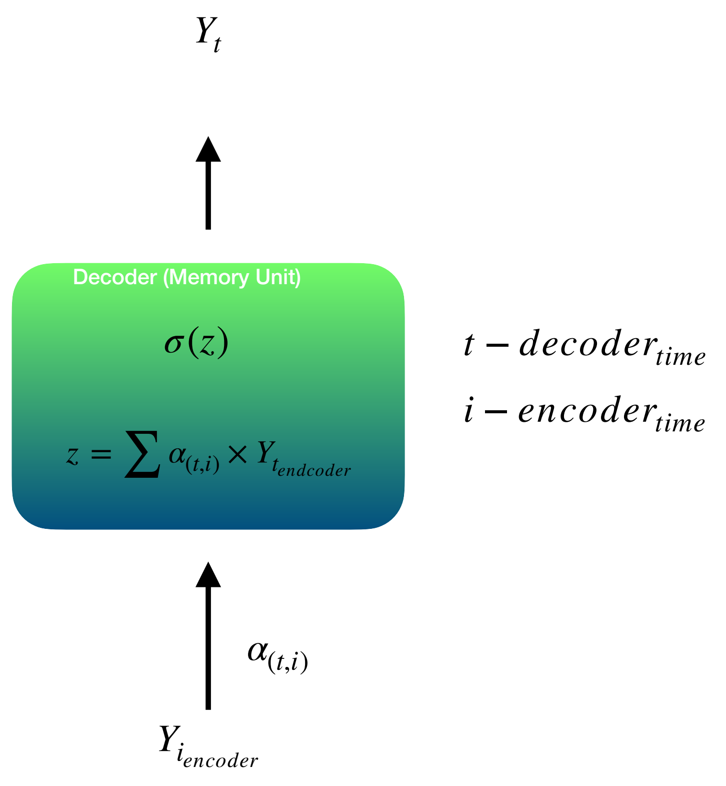
Learns next probable sentence by learning from current, present, and future phrases in different forms of English text

Beam Search (n=3)

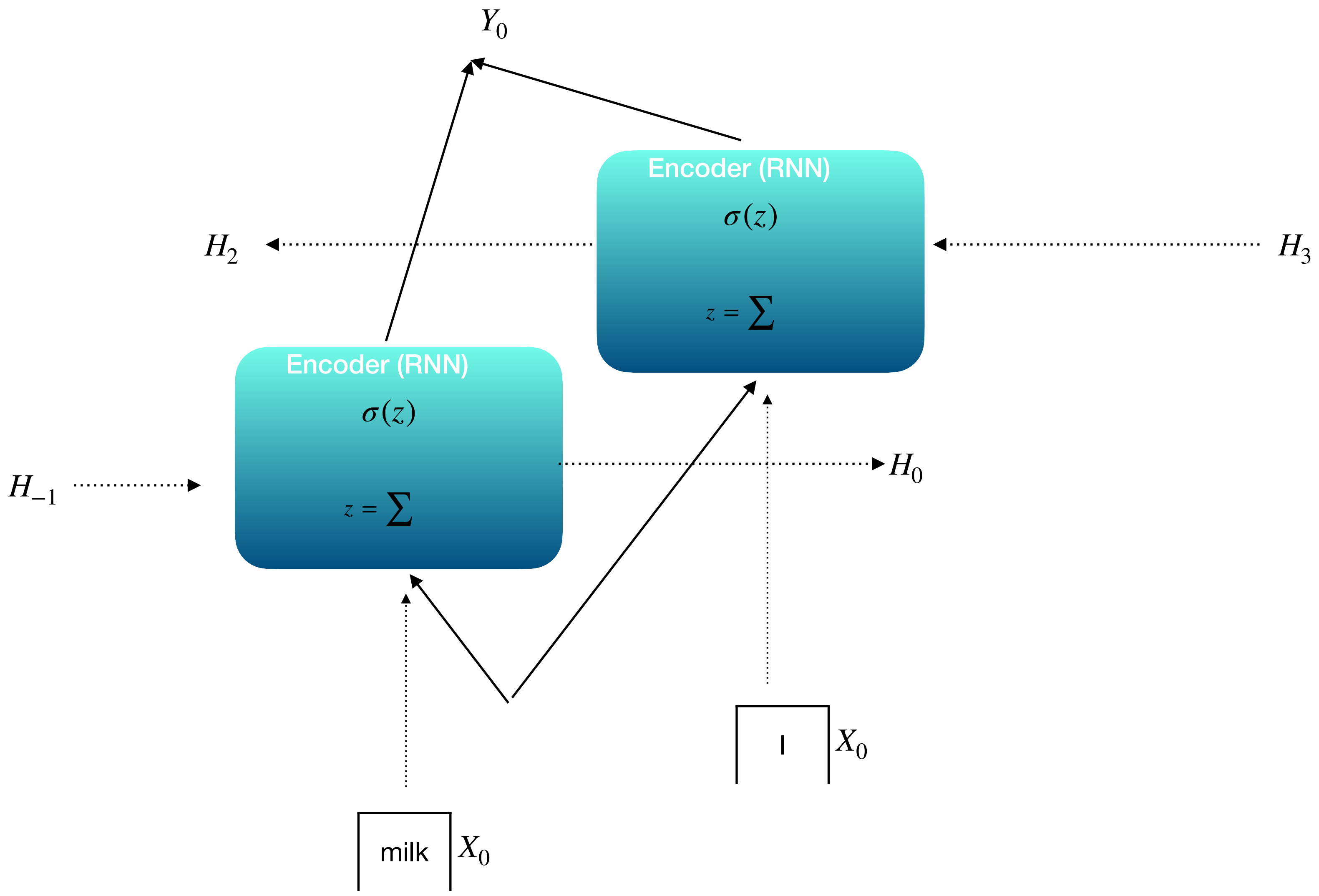


Attention Mechanisms

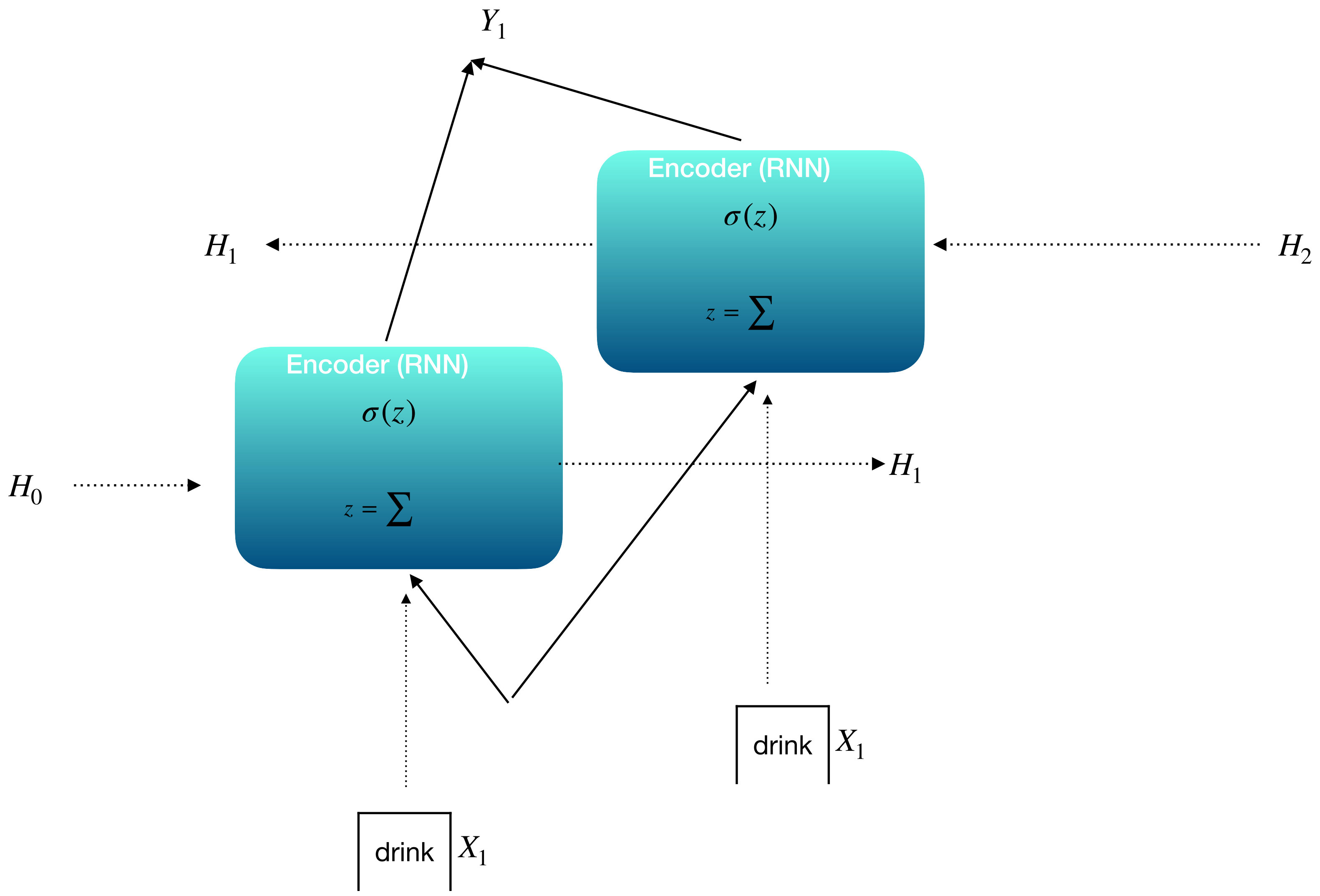




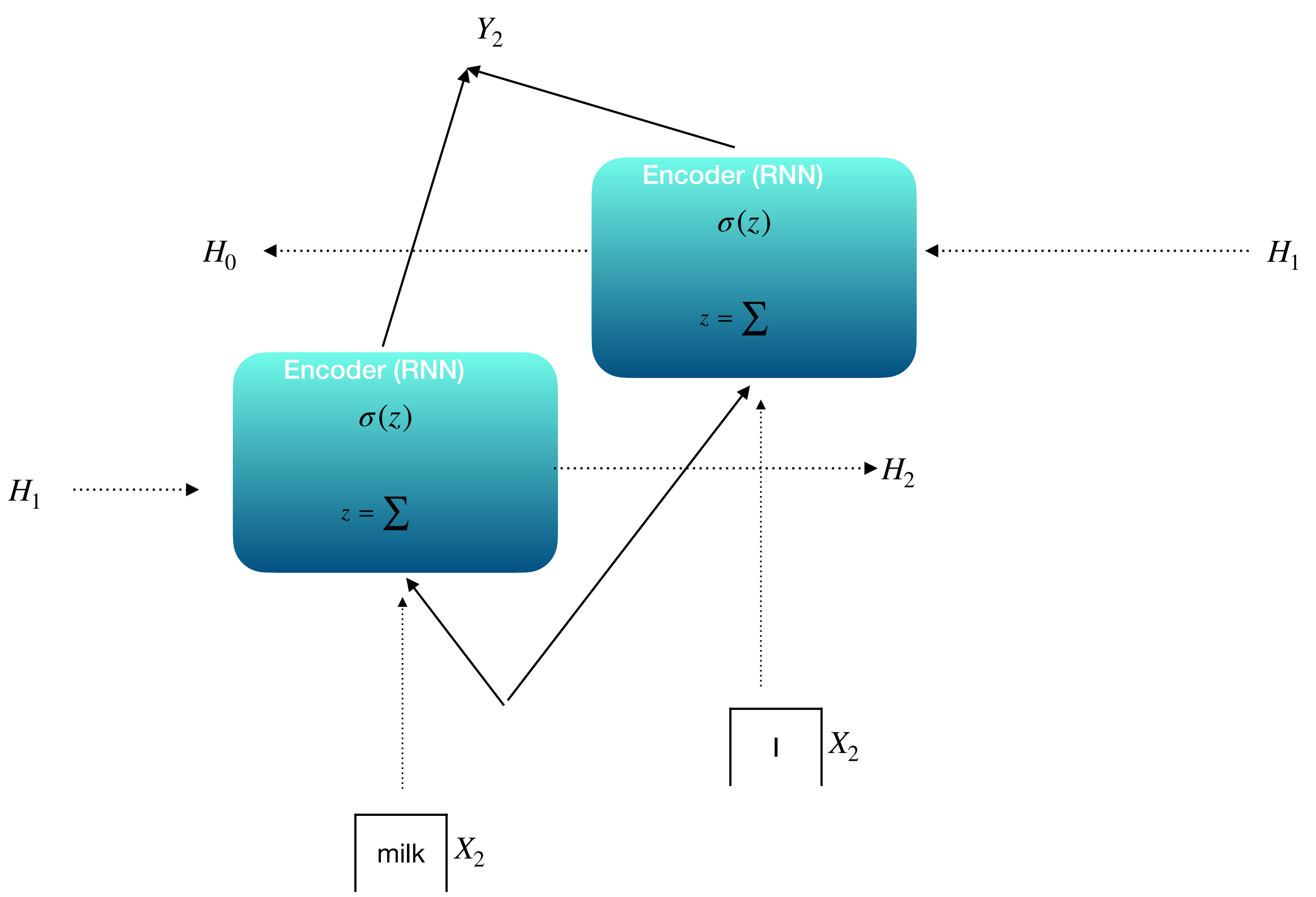
Attention Mechanisms



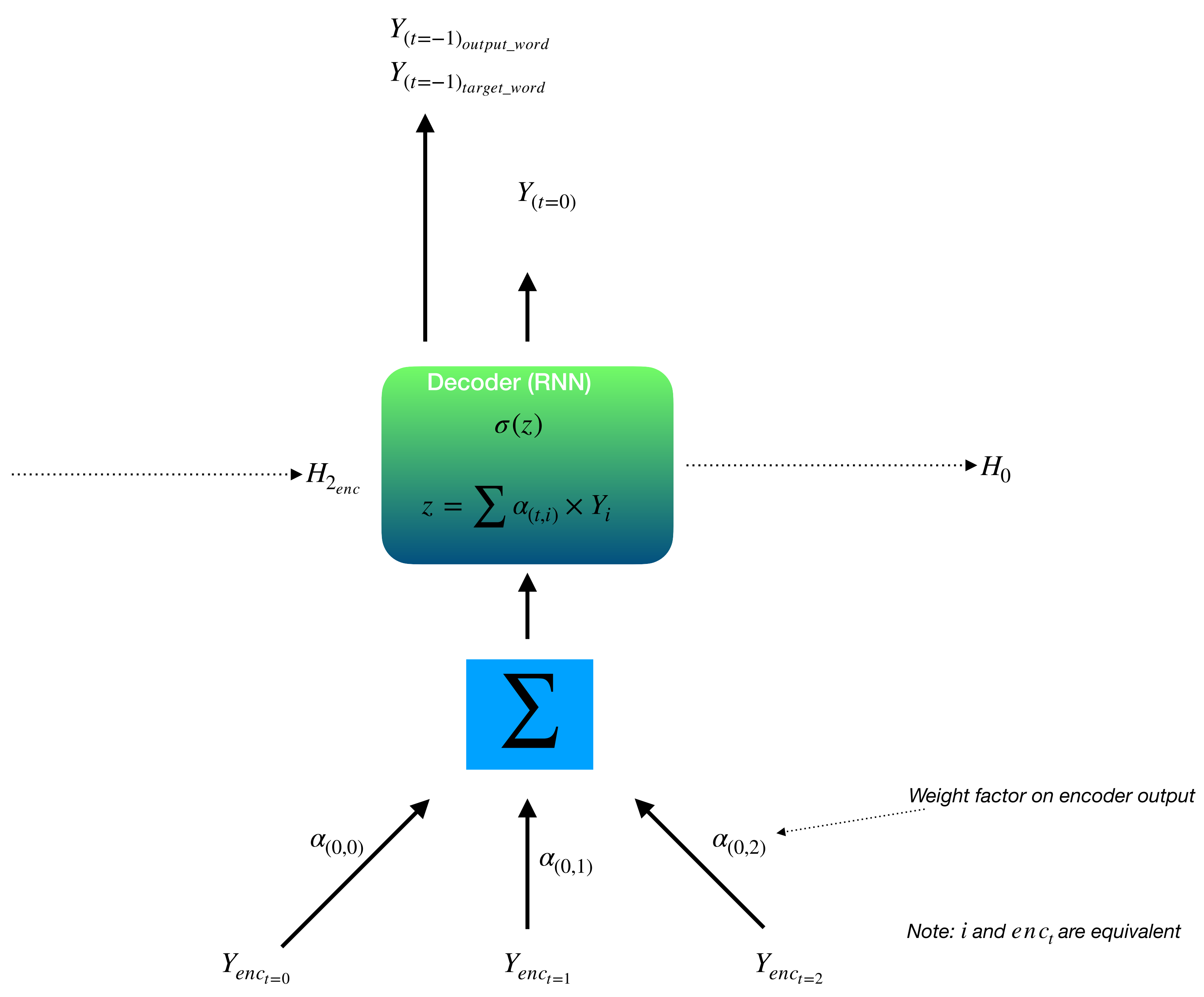
Attention Mechanisms



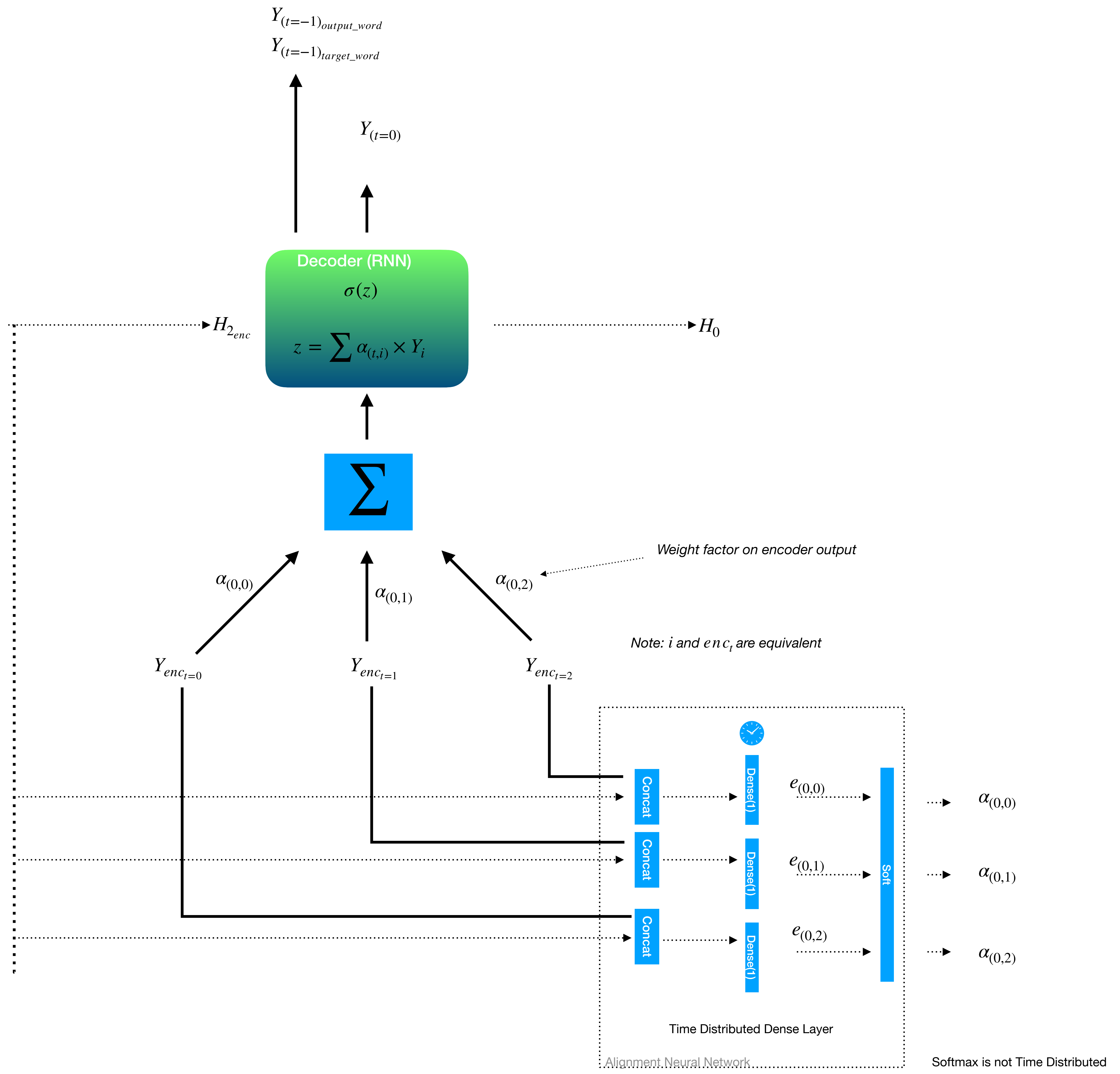
Attention Mechanisms



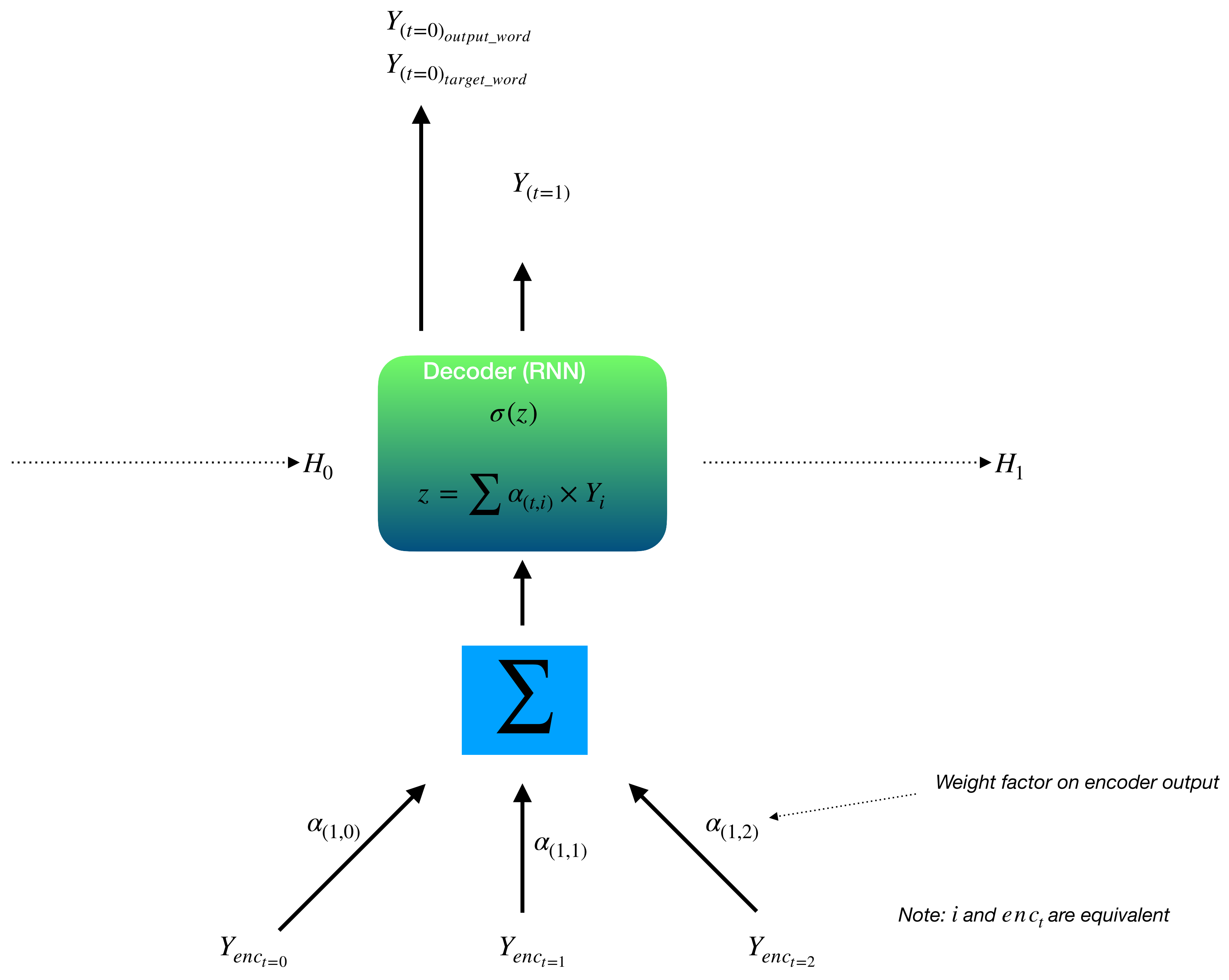
Attention Mechanisms



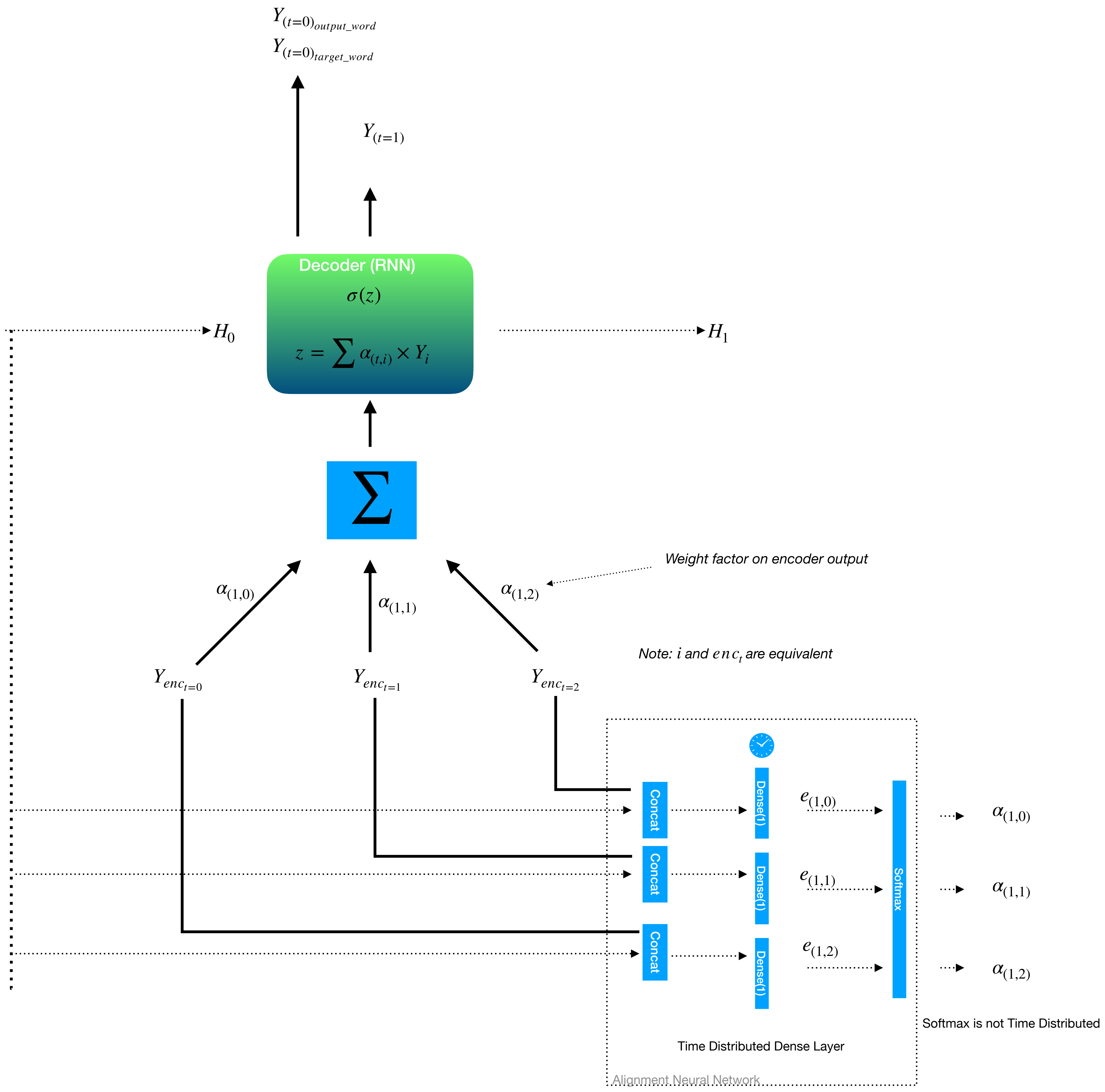
Attention Mechanisms:
Calculating Weights



Attention Mechanisms

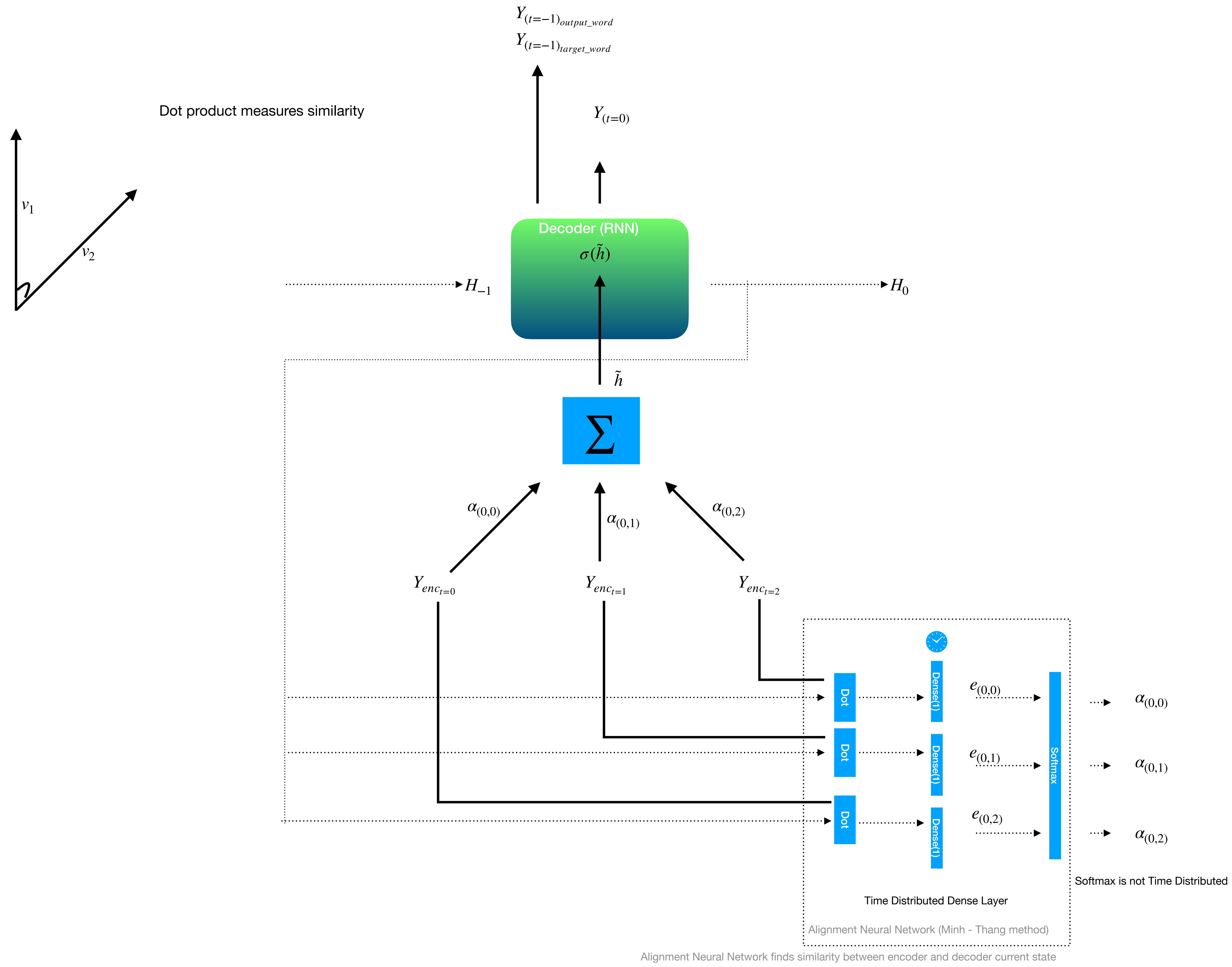


Attention Mechanisms:
Calculating Weights

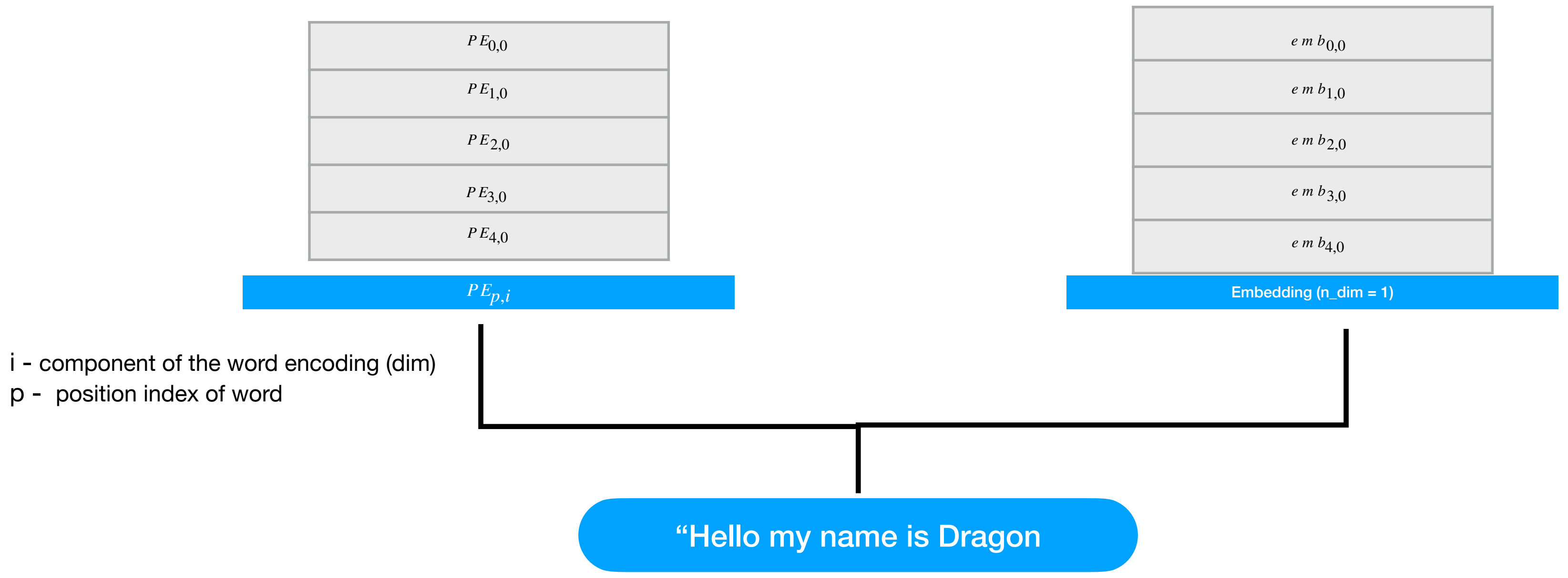


Alignment Neural Network finds similarity between encoders and decoders previous hidden state

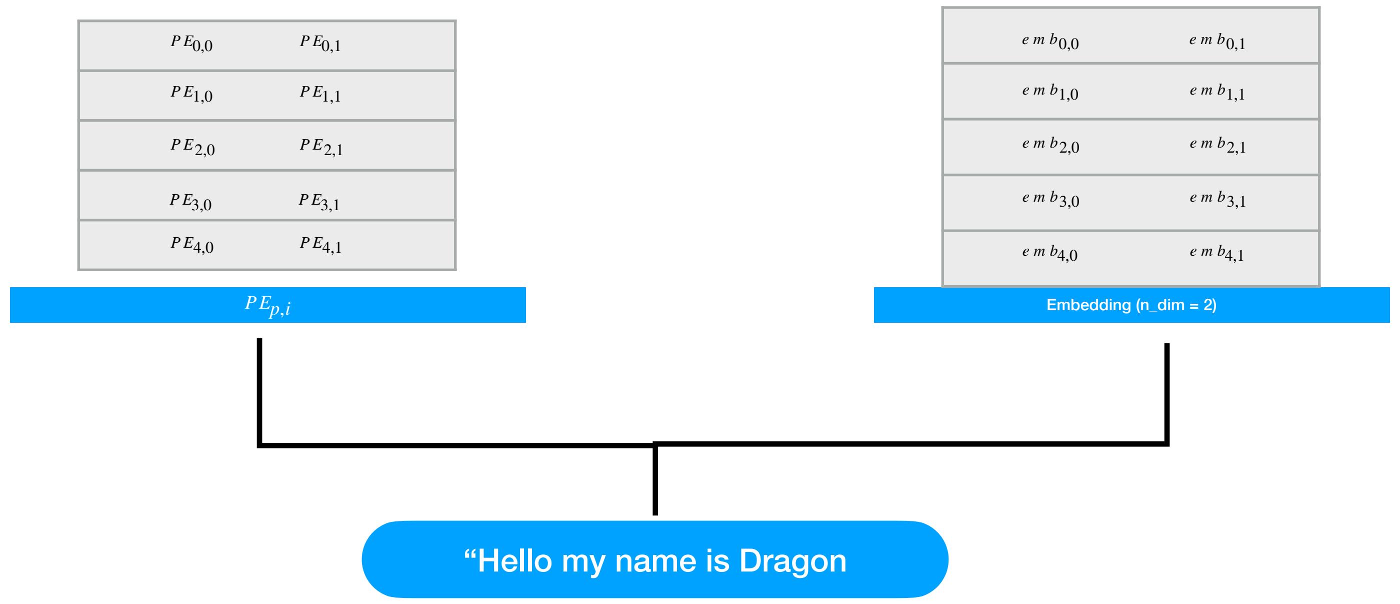
Attention Mechanisms
(Method 2)



Positional Encodings

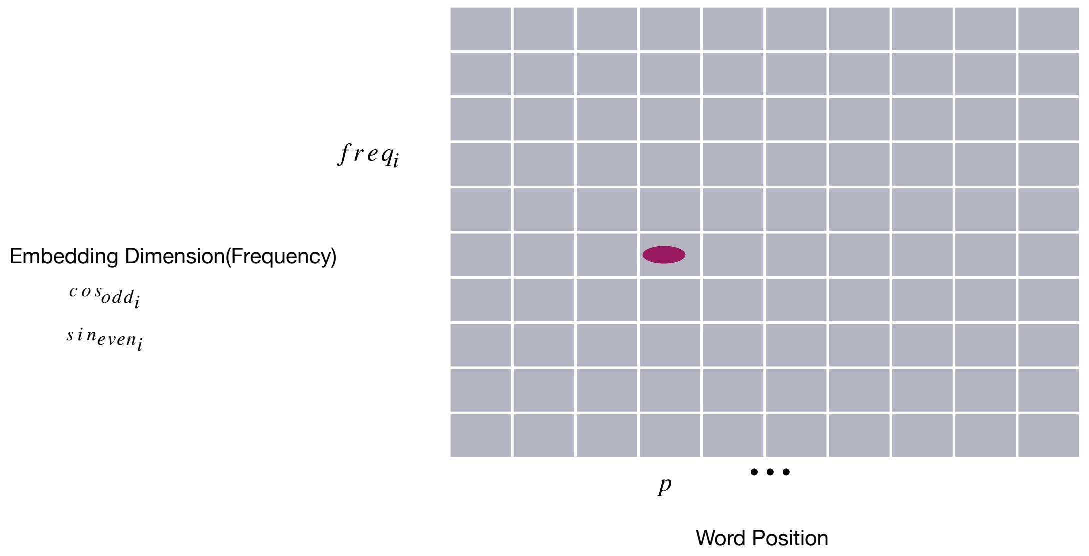


$$PE_{p,i} = \begin{cases} sin(p/10000^{i/d}) & \text{if i is even} \\ cos(p/10000^{(i-1)/d}) & \text{if i is odd} \end{cases}$$

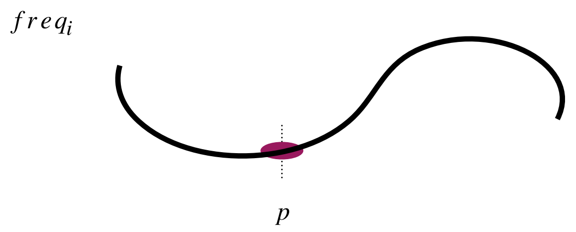


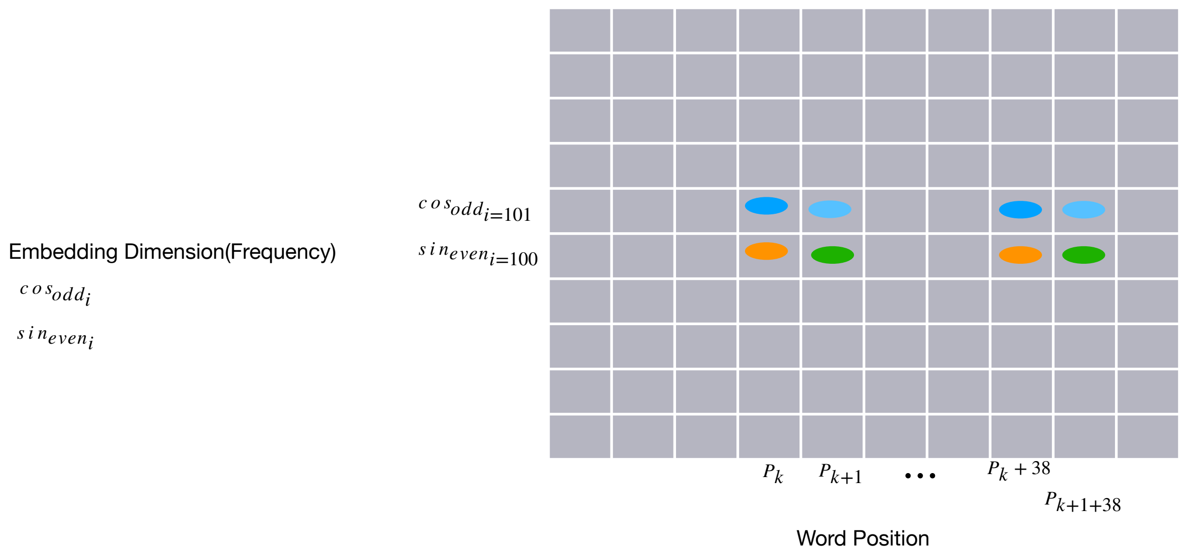
Positional Encodings

$$PE_{p,i} = \begin{cases} sin(p/10000^{i/d}) & \text{if i is even} \\ cos(p/10000^{(i-1)/d}) & \text{if i is odd} \end{cases}$$



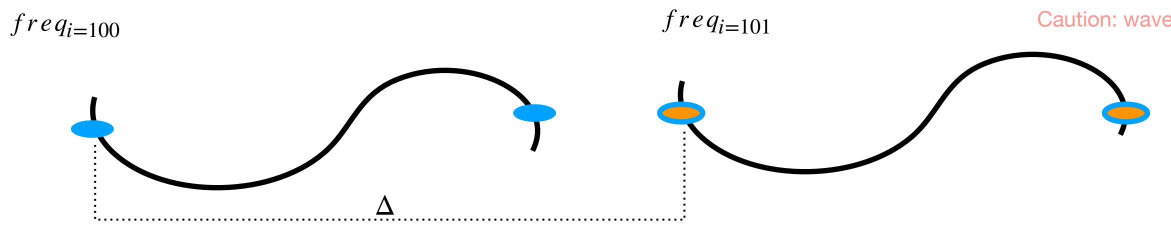
Note: Unique positional encoding at different frequencies (i) and positions(p)



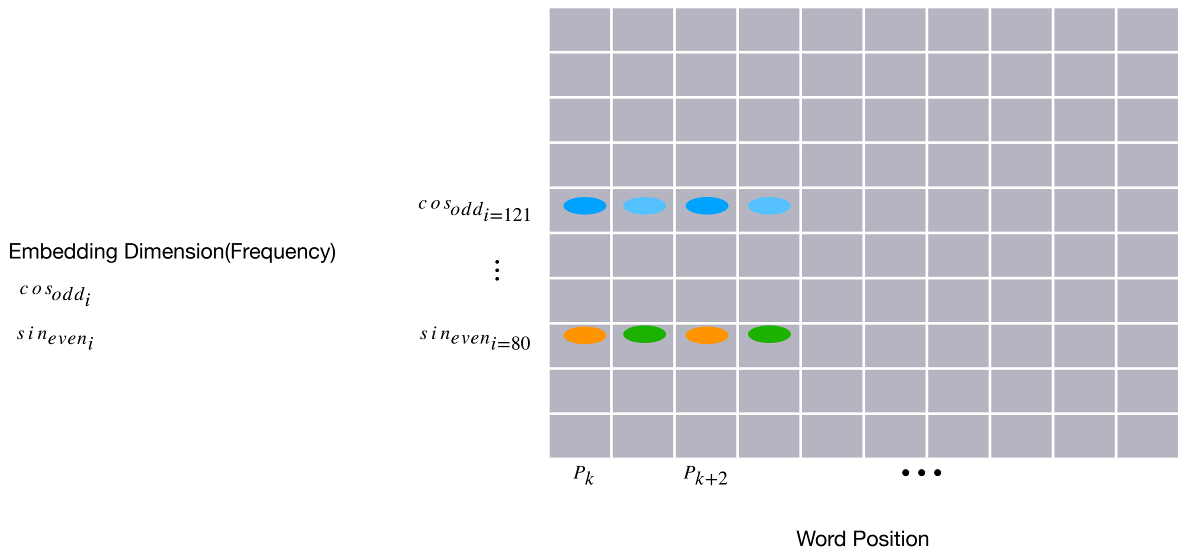


Words located 38 word distances apart have a positional encoding in dimensions 100 and 101

Caution: wave are not representative of cos and sin waves; strictly placeholders



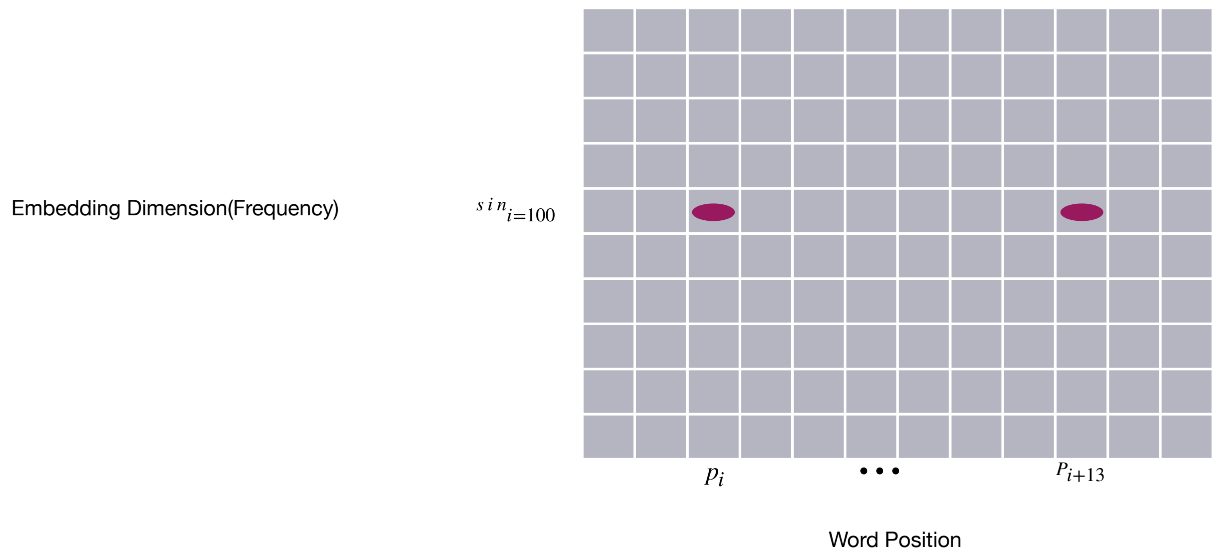
Both sin and cos required for PE encoding to generate unique relative word positions.

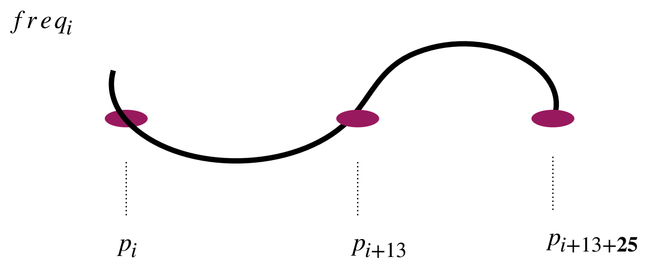


Words located 2 word distances apart have a positional encoding in dimensions 80 and 121

Positional Encodings

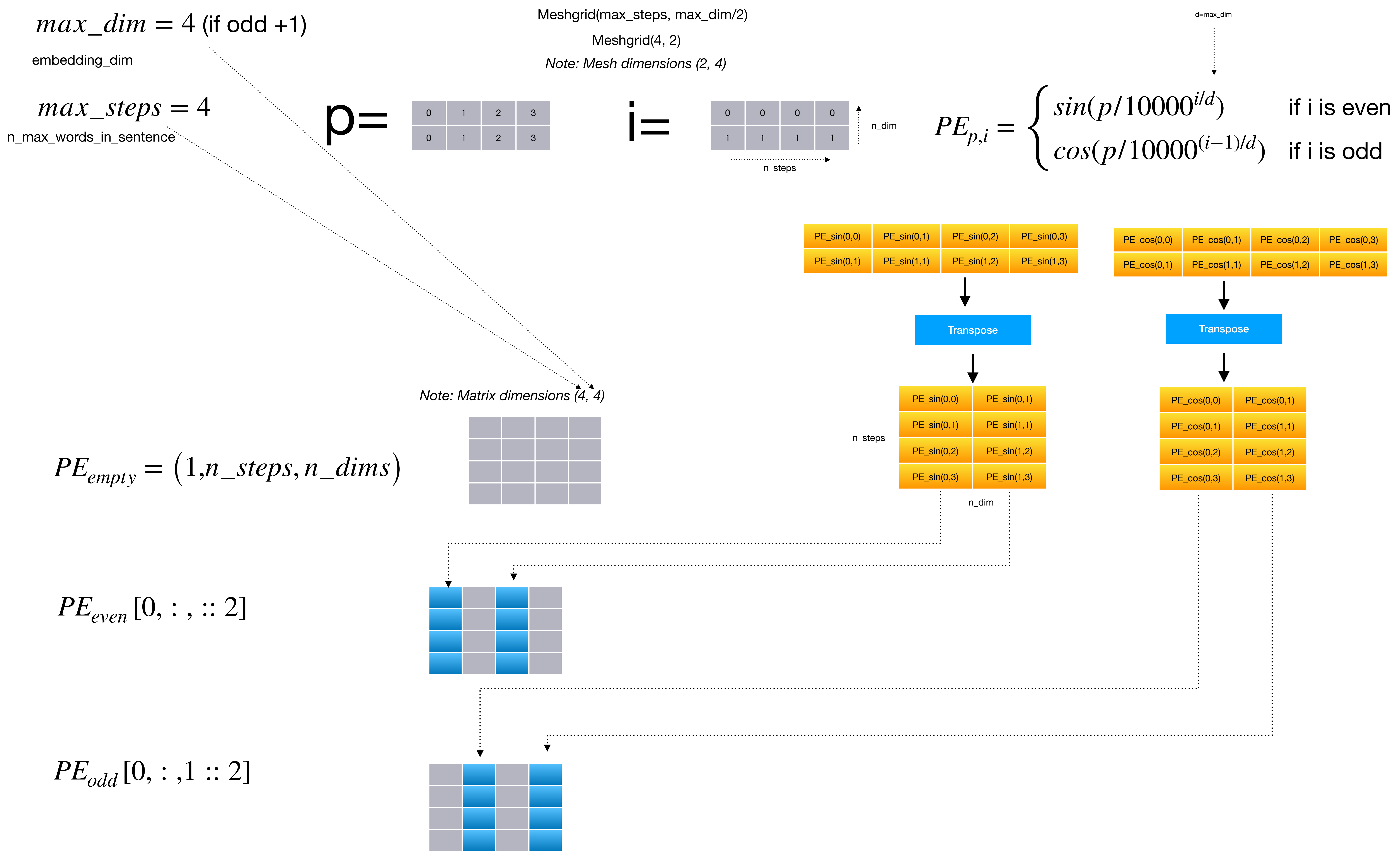
$$PE_{p,i} = \begin{cases} sin(p/10000^{i/d}) \\ \cancel{cos(p/10000^{(i-1)/d})} \end{cases} \quad \text{for all}$$



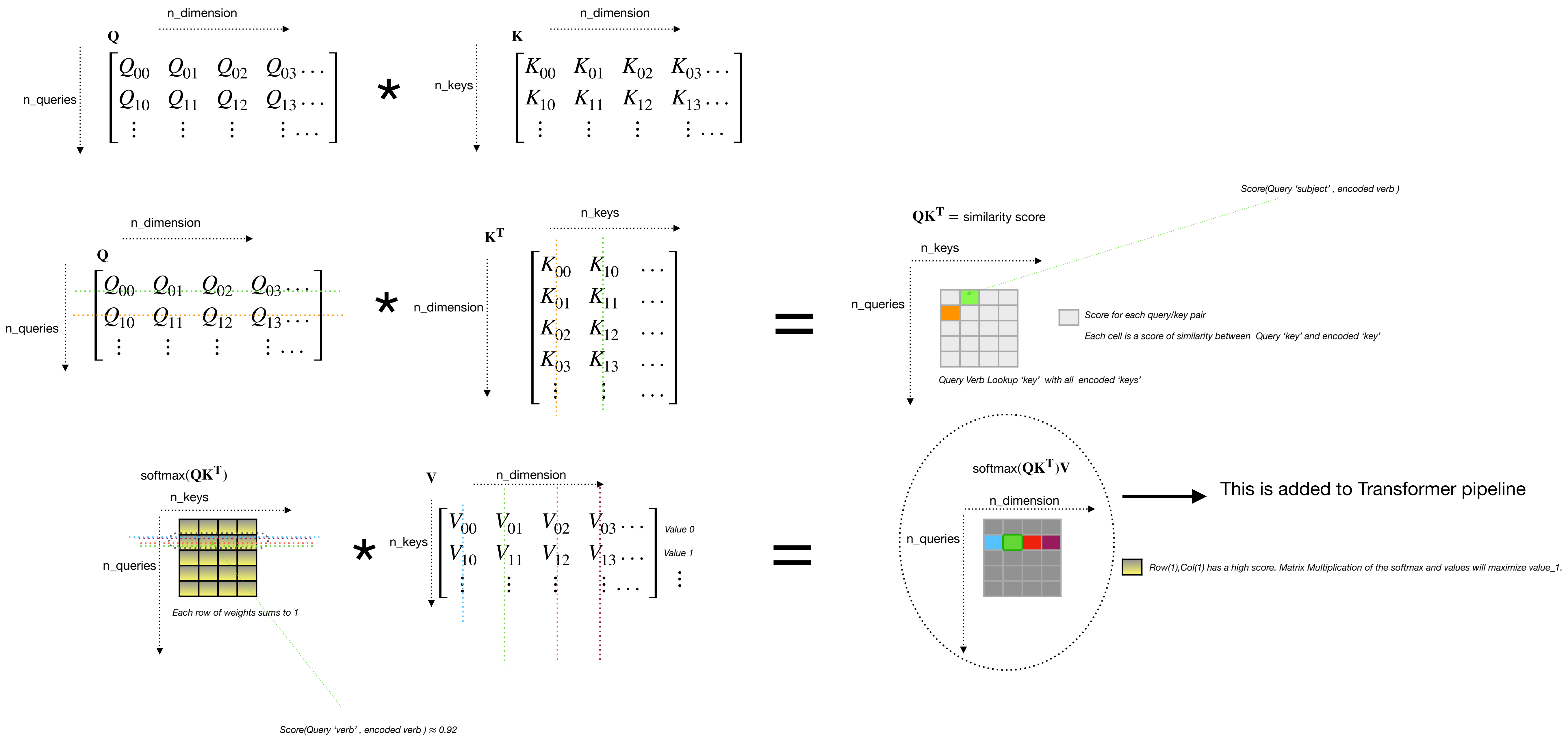


Deltas are not unique. This is an example of aliasing. Model cannot learn from ambiguous encoding.

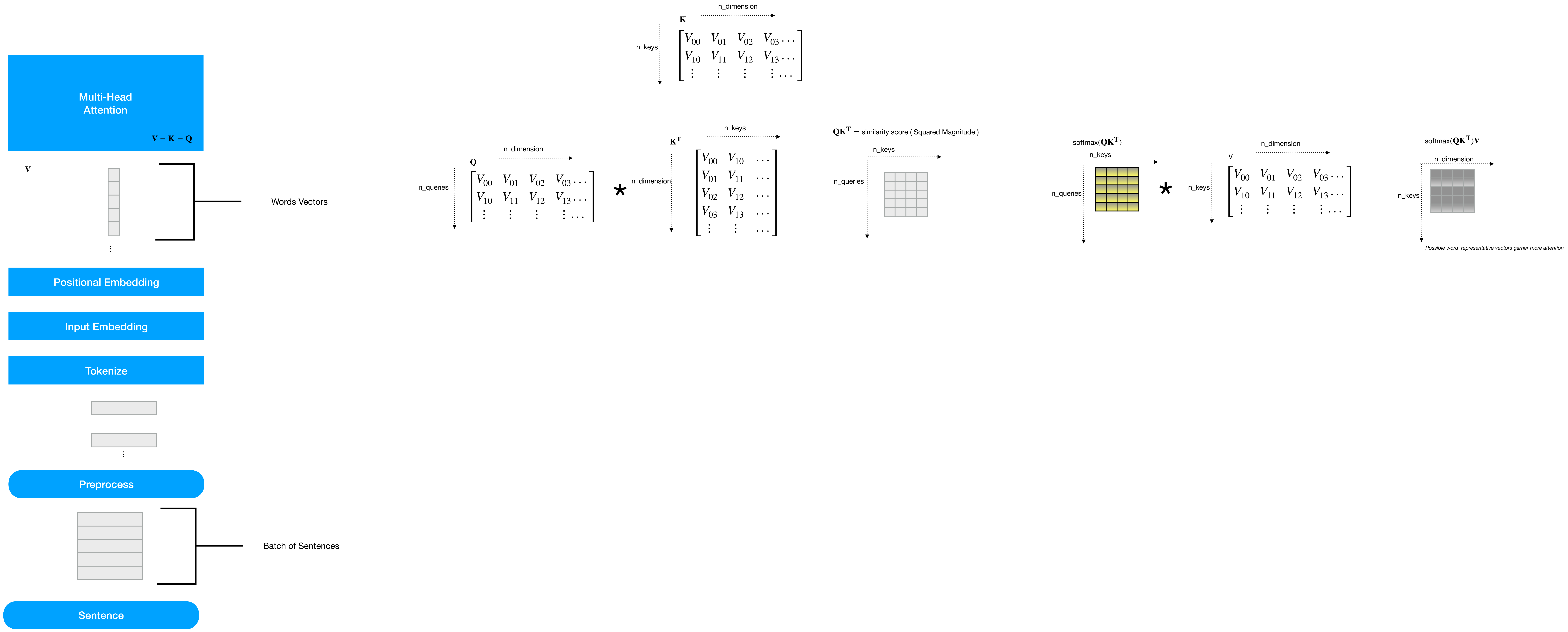
Positional Encodings : Simple Example



MultiHead Attention Layer

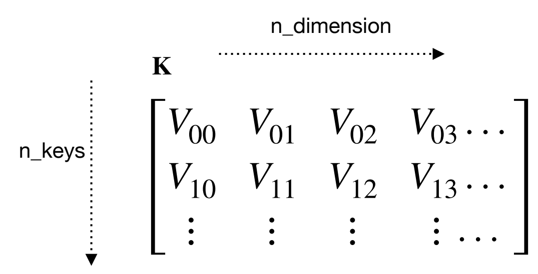


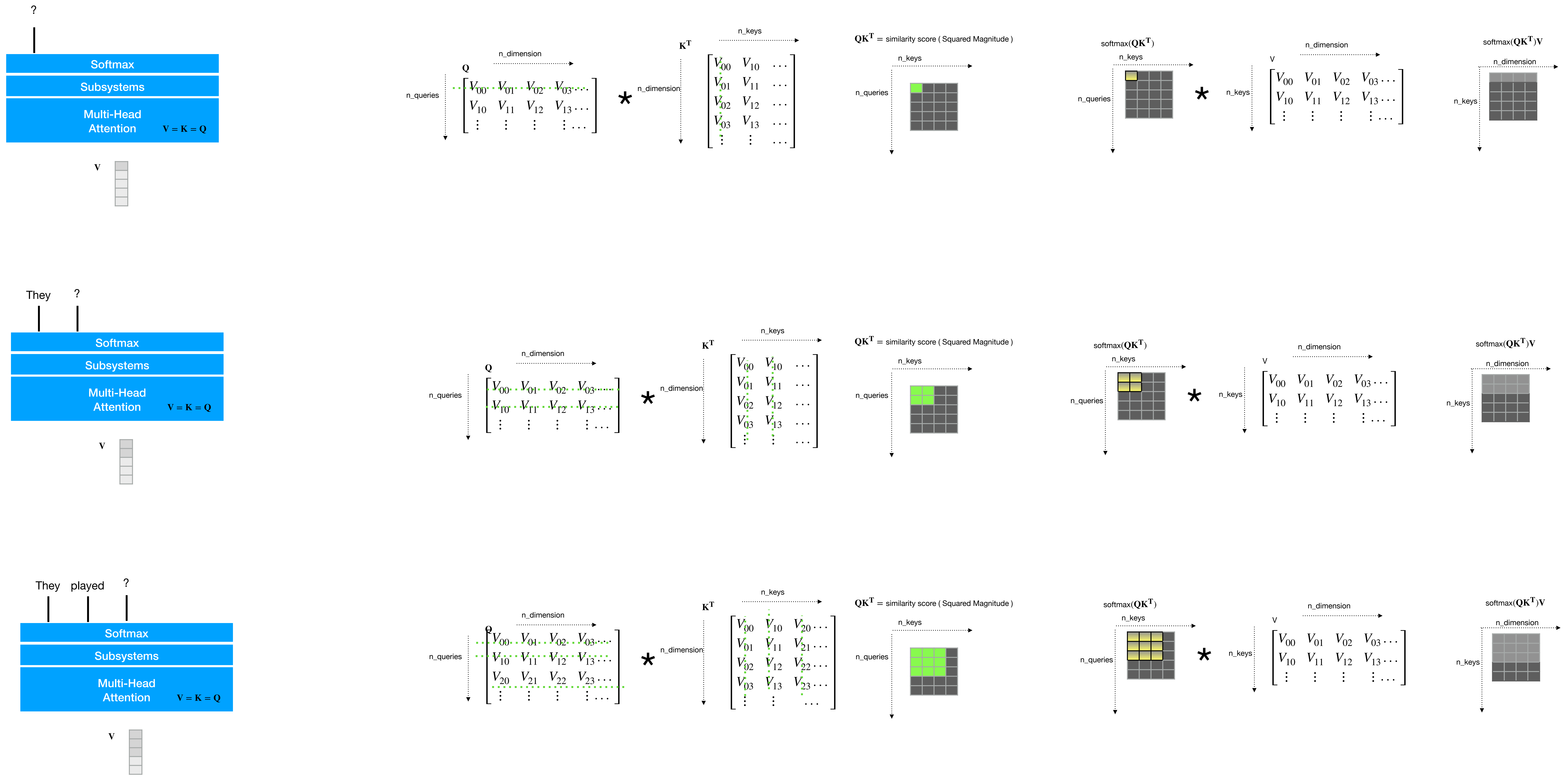
MultiHead Attention Layer : Encoder Subsystem



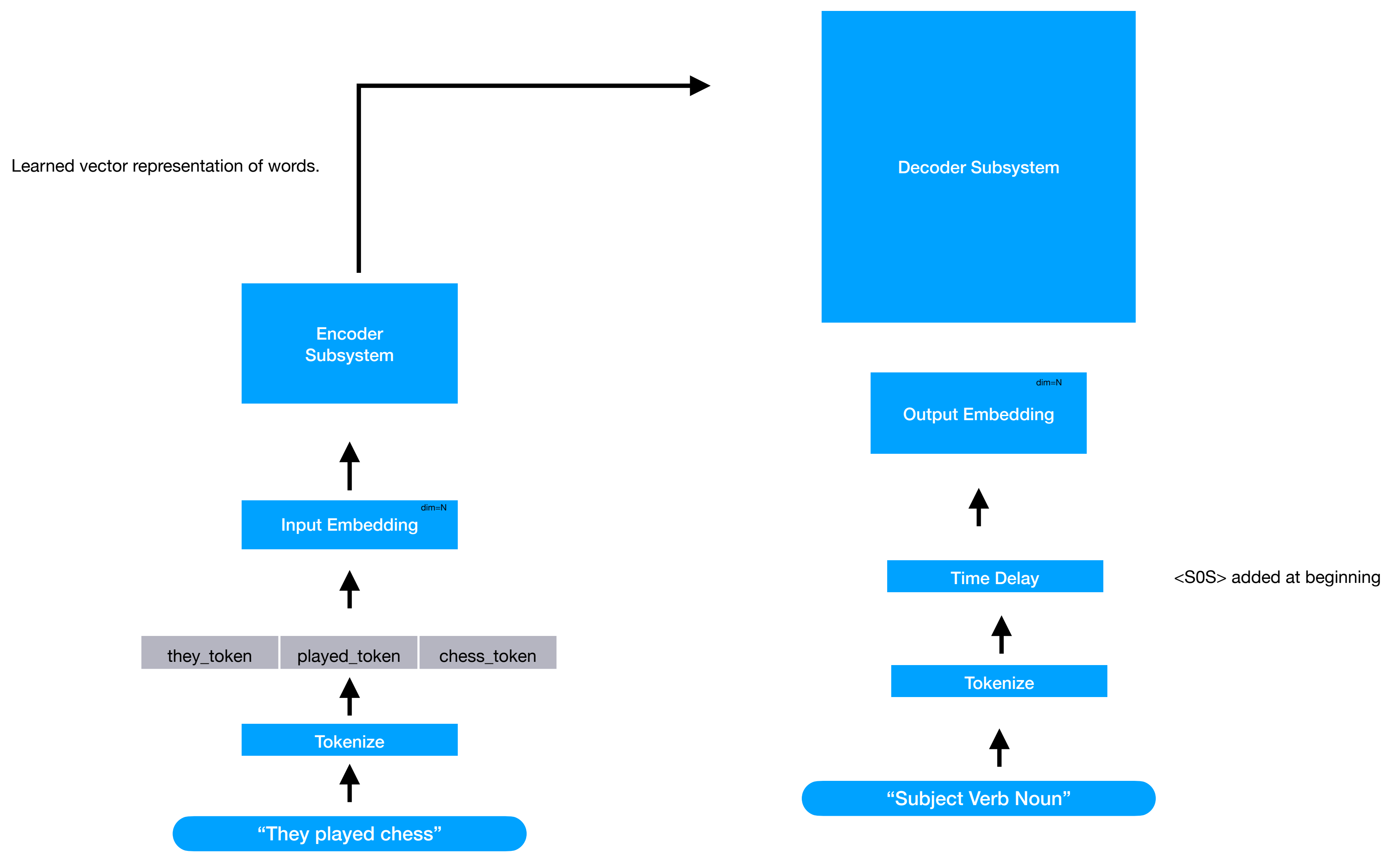
MaskedMultiHead Attention Layer : Decoder Subsystem

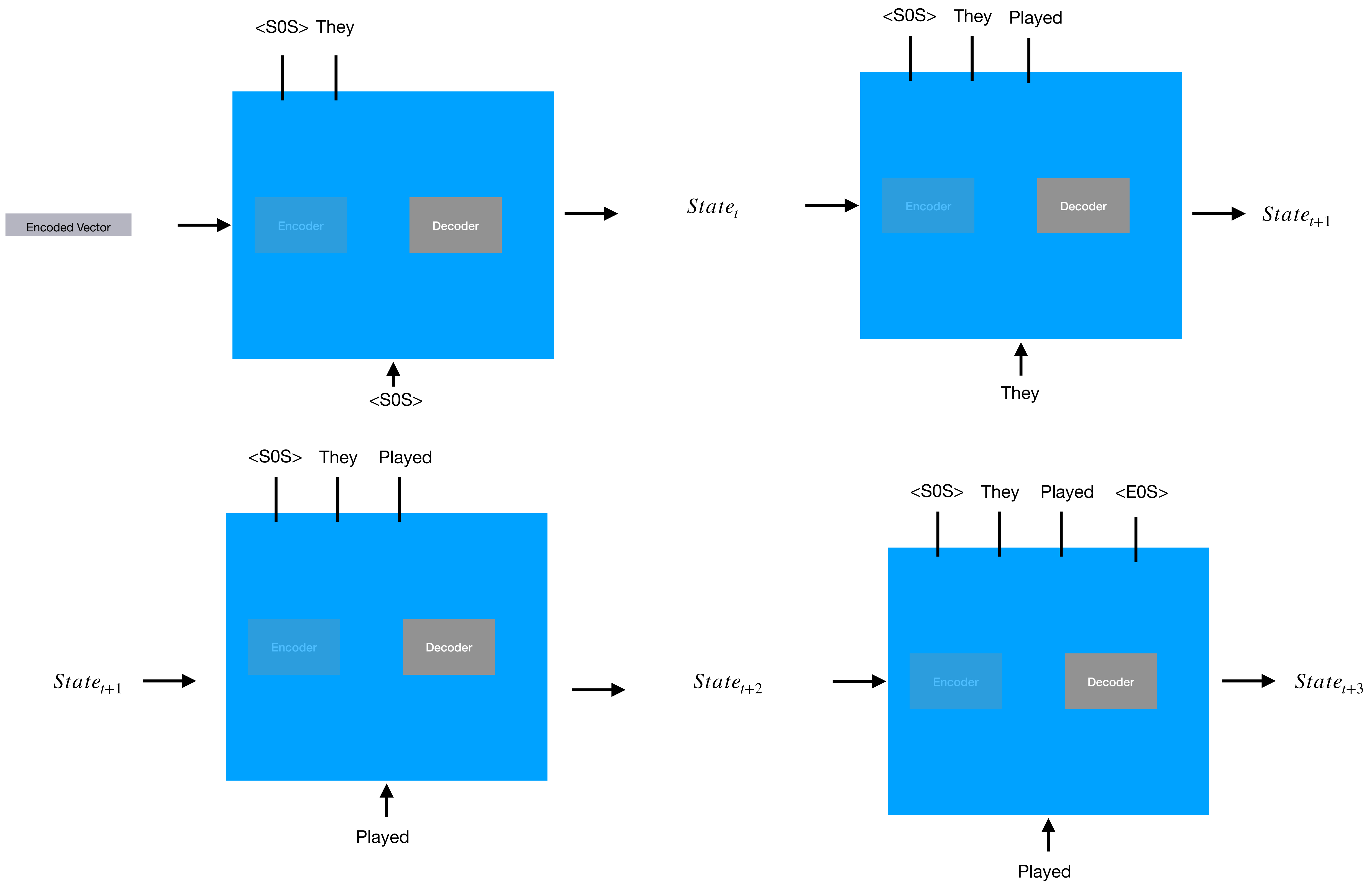
Mask MH Attention. Words cannot compare to words in the future.
Prevent a word from comparing itself to words located after it.
Masking can be done by adding large negative number to
$\mathbf{QK^T}$ = similarity score (Squared Magnitude)



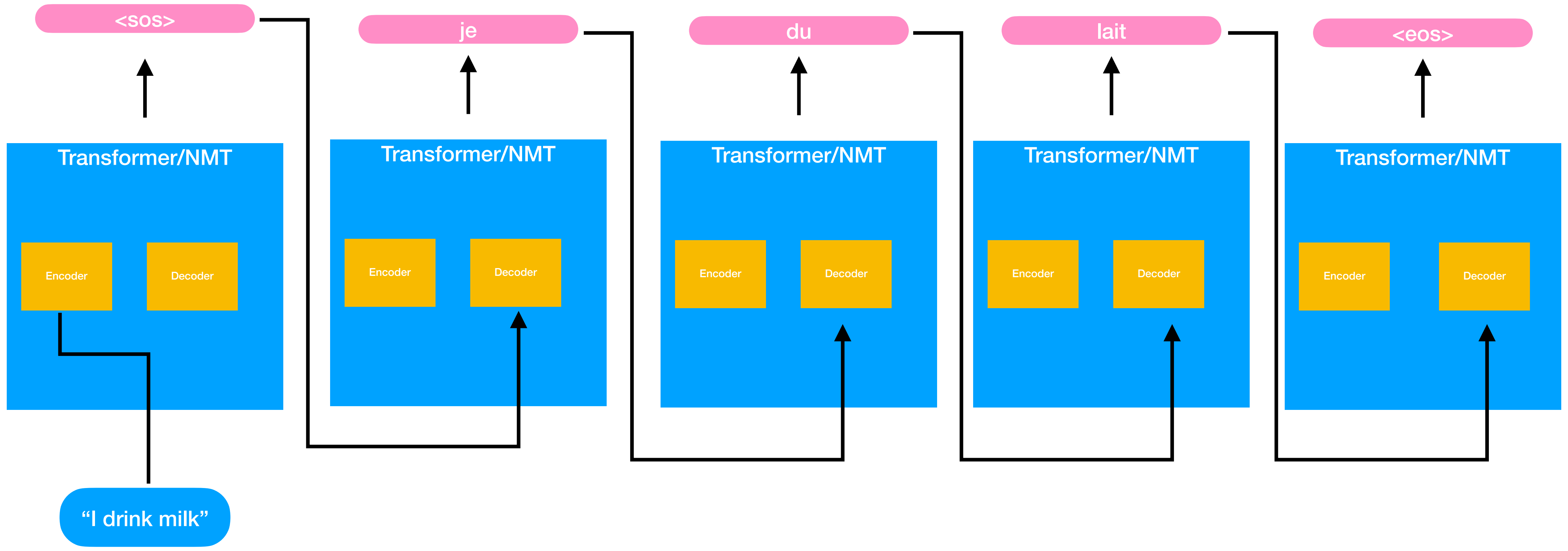


Train/ Infer


Learned vector representation of words.
Decoder Subsystem
Encoder Subsystem
Output Embedding
Input Embedding
Time Delay
<S0S> added at beginning
they_token
played_token
chess_token
Tokenize
Tokenize
"They played chess"
"Subject Verb Noun"

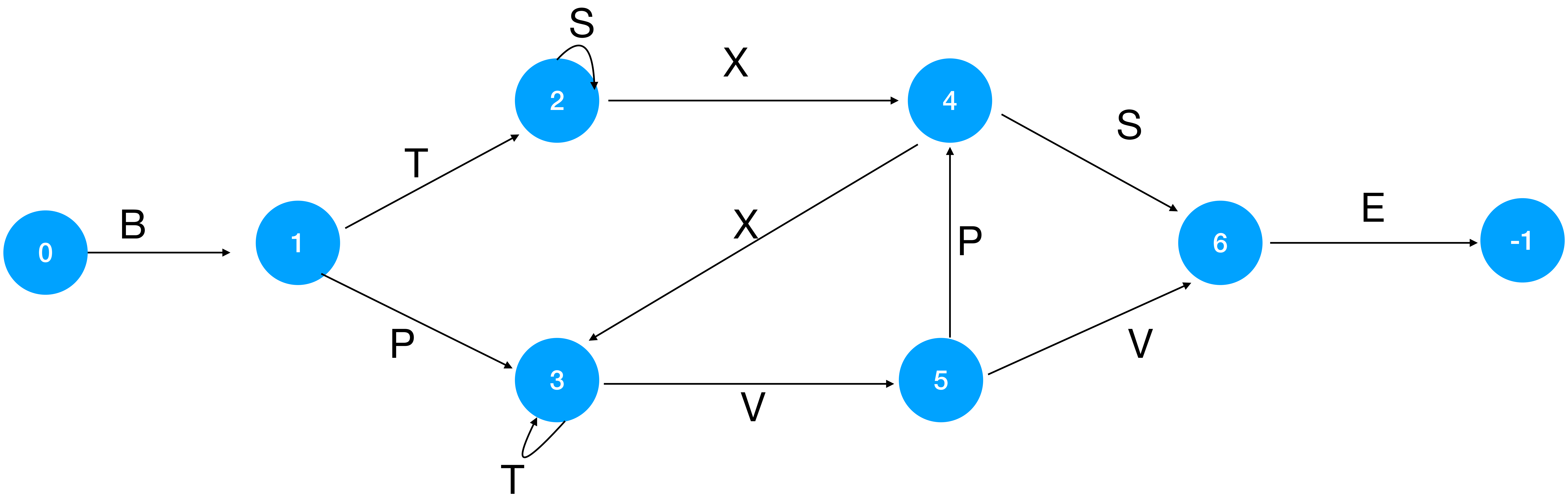

<S0S> They
<S0S> They Played
Encoded Vector
Encoder
Decoder
$State_t$
$State_{t+1}$
<S0S>
They
<S0S> They Played
<S0S> They Played <E0S>
$State_{t+1}$
$State_{t+2}$
$State_{t+3}$
Played
Played

Inference

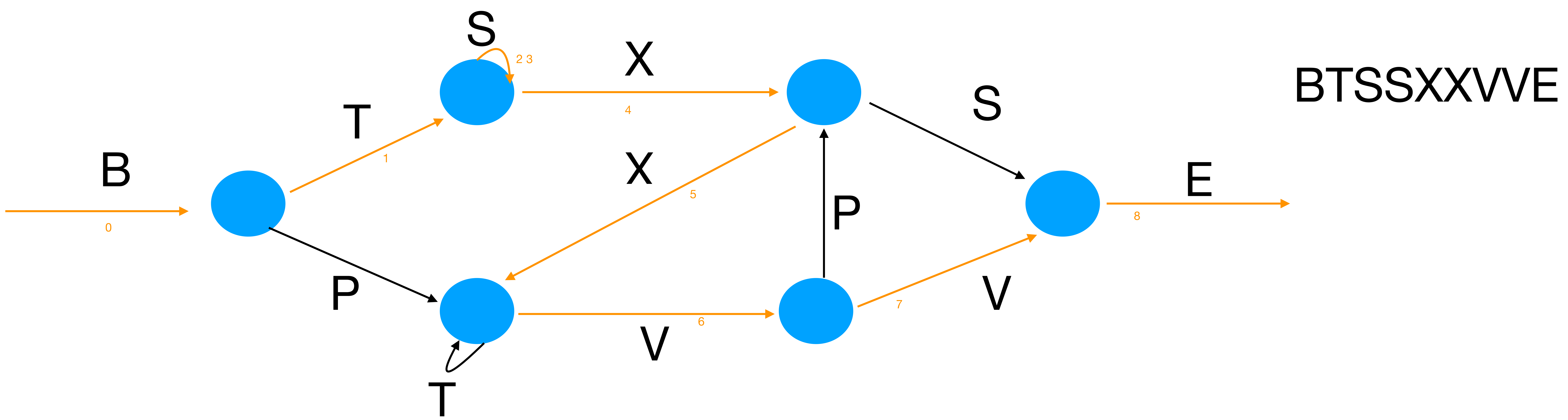


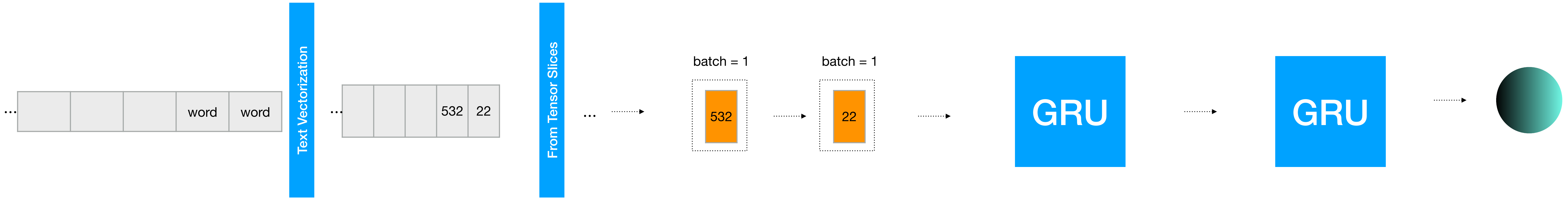
<sos>
je
du
lait
<eos>
Transformer/NMT
Encoder
Decoder
Transformer/NMT
Encoder
Decoder
Transformer/NMT
Encoder
Decoder
Transformer/NMT
Encoder
Decoder
Transformer/NMT
Encoder
Decoder
"I drink milk"

Reber Grammer



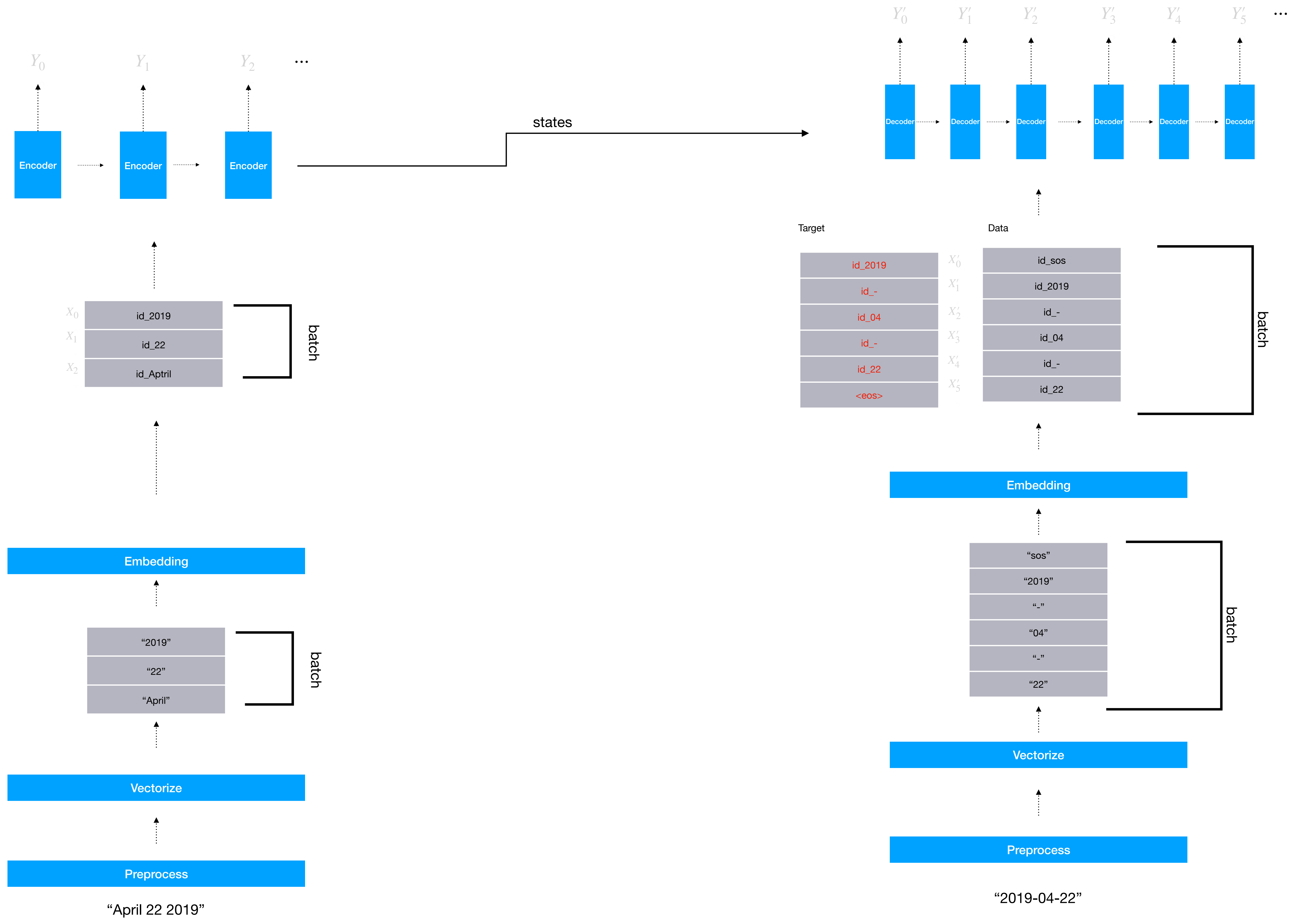
Reber Grammer



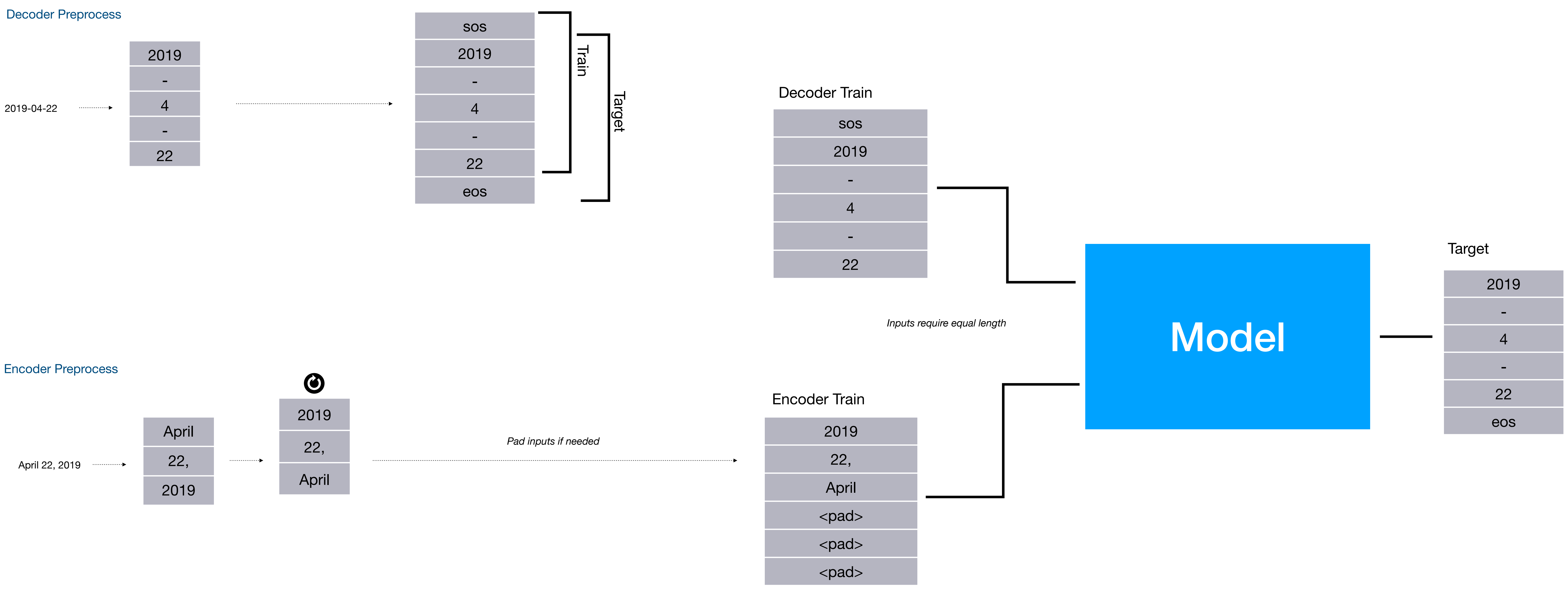


...
word
word
Text Vectorization
...
532
22
From Tensor Slices
...
batch = 1
532
batch = 1
22
GRU
GRU

MDY to ISO



Y_0 Y_1 Y_2 ...
Encoder
Encoder
Encoder
states
Y'_0 Y'_1 Y'_2 Y'_3 Y'_4 Y'_5 ...
Decoder
Decoder
Decoder
Decoder
Decoder
Decoder
X_0 id_2019
X_1 id_22
X_2 id_Aptril
batch
Embedding
"2019"
"22"
"April"
batch
Vectorize
Preprocess
"April 22 2019"
Target
id_2019
id_-
id_04
id_-
id_22
<eos>
X'_0 X'_1 X'_2 X'_3 X'_4 X'_5
Data
id_sos
id_2019
id_-
id_04
id_-
id_22
batch
Embedding
"sos"
"2019"
"-"
"04"
"-"
"22"
batch
Vectorize
Preprocess
"2019-04-22"


Decoder Preprocess
2019-04-22
2019
-
4
-
22
sos
2019
-
4
-
22
eos
Train
Target
Decoder Train
sos
2019
-
4
-
22
Inputs require equal length
Model
Target
2019
-
4
-
22
eos
Encoder Preprocess
April 22, 2019
April
22,
2019
2019
22,
April
Pad inputs if needed
Encoder Train
2019
22,
April
<pad>
<pad>
<pad>