Attention based Encoder Decoder Calculations

CMR@BRACU

Encoder

I go to school Start with embeddings of I, go, to ,school

Add positional encoding to determine the order of the words

Form Query, Key and Value matrix for encoder

Transform each embedding into K, Q and V embedding by multiplying with Query, Key and value matrix

Calculate self attention of I, between I and go, I and to, I and school and update the V embedding of I. Now calculate the self attention of go, between go and I, go and to, go and school and update the value embedding of go. Do the same for to and school

Add the V embedding of each token to their original embedding

Decoder

<EOS>

Form a new set of Query, Key and Value matrix fpr decoder and transform <EOS> embedding into Q embedding, K embedding and V embedding

Calculate self attention of EOS , update the V embedding of EOS and add the V embedding to the original embedding of EOS

Form a new set of Query, Key and Value matrix for cross attention between encoder and Decoder and transform the encoding to I, go, to, school and EOS

Calculate the attention between EOS and I, EOS and go, EOS and to and
EOS and school. Update the V embedding of EOS. Add this V embedding of EOS to its latest embedding

Feed latest embedding of EOS to a dense network to produce the embedding of আমি

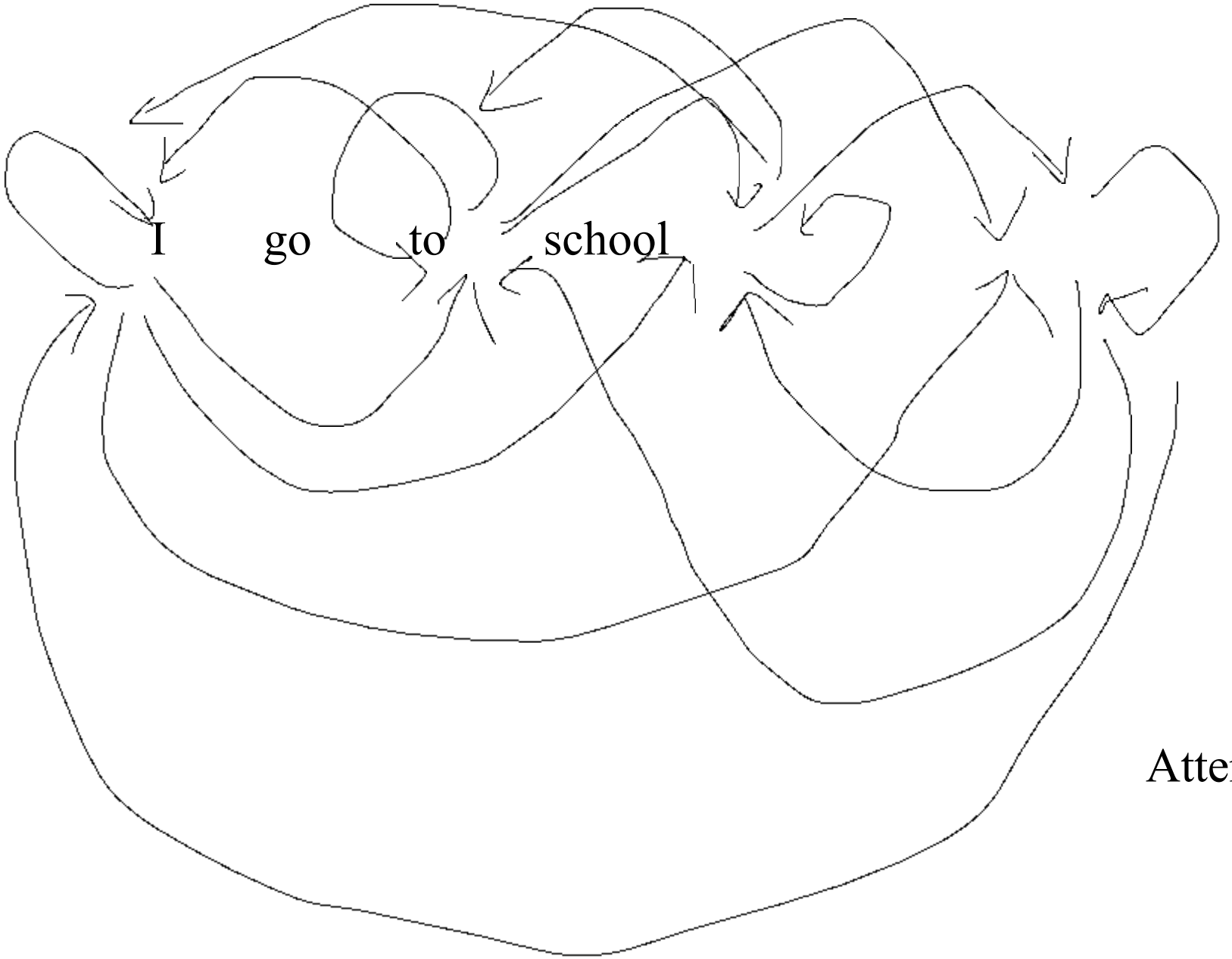
Embedding of আমি is converted to Q, K and V embedding using matrices for decoder

Calculate self attention of আমি and between আমি and <EOS>. Update the V embedding of আমি and add this V embedding with its original embedding

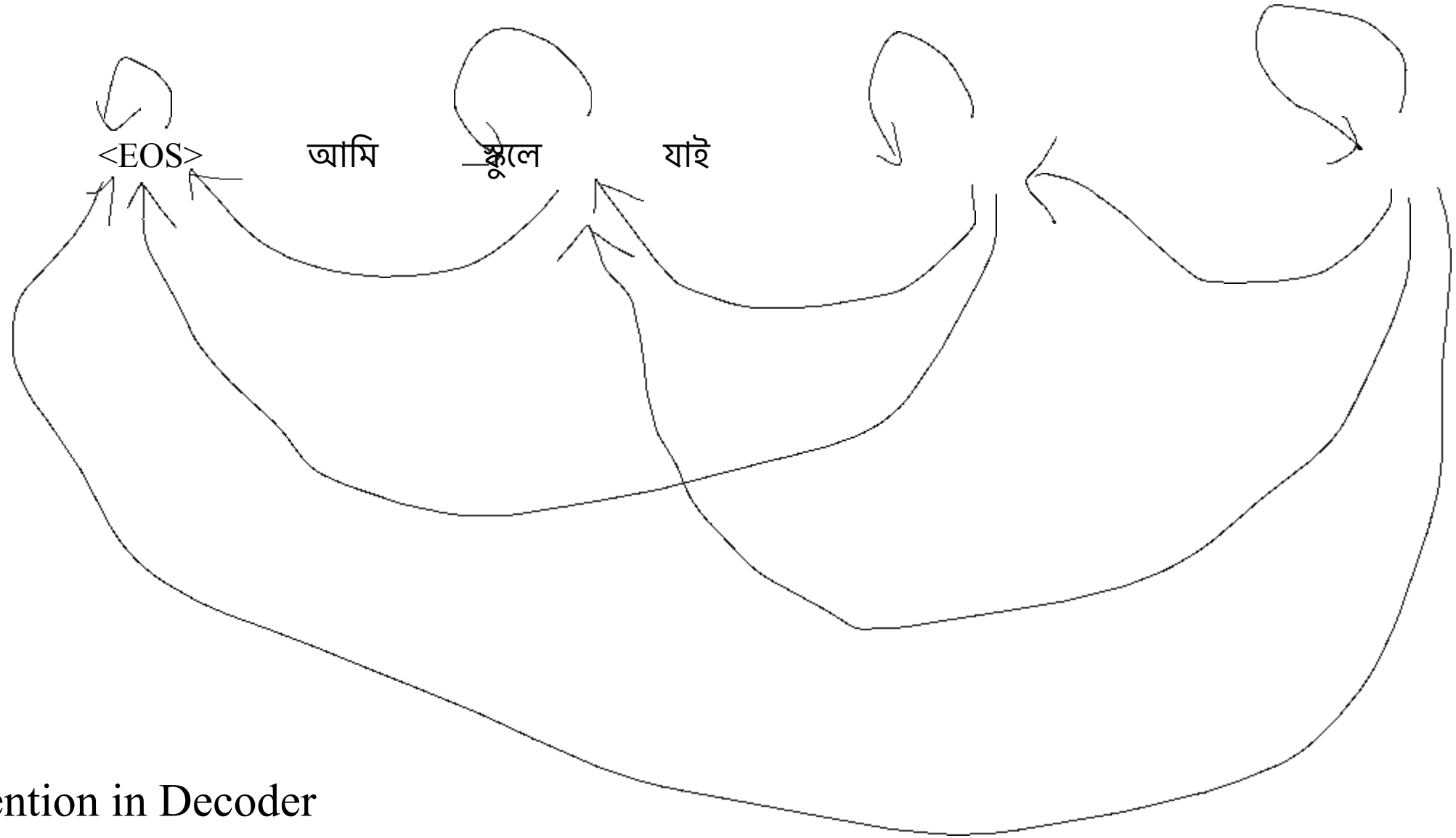
Now convert the latest embedding of আমি to Q, K and V embedding using the matrices of cross attention

Calculate the attention between আমি and I, আমি and go, আমি and to and আমি and School. Update the V encoding of আমি and add this with the latest encoding of আমি

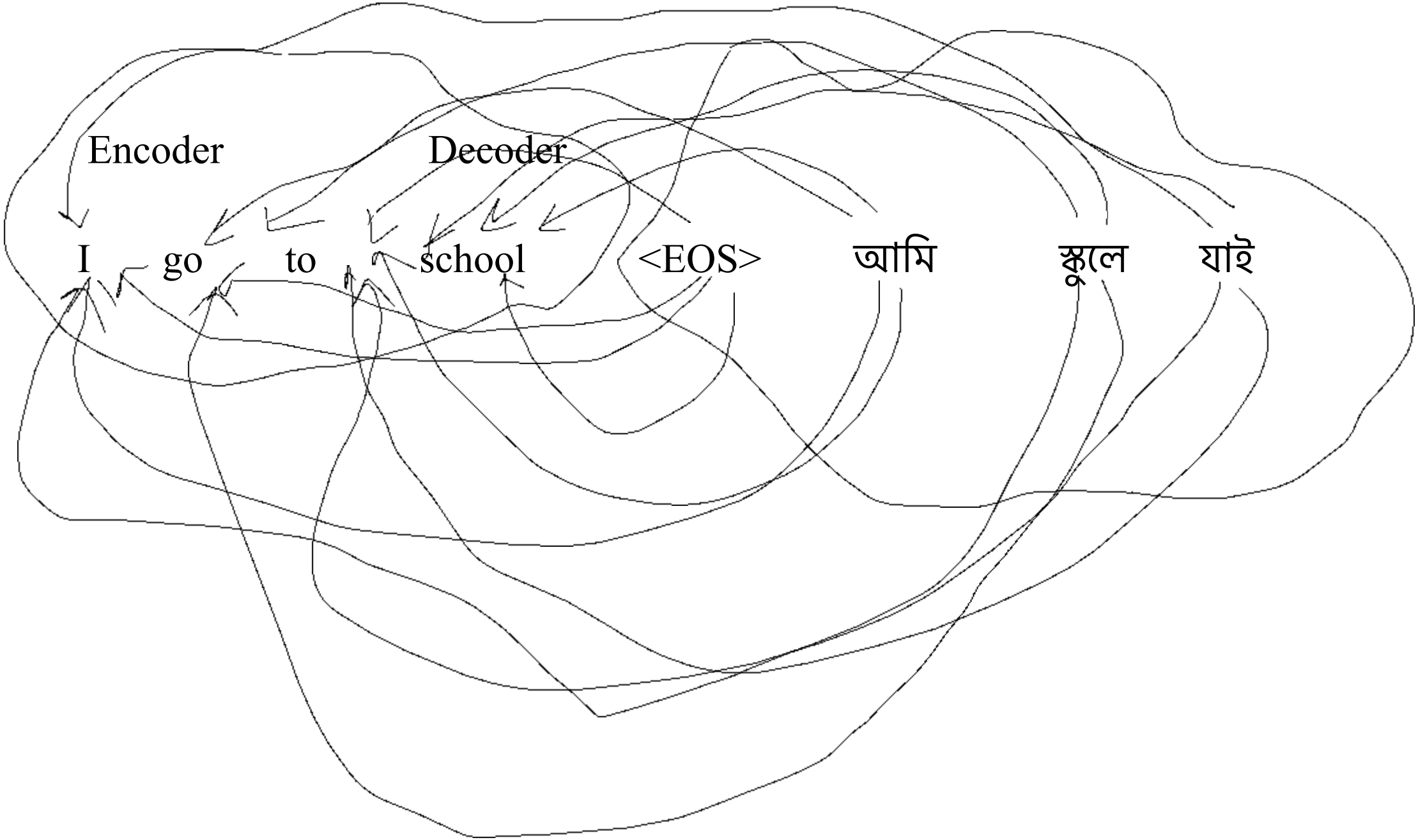
Feed the latest encoding of আমি to a dense neural network and produce the second word



Attention in Encoder



Attention in Decoder



Decoder encoder cross attention

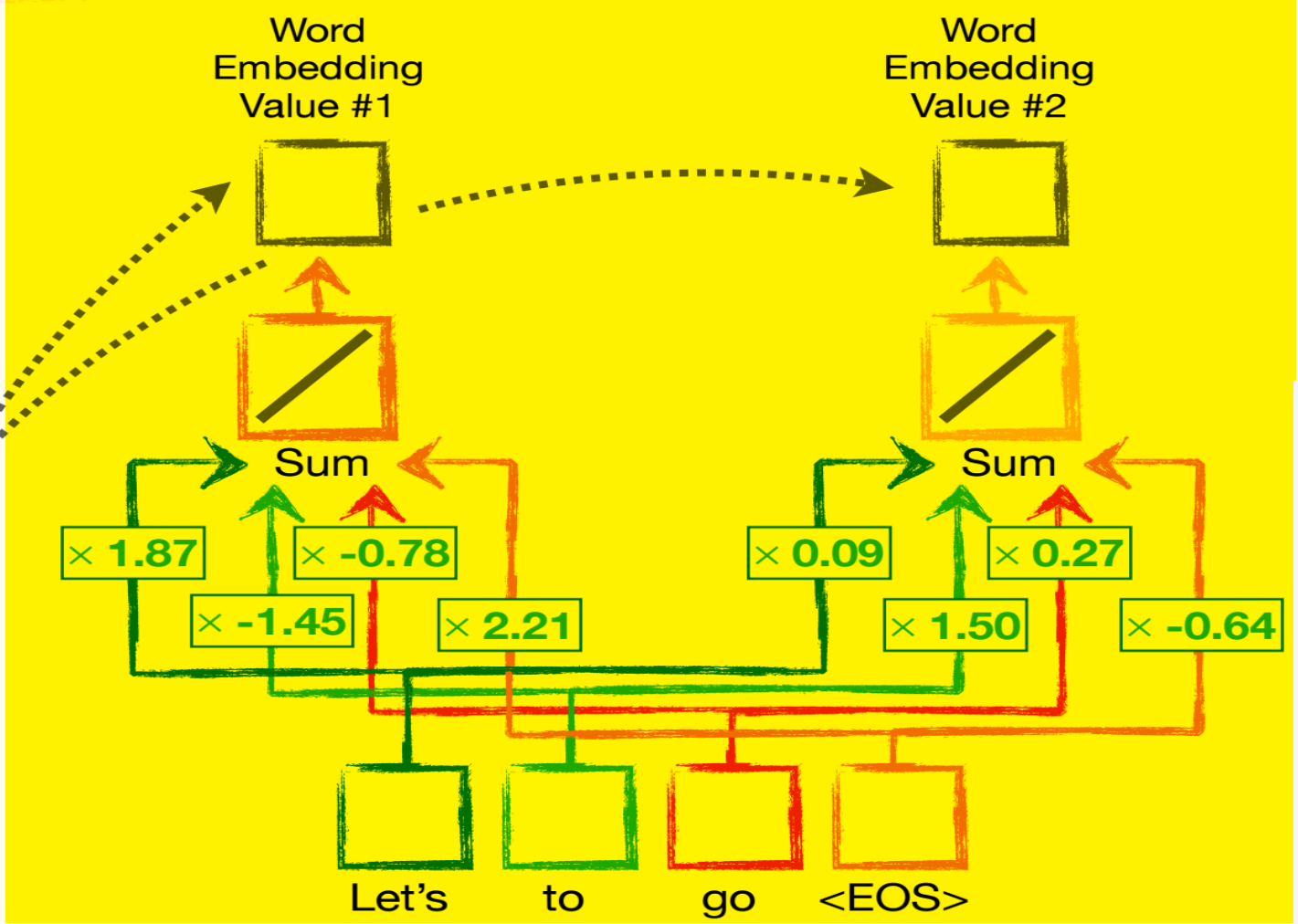
Embedding of স্কুলে is converted to Q, K and V embedding using matrices for decoder

Calculate self attention of স্কুলে and between স্কুলে and আমি and স্কুলে and <EOS>.
Update the V embedding of স্কুলে and add this V embedding with its original embedding

Now convert the latest embedding of স্কুলে to Q, K and V embedding using the matrices of cross attention

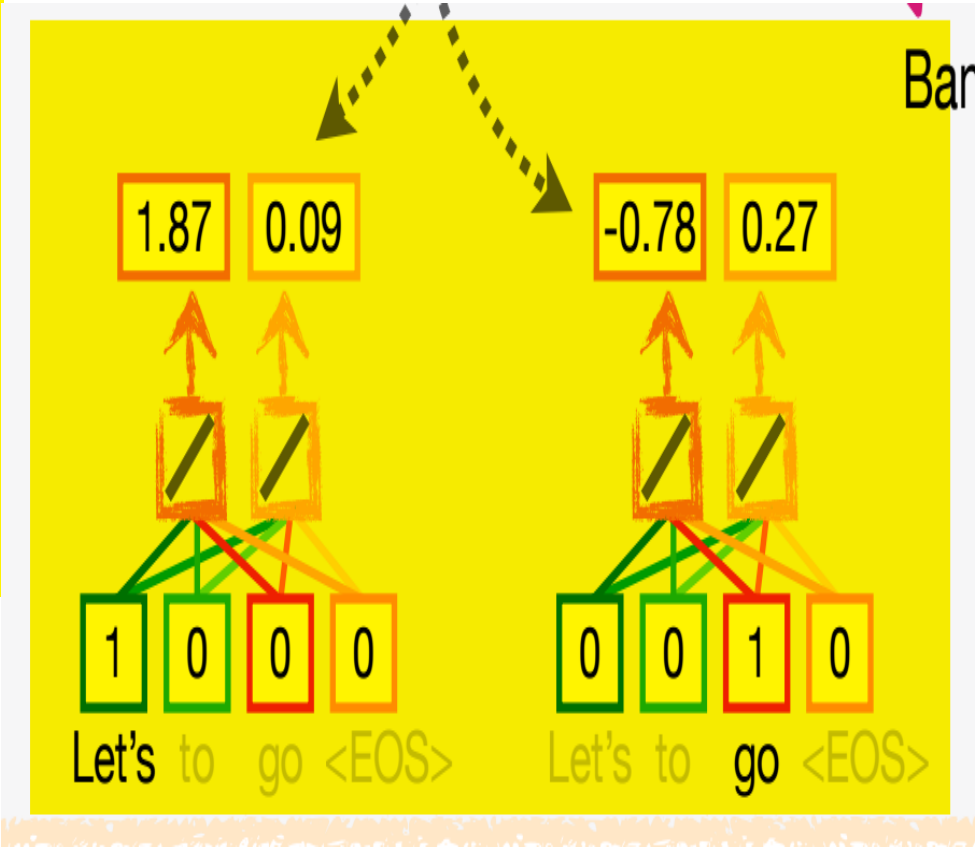
Calculate the attention between স্কুলে and I, স্কুলে and go, স্কুলে and to and স্কুলে and School. Update the V encoding of স্কুলে and add this with the latest encoding of স্কুলে

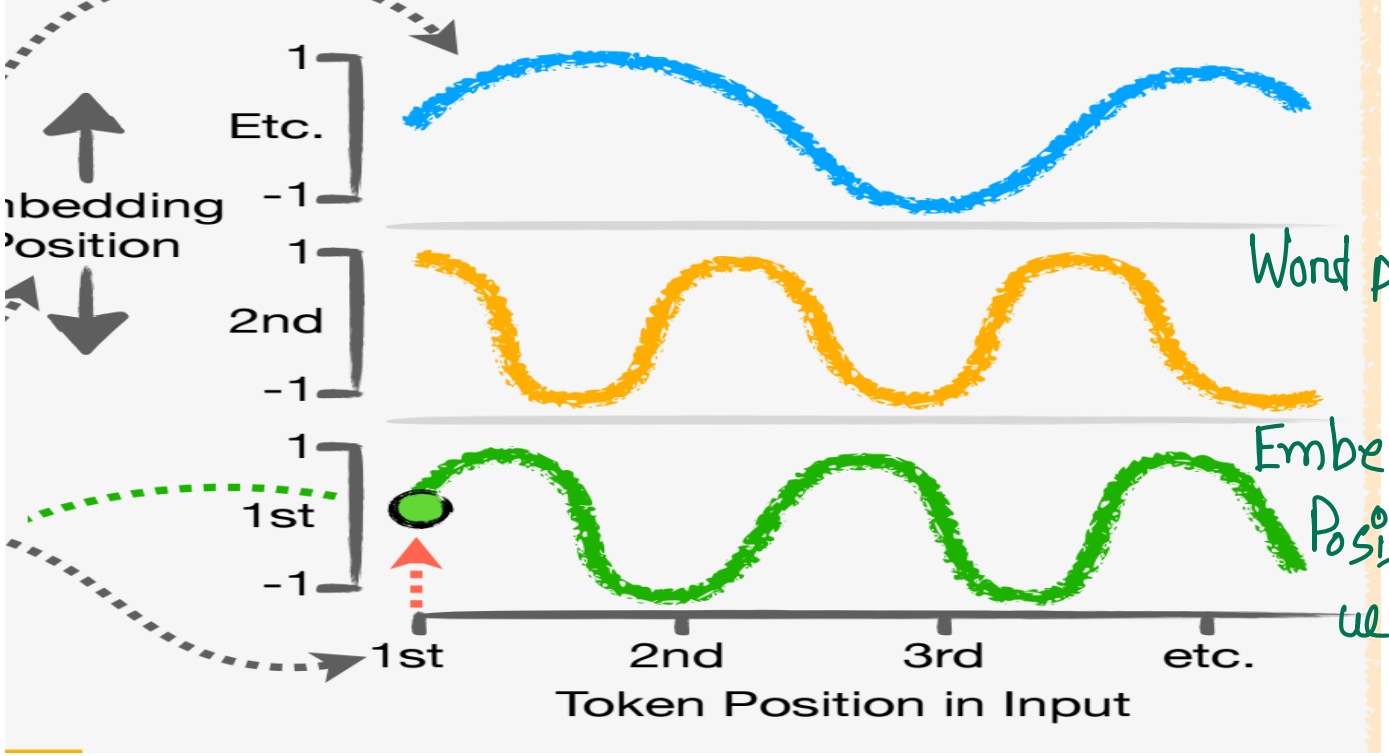
Feed the latest encoding of স্কুলে to a dense neural network and produce the third word Embedding for যাই



⇒ Word embedding

⇒ 2 dimensional word embedding





Formula

Word position

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

Embedding Dimension.

Embedding Position within a word.

$$PE(pos, \underset{+1}{2i}) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

Positional encoding (Order matters)

Encoding of Let's = .34, .56, .78, .90

PE → 2*0

PE → (0, 0)

$$= \sin \frac{0}{10000^{\frac{2*0}{4}}} = 0$$

$= .34 + 0$

PE (3, 3)?

PE (0,0) = sin(0) = 0, PE(0,1) = cos (0), PE(0,2) = sin(0), PE(0,3) = 1

PE(1,0) = sin(1), PE (1,1) = cos(1), PE (1,2) = sin(1/10000^(2/4)), PE(1,3) = cos(1/10000^(2/4)

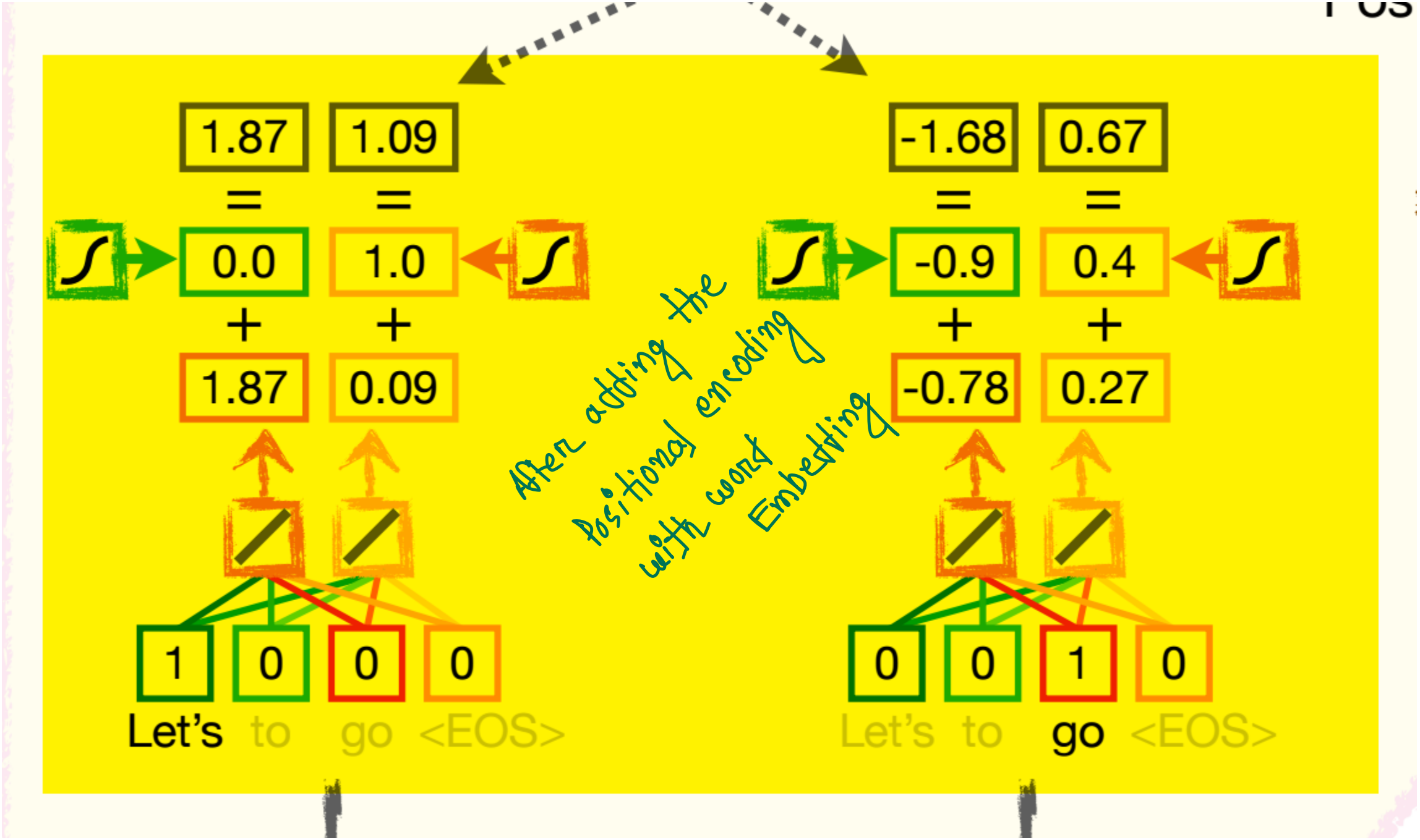
pos → 0

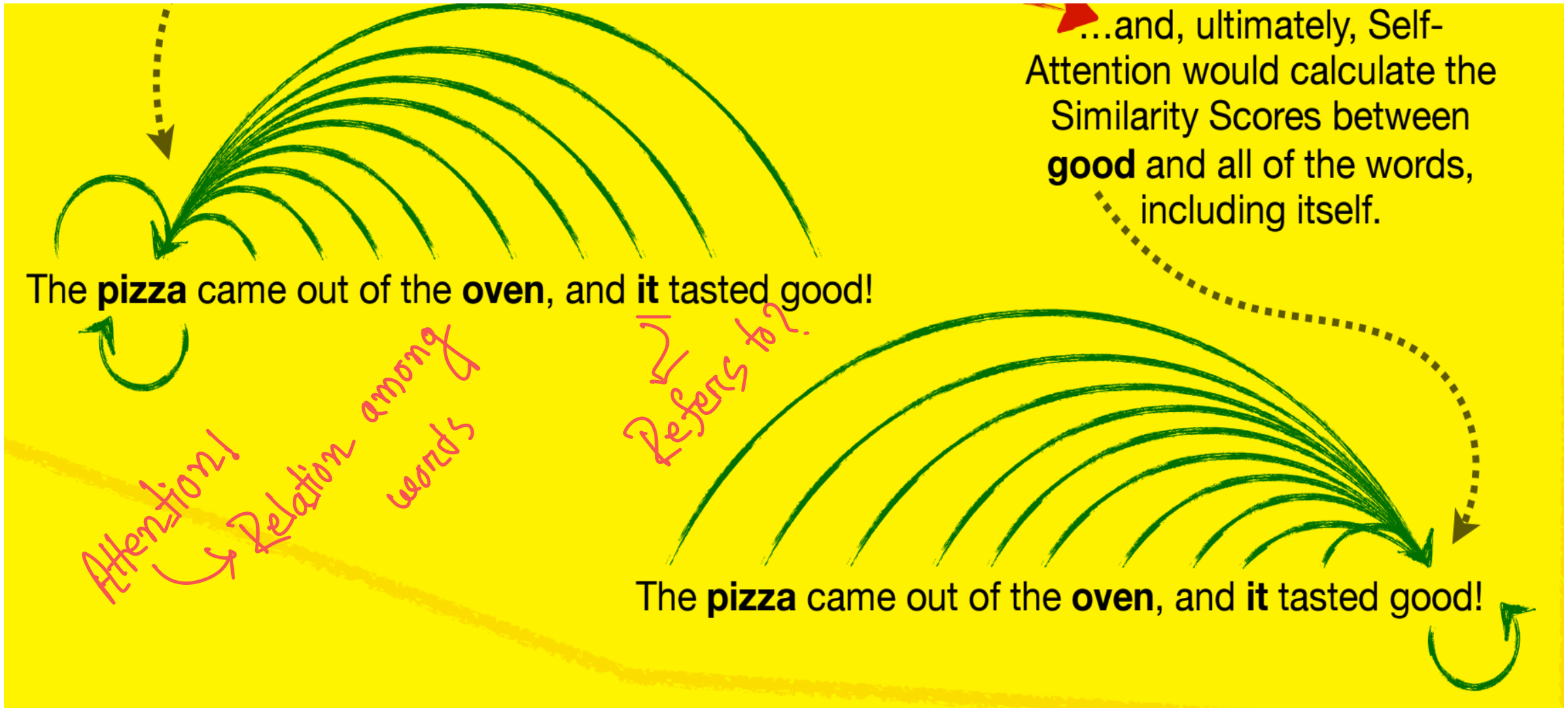
word1 word2 word3

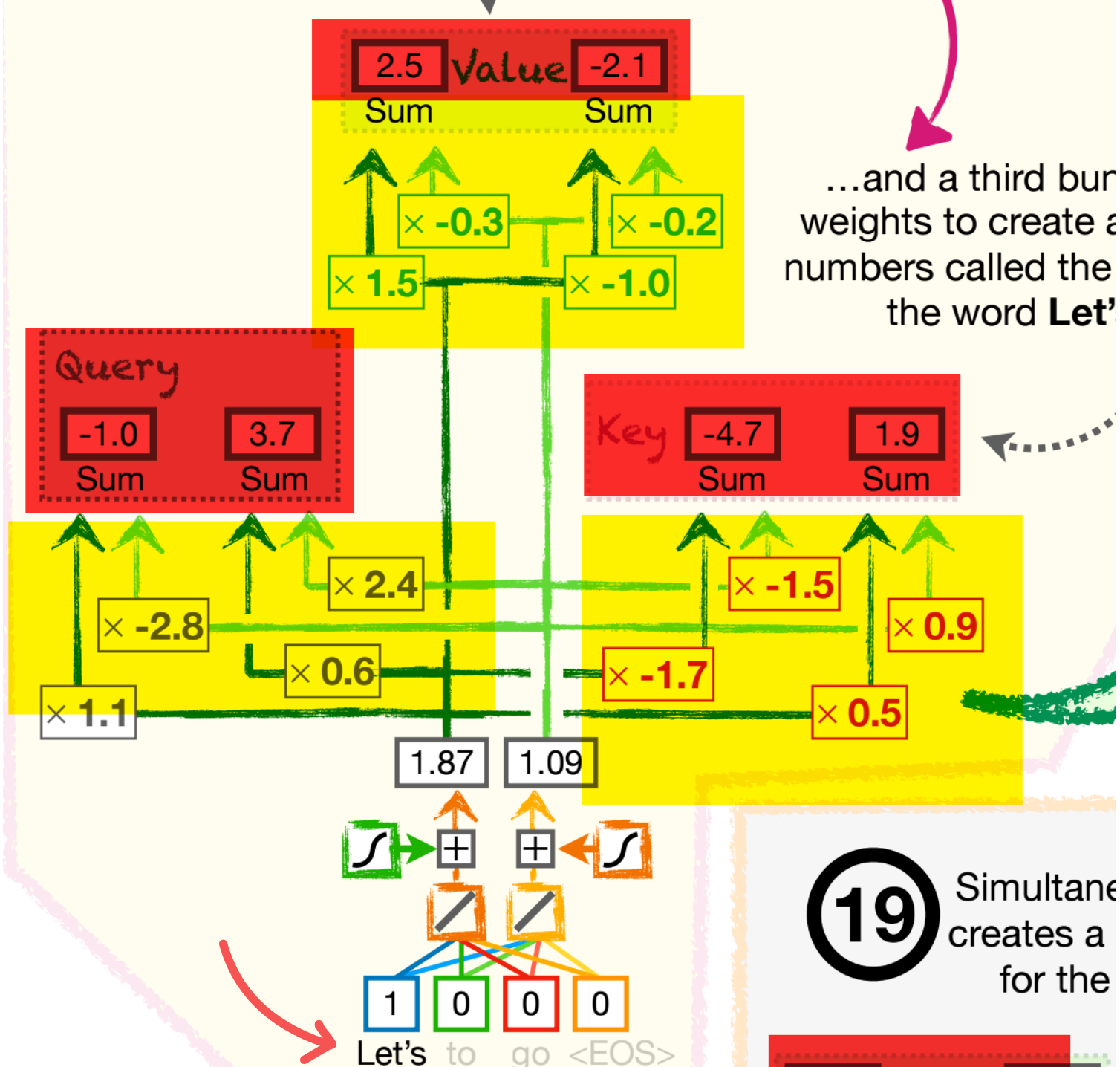
.34 .56 .78 .90 .56 .64 .88 .76 .34 .12 .39 .98

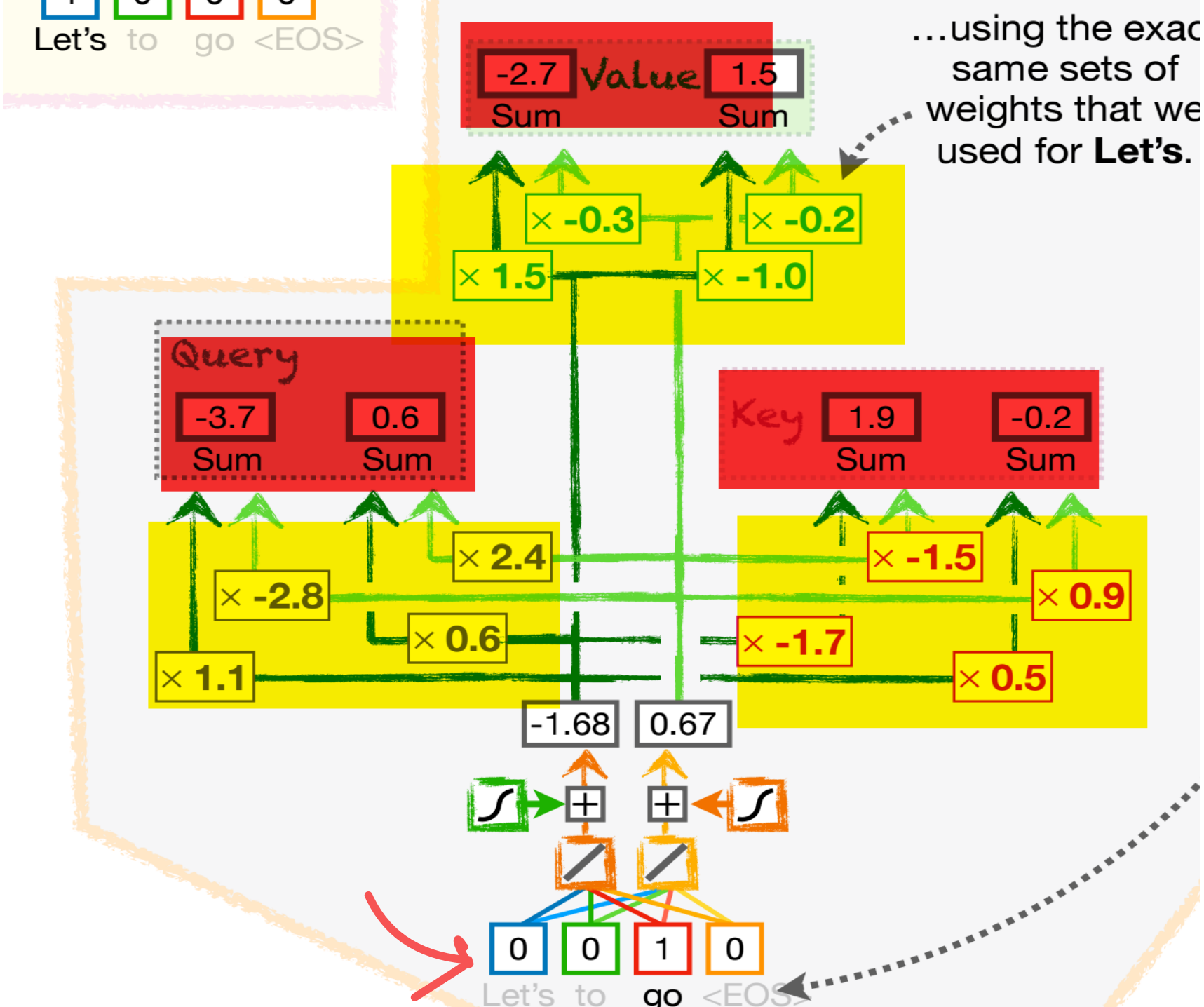
i=0 i=1

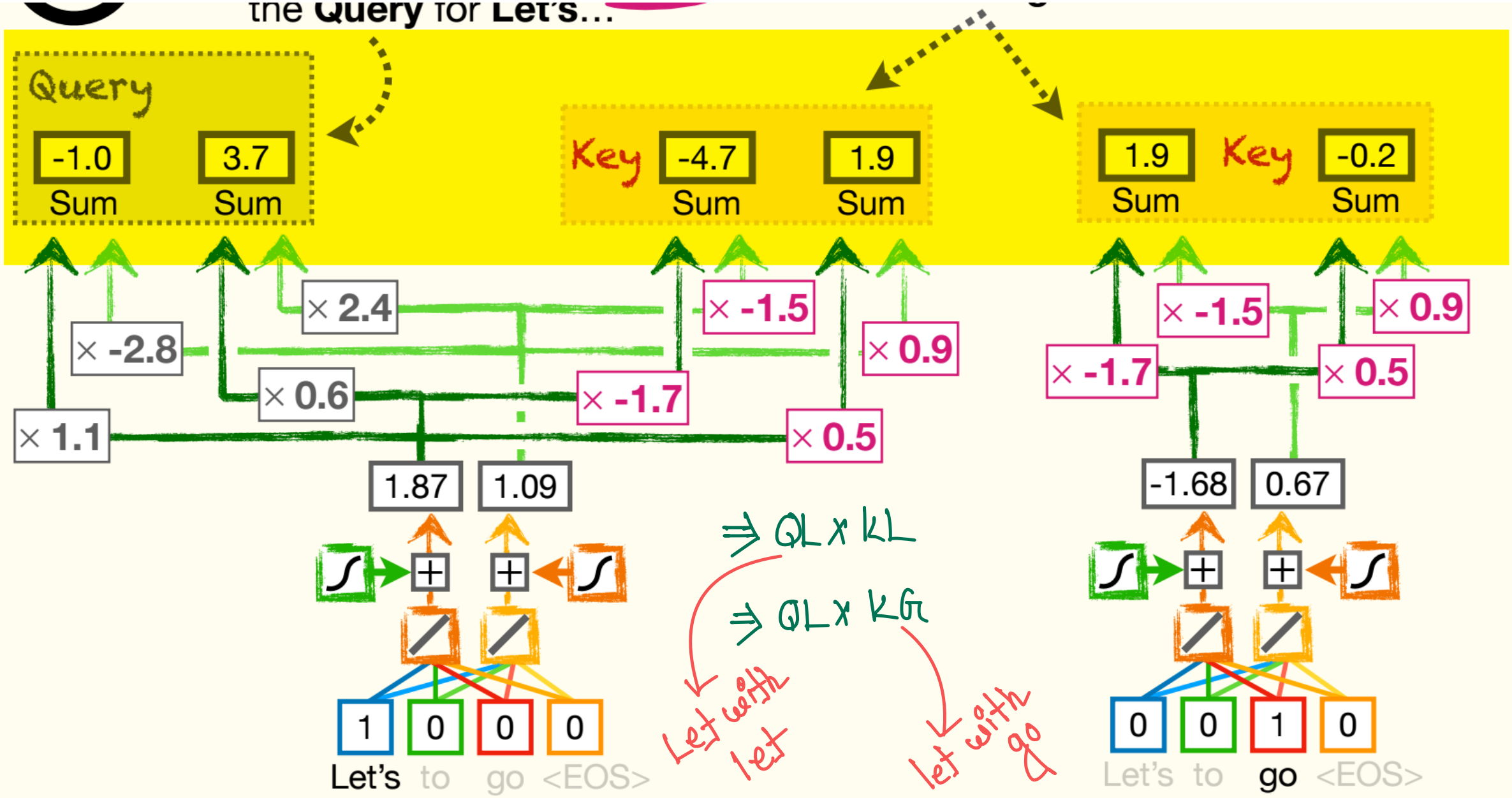
.34+0, .56+1, .78+0, .90+1 .56+0.02, .64+1, .88+ 0, .76+1

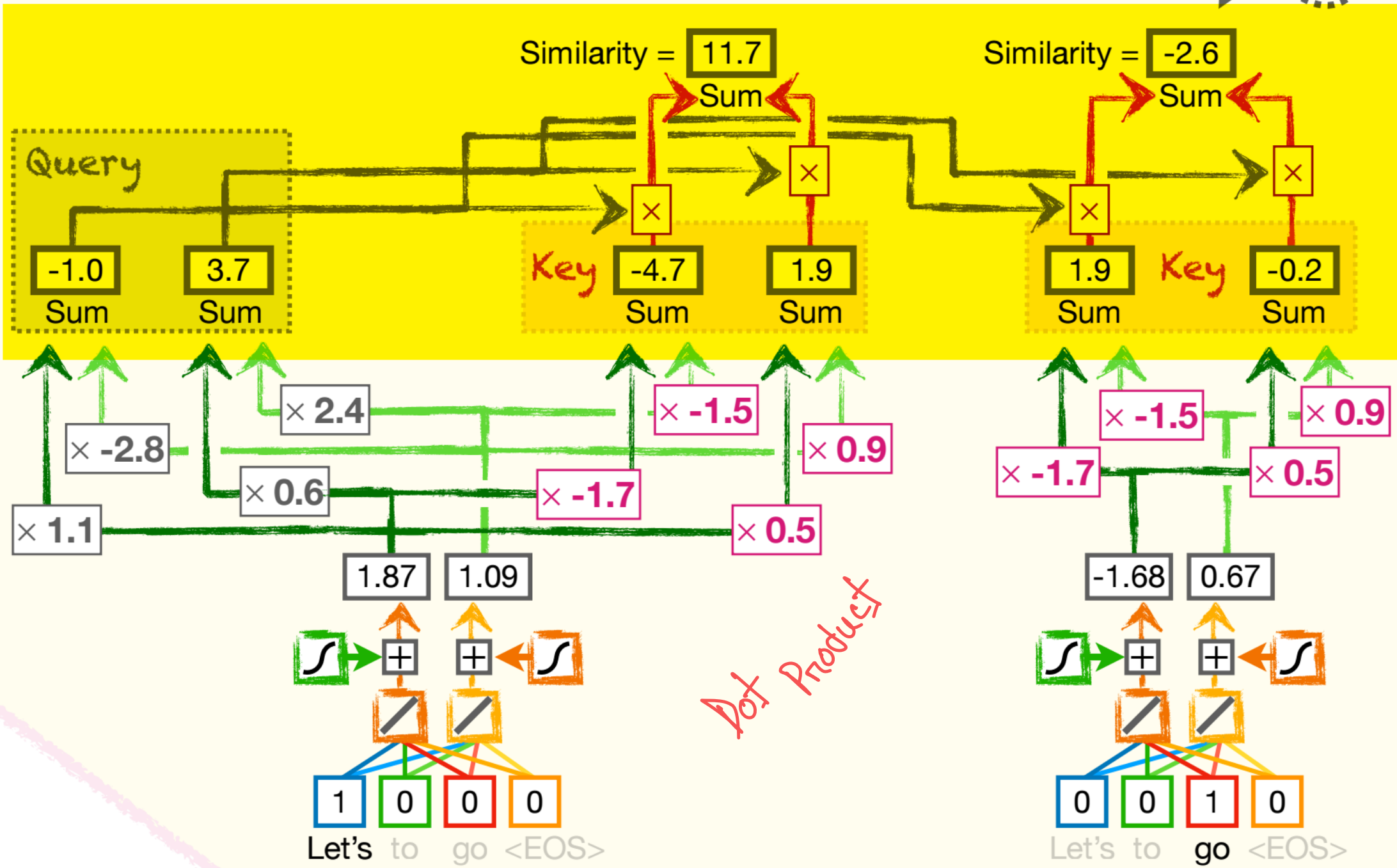

1.87
1.09
=
=
0.0
1.0
+
+
1.87
0.09
-1.68
0.67
=
=
-0.9
0.4
+
+
-0.78
0.27
After adding the positional encoding with word Embedding
1
0
0
0
Let's
to
go
<EOS>
0
0
1
0
Let's
to
go
<EOS>

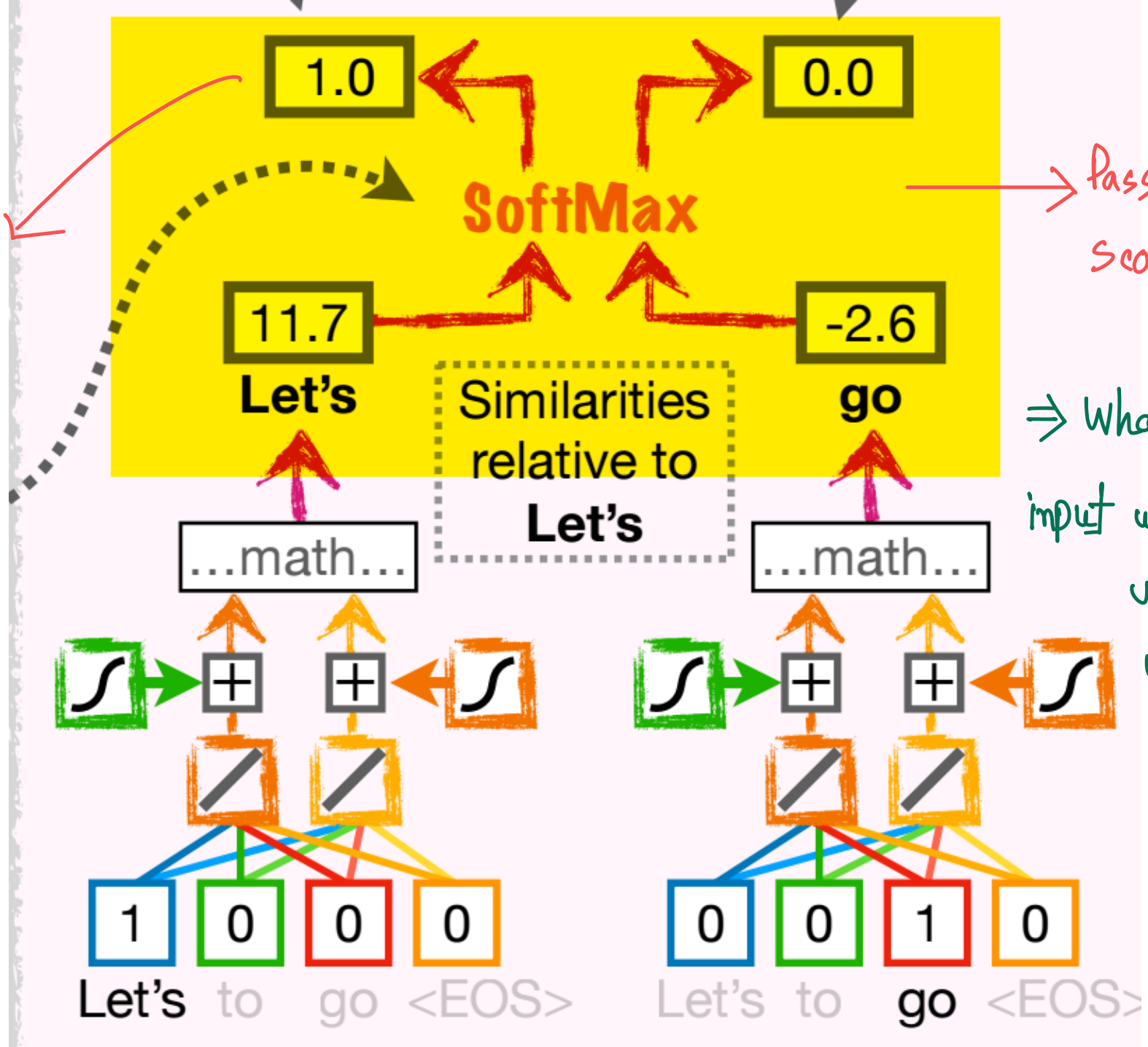

...and, ultimately, Self-Attention would calculate the Similarity Scores between **good** and all of the words, including itself.
The **pizza** came out of the **oven**, and **it** tasted good!
Attention! Relation among words
it Refers to?
The **pizza** came out of the **oven**, and **it** tasted good!


2.5
Value
-2.1
Sum
Sum
× -0.3
× -0.2
× 1.5
× -1.0
Query
-1.0
3.7
Sum
Sum
Key
-4.7
1.9
Sum
Sum
× 2.4
× -1.5
× -2.8
× 0.9
× 0.6
× -1.7
× 1.1
× 0.5
1.87
1.09
1
0
0
0
Let's
to
go
<EOS>
…and a third bun
weights to create a
numbers called the
the word Let'
19
Simultane
creates a
for the


Let's to go <EOS>
-2.7 Value 1.5
Sum Sum
...using the exac
same sets of
weights that we
used for Let's.
× -0.3
× -0.2
× 1.5
× -1.0
Query
-3.7 0.6
Sum Sum
Key
1.9 -0.2
Sum Sum
× 2.4
× -1.5
× -2.8
× 0.9
× 0.6
× -1.7
× 1.1
× 0.5
-1.68 0.67
0 0 1 0
Let's to go <EOS>


the Query for Let's...
Query
-1.0
Sum
3.7
Sum
Key
-4.7
Sum
1.9
Sum
1.9
Sum
Key
-0.2
Sum
× 1.1
× -2.8
× 2.4
× 0.6
× -1.7
× 0.5
× -1.5
× 0.9
× -1.7
× -1.5
× 0.5
× 0.9
1.87
1.09
-1.68
0.67
⇒ QL x kL
⇒ QL x kG
Let with let
let with go
1
0
0
0
0
0
1
0
Let's
to
go
<EOS>
Let's
to
go
<EOS>

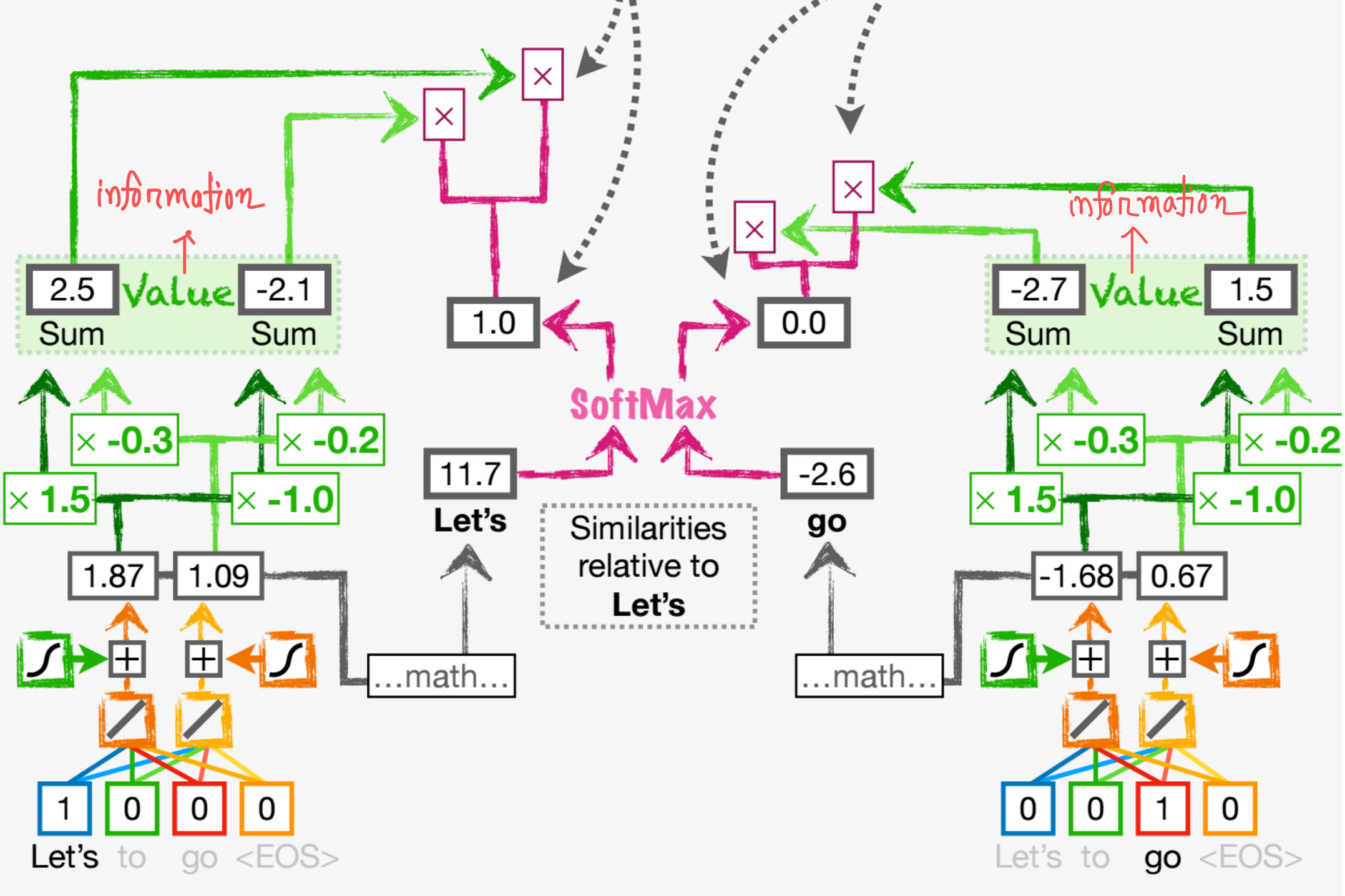

Similarity = 11.7
Sum
Similarity = -2.6
Sum
Query
-1.0
Sum
3.7
Sum
Key
-4.7
Sum
1.9
Sum
1.9
Sum
Key
-0.2
Sum
× 1.1
× -2.8
× 2.4
× 0.6
× -1.7
× -1.5
× 0.9
× 0.5
× -1.5
× 0.9
× -1.7
× 0.5
1.87
1.09
-1.68
0.67
Dot Product
1 0 0 0
Let's to go <EOS>
0 0 1 0
Let's to go <EOS>



$$\frac{e^{11.7}}{e^{11.7}+e^{-2.6}} \approx 0.99 \approx 1.$$

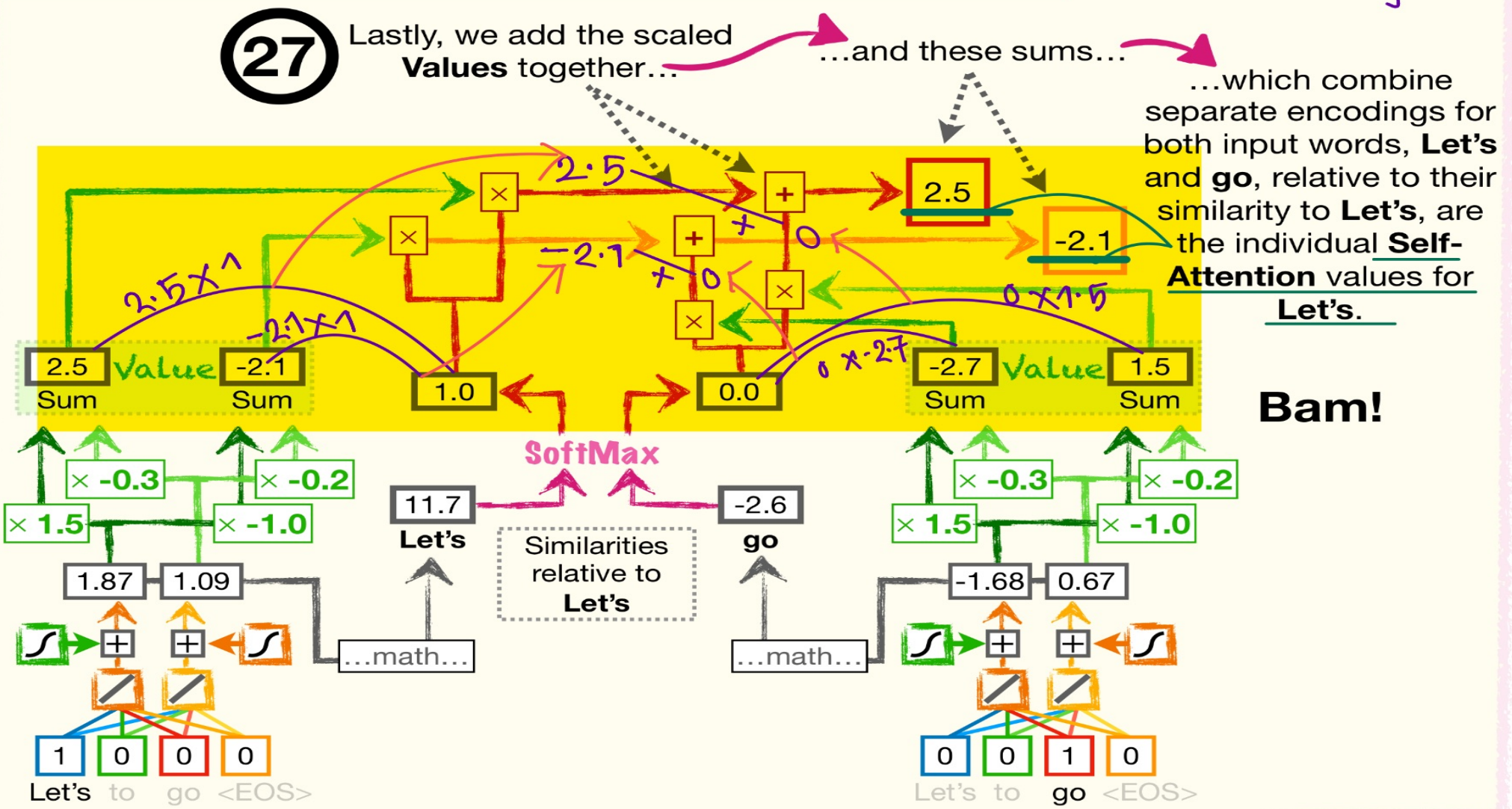
→ Pass the similarity score into SoftMax function.

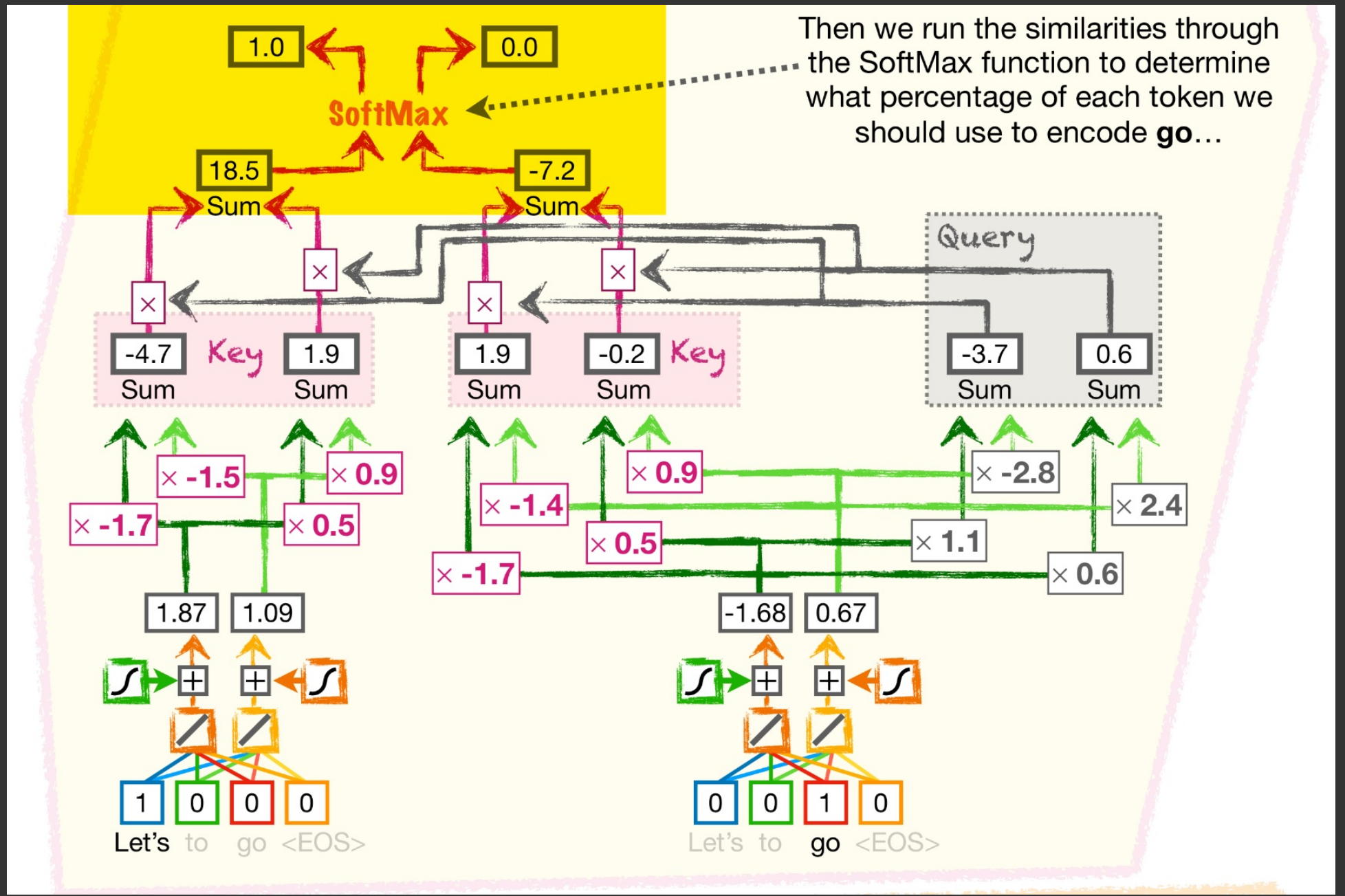
⇒ What 1. of each input word should we use to encode the word Let.

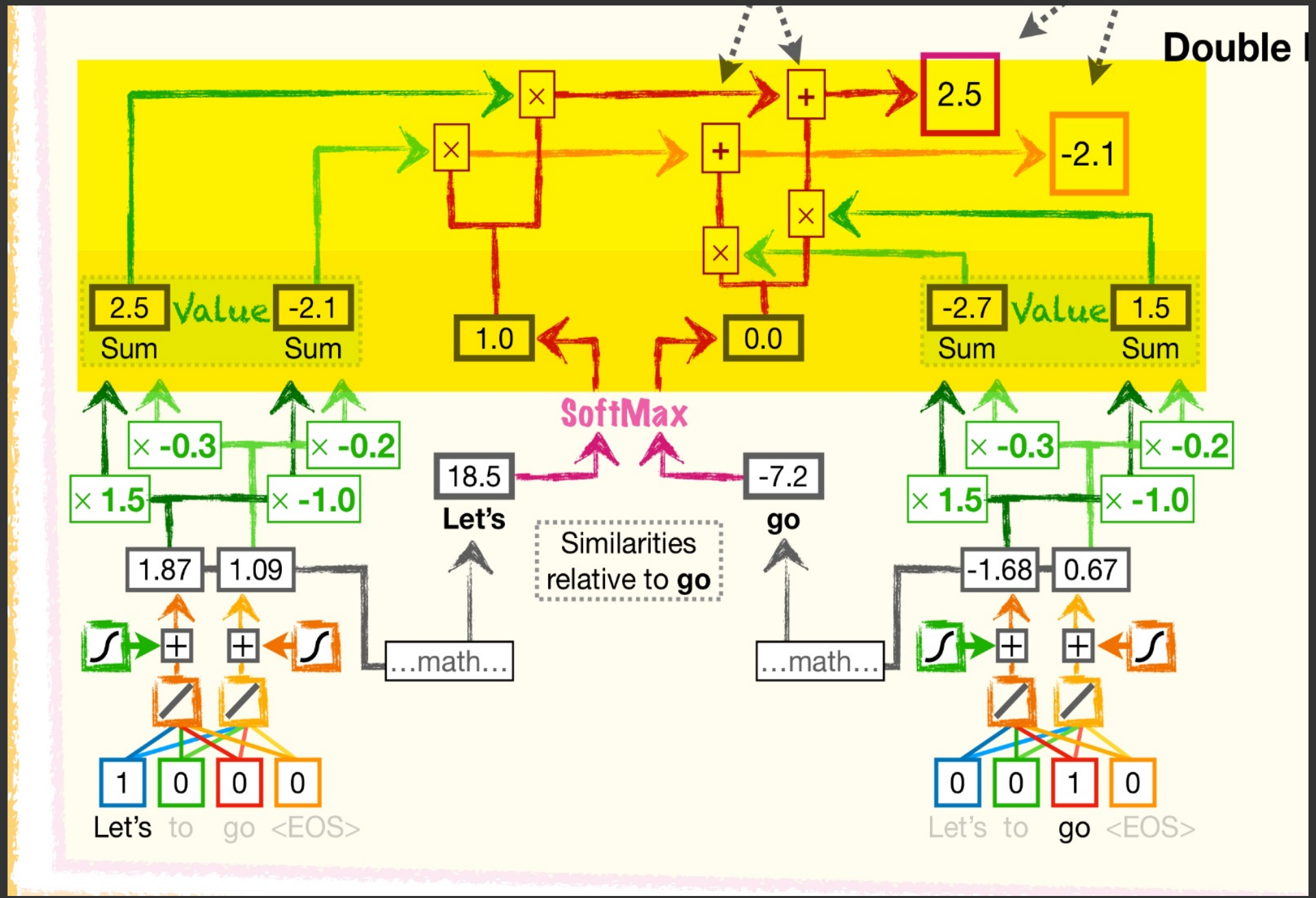

information
2.5
Value
-2.1
Sum
Sum
1.0
0.0
SoftMax
information
-2.7
Value
1.5
Sum
Sum
× -0.3
× -0.2
× 1.5
× -1.0
1.87
1.09
11.7
Let's
Similarities relative to Let's
-2.6
go
× -0.3
× -0.2
× 1.5
× -1.0
-1.68
0.67
...math...
...math...
1
0
0
0
Let's
to
go
<EOS>
0
0
1
0
Let's
to
go
<EOS>

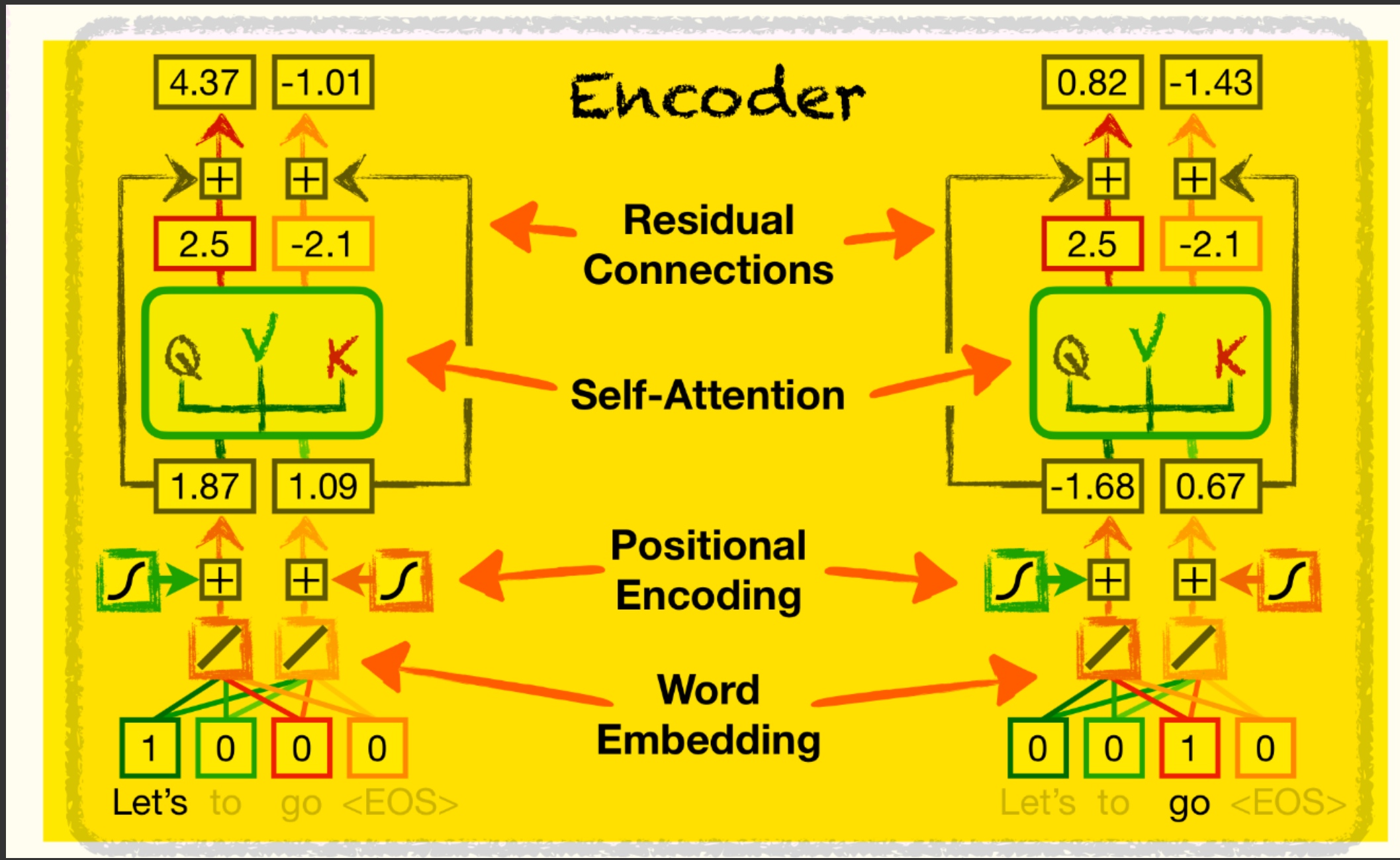
Transformers: Encoding Details

inner + inner / self
outer + outer / attention

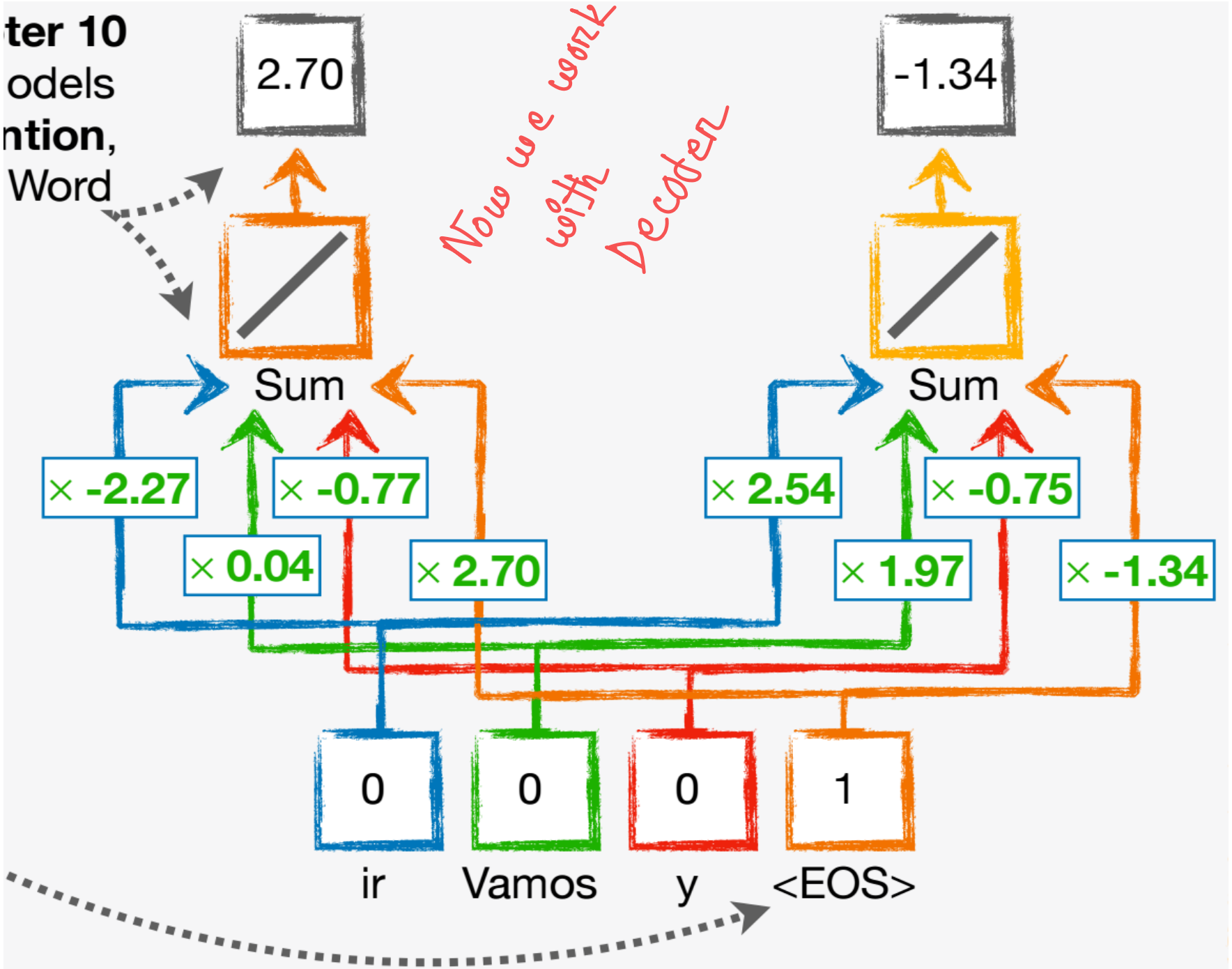



1.0
0.0
SoftMax
18.5
-7.2
Sum
Sum
Then we run the similarities through the SoftMax function to determine what percentage of each token we should use to encode **go**…
Query
-4.7
Key
1.9
1.9
-0.2
Key
-3.7
0.6
Sum
Sum
Sum
Sum
Sum
Sum
× -1.5
× 0.9
× -1.7
× 0.5
× 0.9
× -2.8
× -1.4
× 2.4
× 0.5
× 1.1
× -1.7
× 0.6
1.87
1.09
-1.68
0.67
1
0
0
0
0
0
1
0
Let's
to
go
<EOS>
Let's
to
go
<EOS>


Double
2.5
-2.1
1.0
0.0
2.5
Value
-2.1
Sum
Sum
-2.7
Value
1.5
Sum
Sum
SoftMax
× -0.3
× -0.2
× 1.5
× -1.0
18.5
Let's
-7.2
go
Similarities relative to go
1.87
1.09
-1.68
0.67
…math…
…math…
1
0
0
0
0
0
1
0
Let's
to
go
<EOS>
Let's
to
go
<EOS>

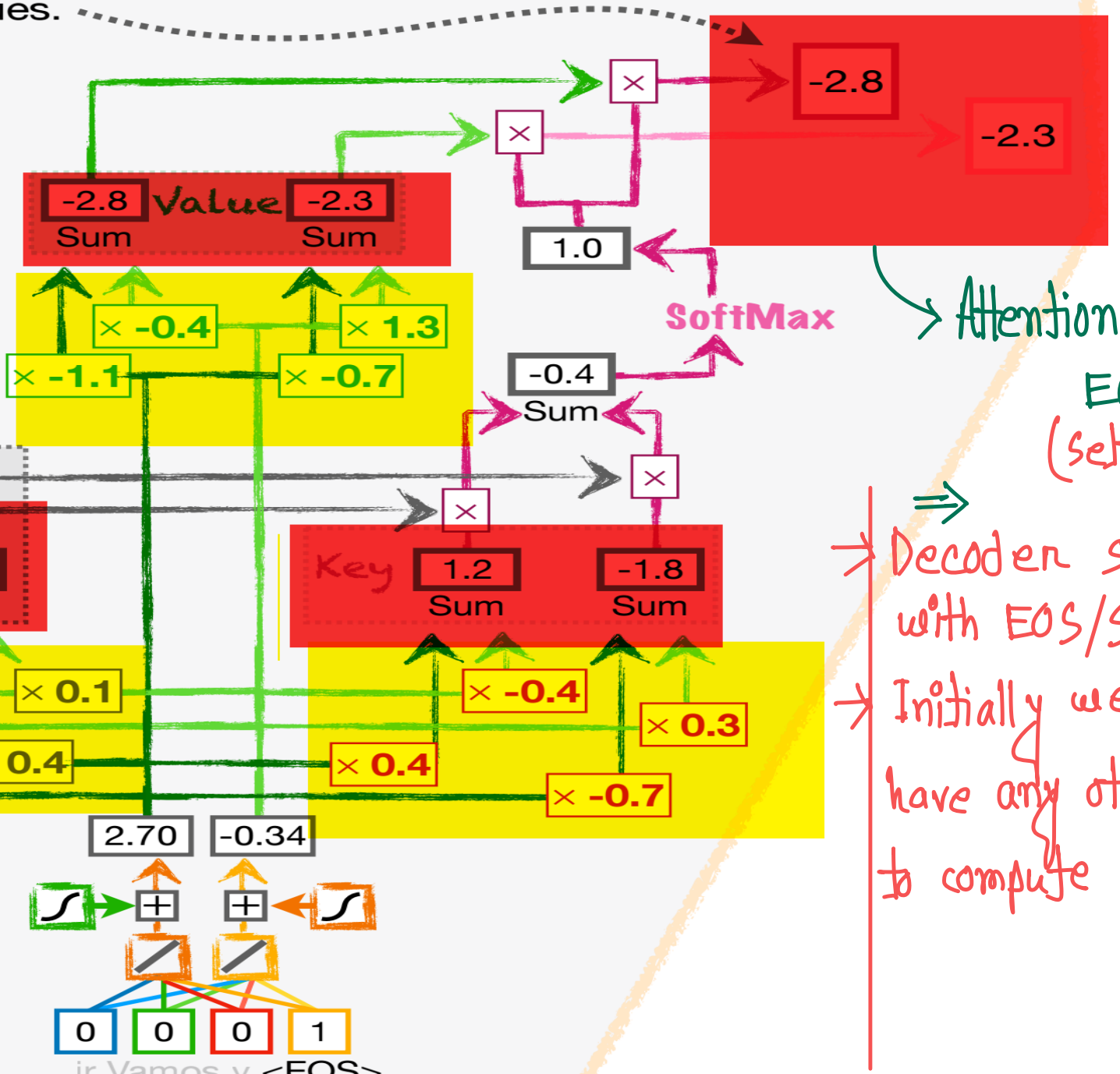

Encoder
4.37
-1.01
0.82
-1.43
Residual Connections
2.5
-2.1
2.5
-2.1
Q
V
K
Q
V
K
Self-Attention
1.87
1.09
-1.68
0.67
Positional Encoding
Word Embedding
1
0
0
0
0
0
1
0
Let's
to
go
<EOS>
Let's
to
go
<EOS>

ter 10
odels
ntion,
Word



Self-Attention values.

only one token in the
h is all we have right
a Self-Attention value
silly, since we do a lot
ust to use **100%** of its
rs. However, as we
okens, calculating Self-
t more interesting, and
don't need to code a
for the first token.



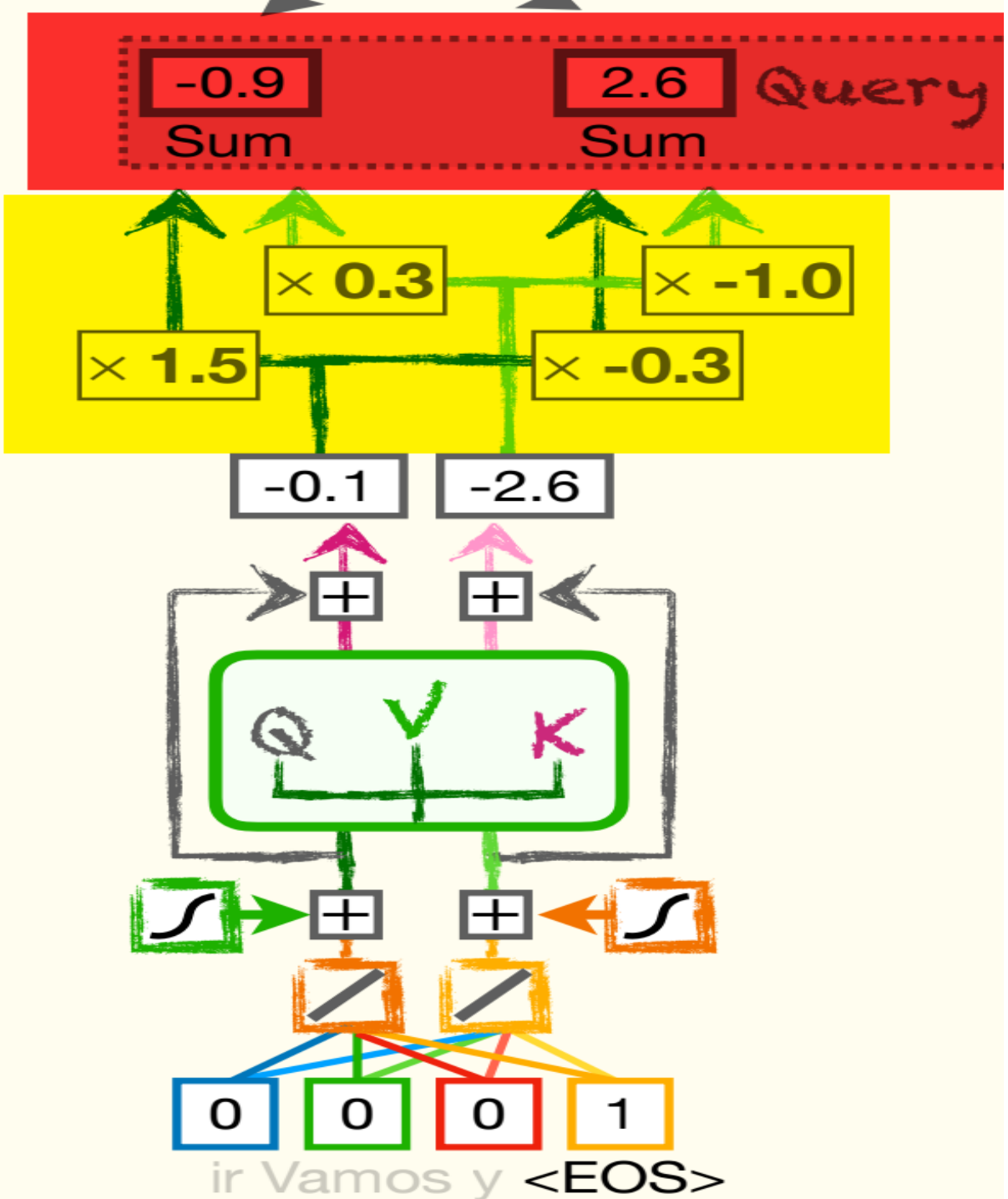
Attention for EOS. (self)

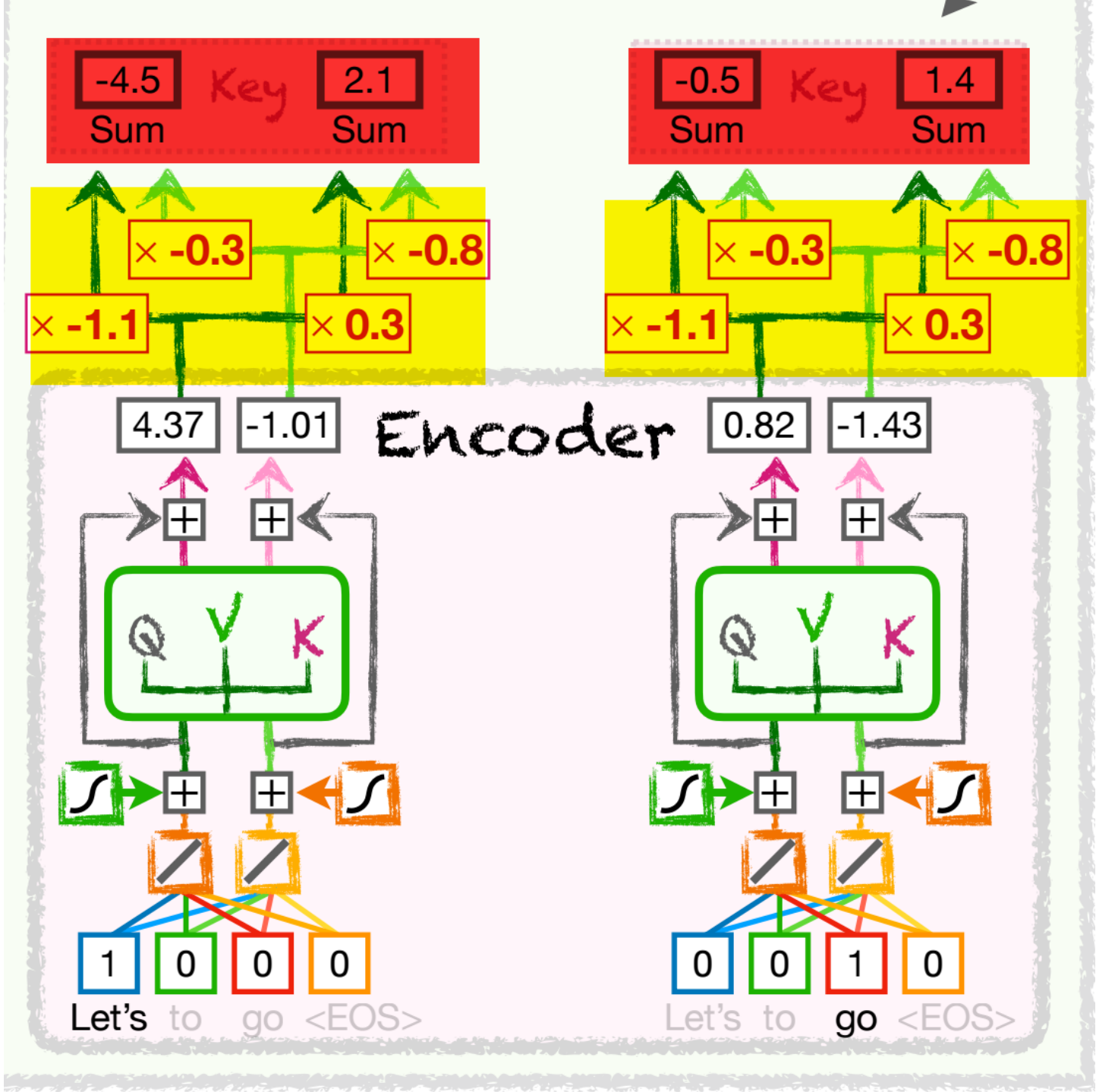
⇒

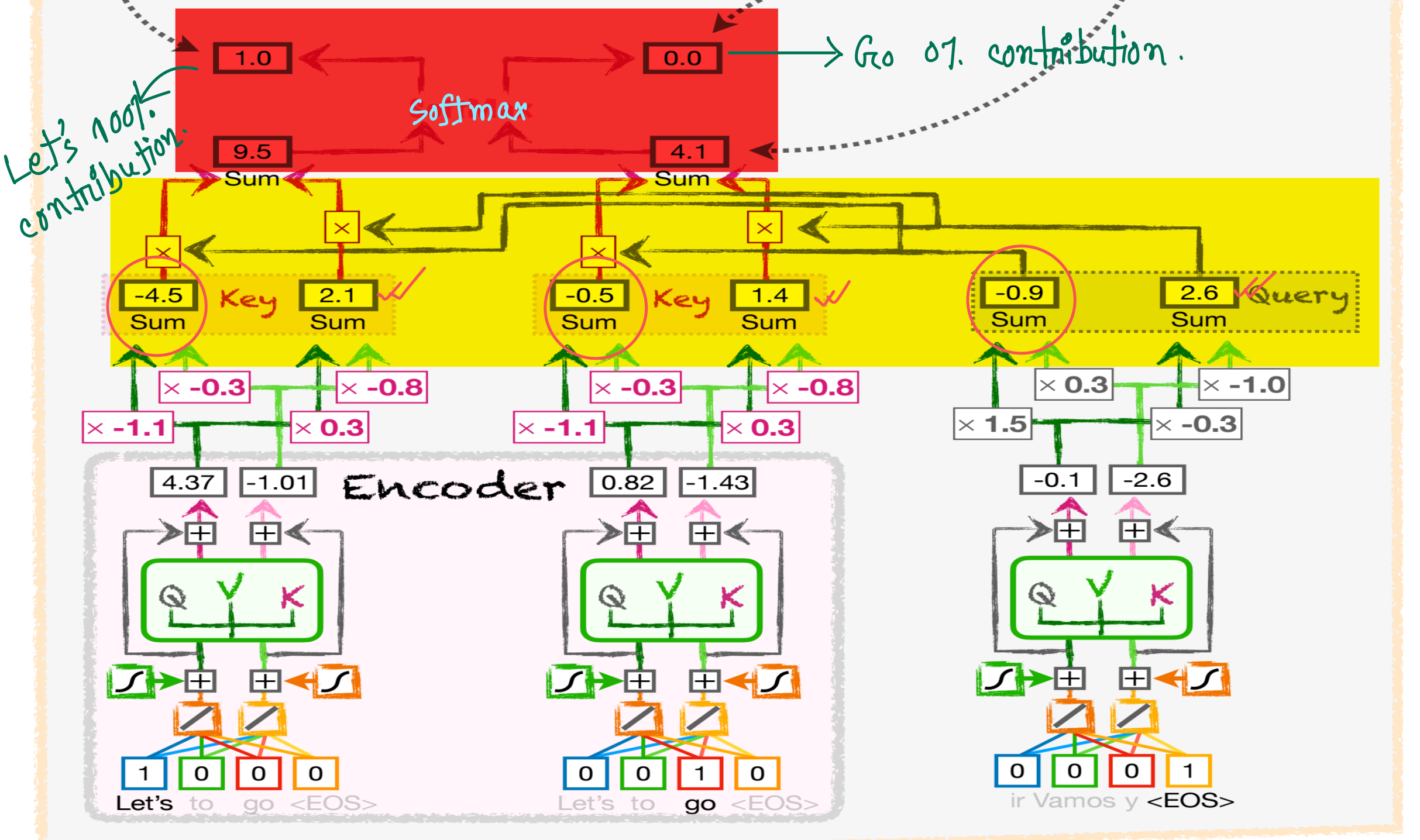
- Decoder starts with EOS/SOS.
- Initially we do not have any other words to compute attention.

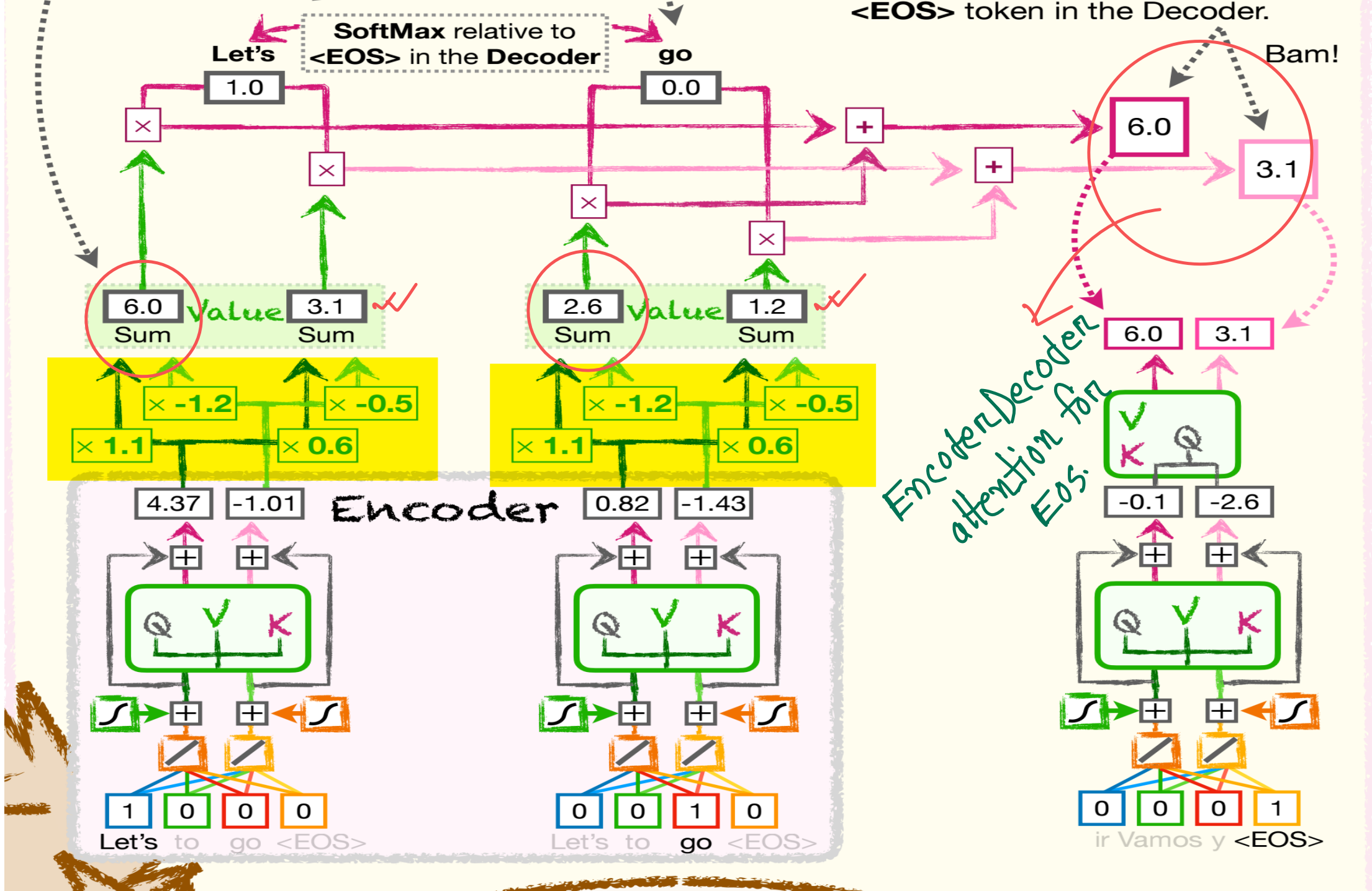
s of weights that we
te the **Query**, **Key**,
bers in the Decoder
from the sets of
se in the Encoder.

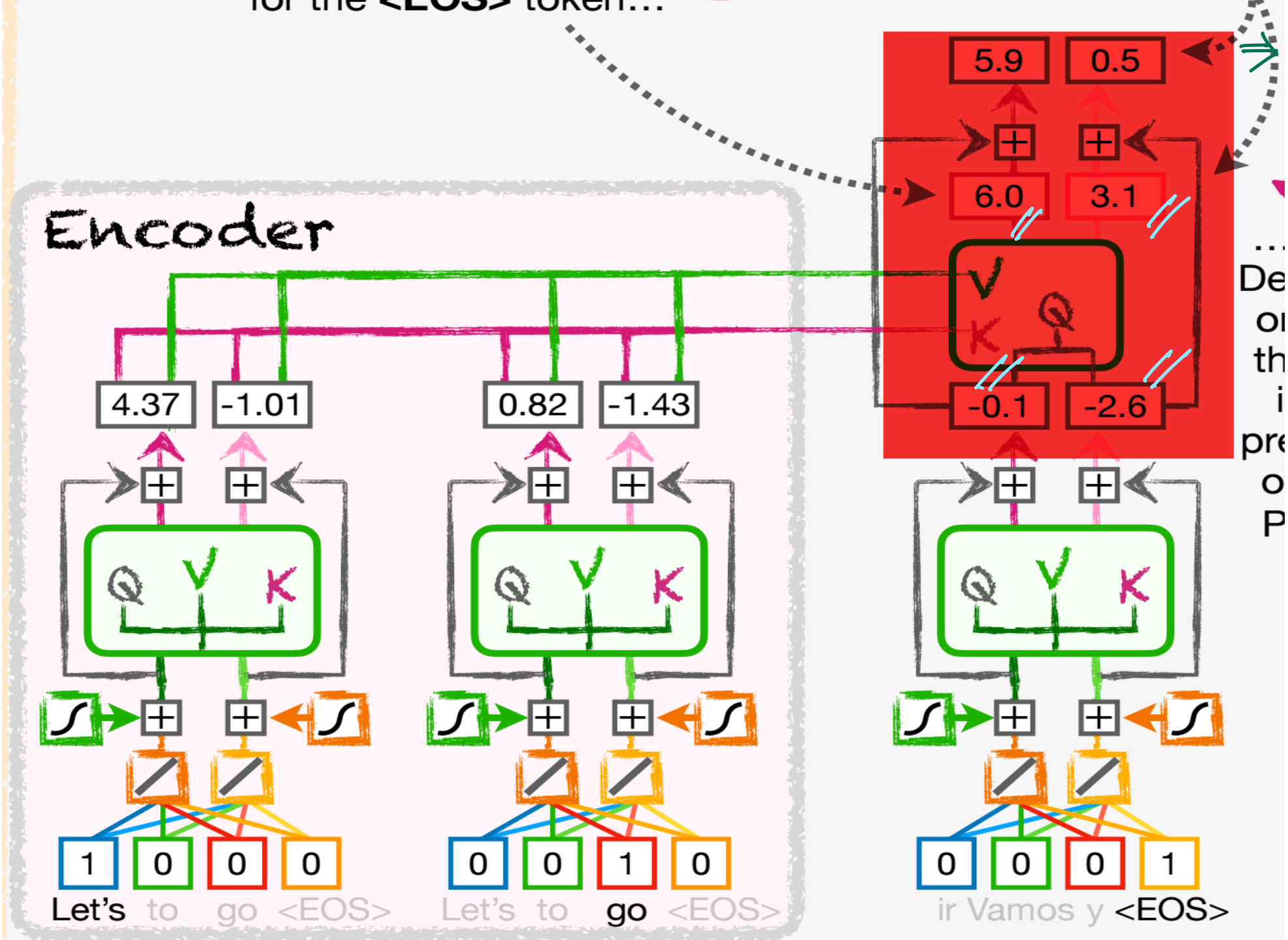
Now we will compute attention: EOS — Let's — Go

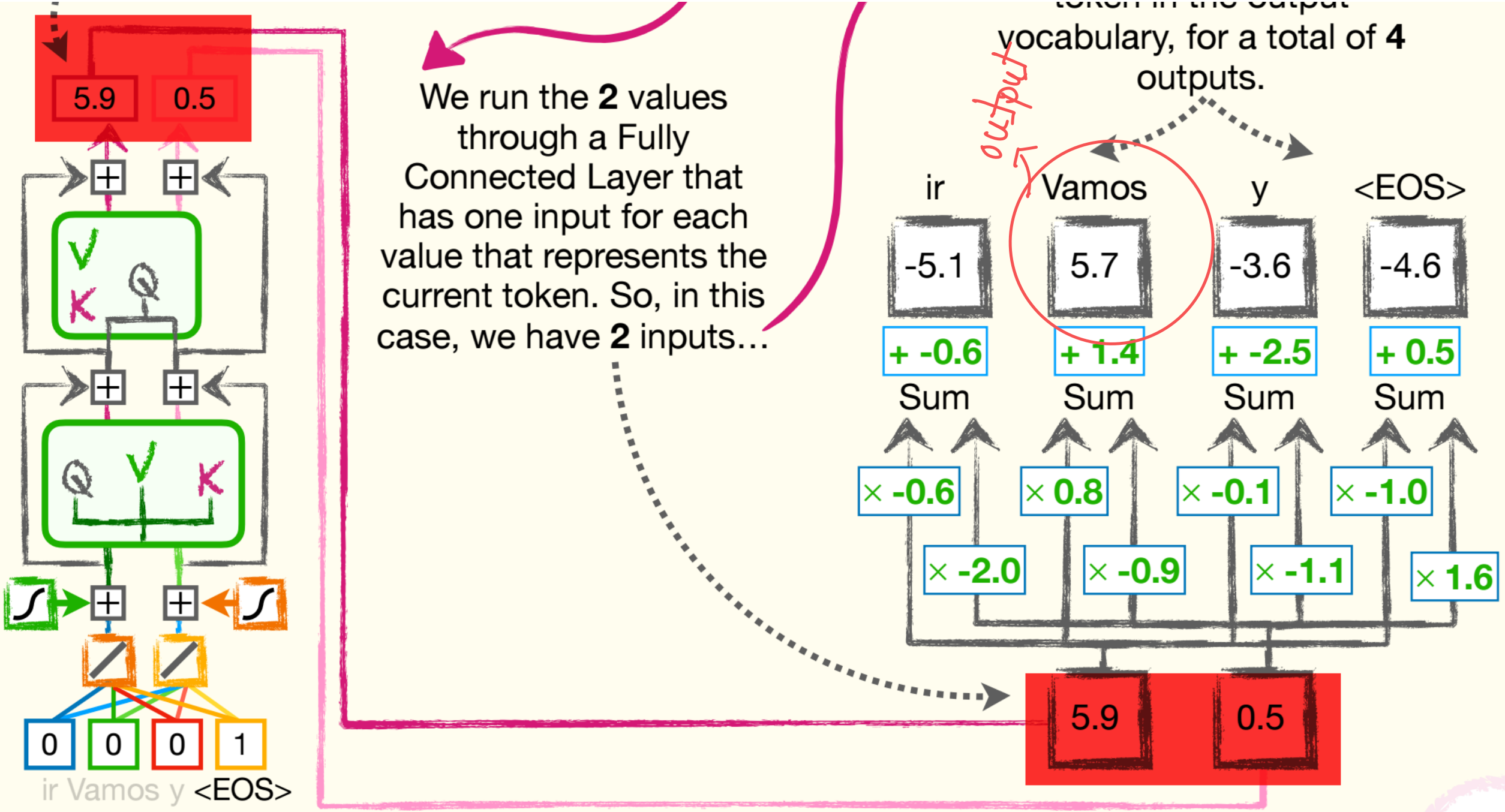


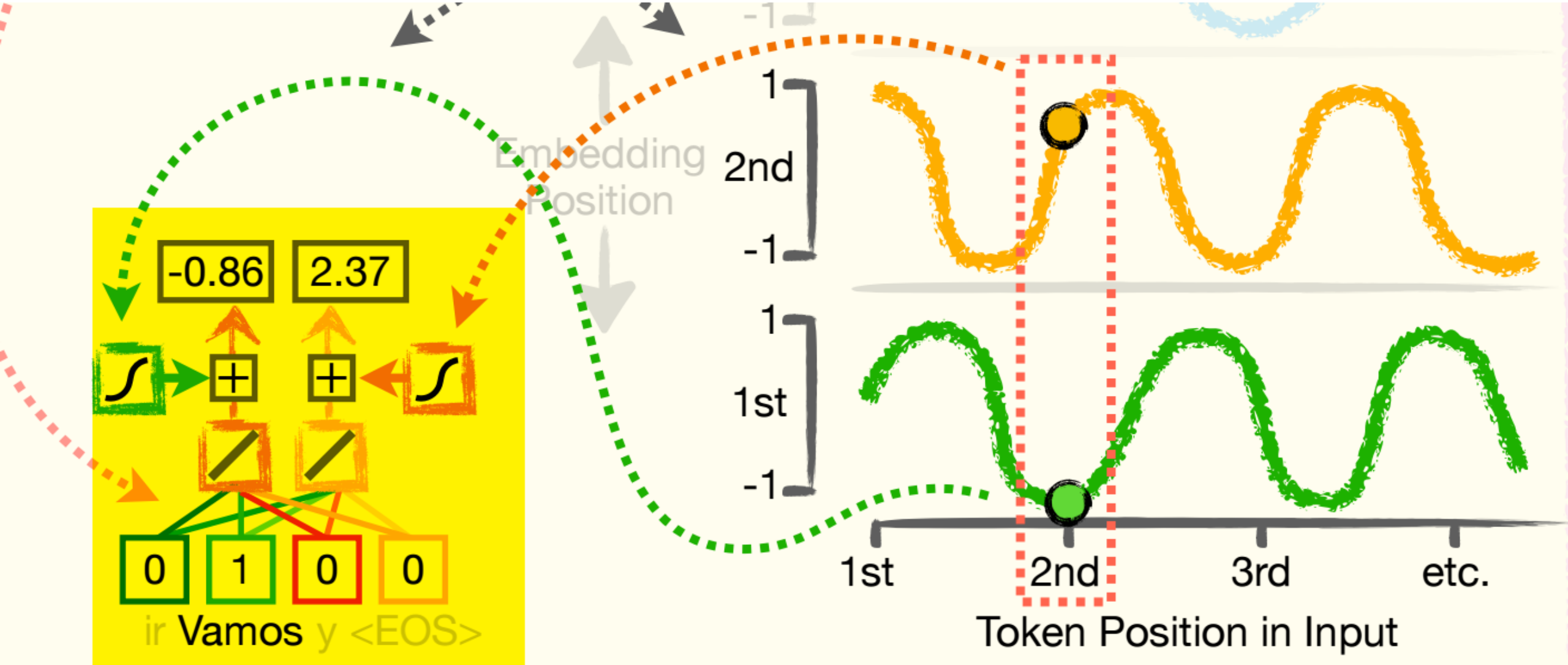

-4.5
Key
2.1
Sum
Sum
-0.5
Key
1.4
Sum
Sum
× -0.3
× -0.8
× -1.1
× 0.3
× -0.3
× -0.8
× -1.1
× 0.3
4.37
-1.01
Encoder
0.82
-1.43
Q
V
K
Q
V
K
1
0
0
0
0
0
1
0
Let's
to
go
<EOS>
Let's
to
go
<EOS>

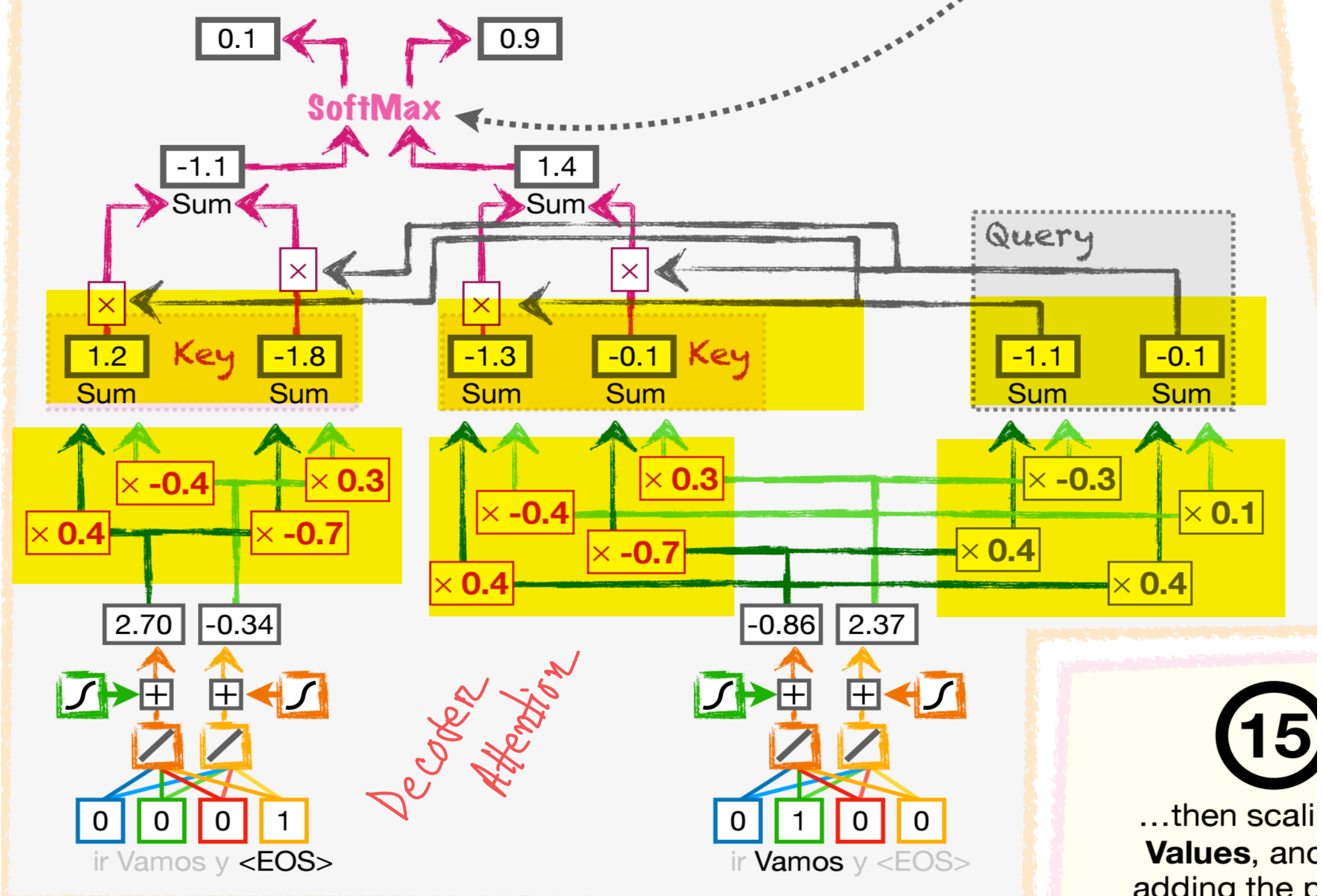

Let's 100% contribution.
1.0
0.0
Go 0% contribution.
Softmax
9.5
4.1
Sum
Sum
-4.5
Key
2.1
Sum
Sum
-0.5
Key
1.4
Sum
Sum
-0.9
Sum
2.6
Query
Sum
× -0.3
× -0.8
× -1.1
× 0.3
× -0.3
× -0.8
× -1.1
× 0.3
× 0.3
× -1.0
× 1.5
× -0.3
4.37
-1.01
Encoder
0.82
-1.43
-0.1
-2.6
Q
V
K
1
0
0
0
Let's
to
go
<EOS>
0
0
1
0
Let's
to
go
<EOS>
0
0
0
1
ir
Vamos
y
<EOS>


SoftMax relative to <EOS> in the Decoder
Let's
go
1.0
0.0
<EOS> token in the Decoder.
Bam!
6.0
3.1
6.0
Sum
Value
3.1
Sum
2.6
Sum
Value
1.2
Sum
× -1.2
× -0.5
× 1.1
× 0.6
× -1.2
× -0.5
× 1.1
× 0.6
Encoder
4.37
-1.01
0.82
-1.43
Encoder Decoder attention for Eos.
6.0
3.1
-0.1
-2.6
Q
V
K
1
0
0
0
Let's to go <EOS>
0
0
1
0
Let's to go <EOS>
0
0
0
1
ir Vamos y <EOS>


Encoder
5.9
0.5
6.0
3.1
V
K
Q
-0.1
-2.6
4.37
-1.01
0.82
-1.43
Q V K
1 0 0 0
0 0 1 0
0 0 0 1
Let's to go <EOS>
Let's to go <EOS>
ir Vamos y <EOS>
⇒ Attention + RC


5.9
0.5
V
K
Q
Q
V
K
0
0
0
1
ir Vamos y <EOS>
We run the 2 values through a Fully Connected Layer that has one input for each value that represents the current token. So, in this case, we have 2 inputs…
vocabulary, for a total of 4 outputs.
output
ir
Vamos
y
<EOS>
-5.1
5.7
-3.6
-4.6
+ -0.6
+ 1.4
+ -2.5
+ 0.5
Sum
Sum
Sum
Sum
× -0.6
× 0.8
× -0.1
× -1.0
× -2.0
× -0.9
× -1.1
× 1.6
5.9
0.5


Embedding
Position
-0.86
2.37
0
1
0
0
ir Vamos y <EOS>
1
2nd
-1
1
1st
-1
1st
2nd
3rd
etc.
Token Position in Input

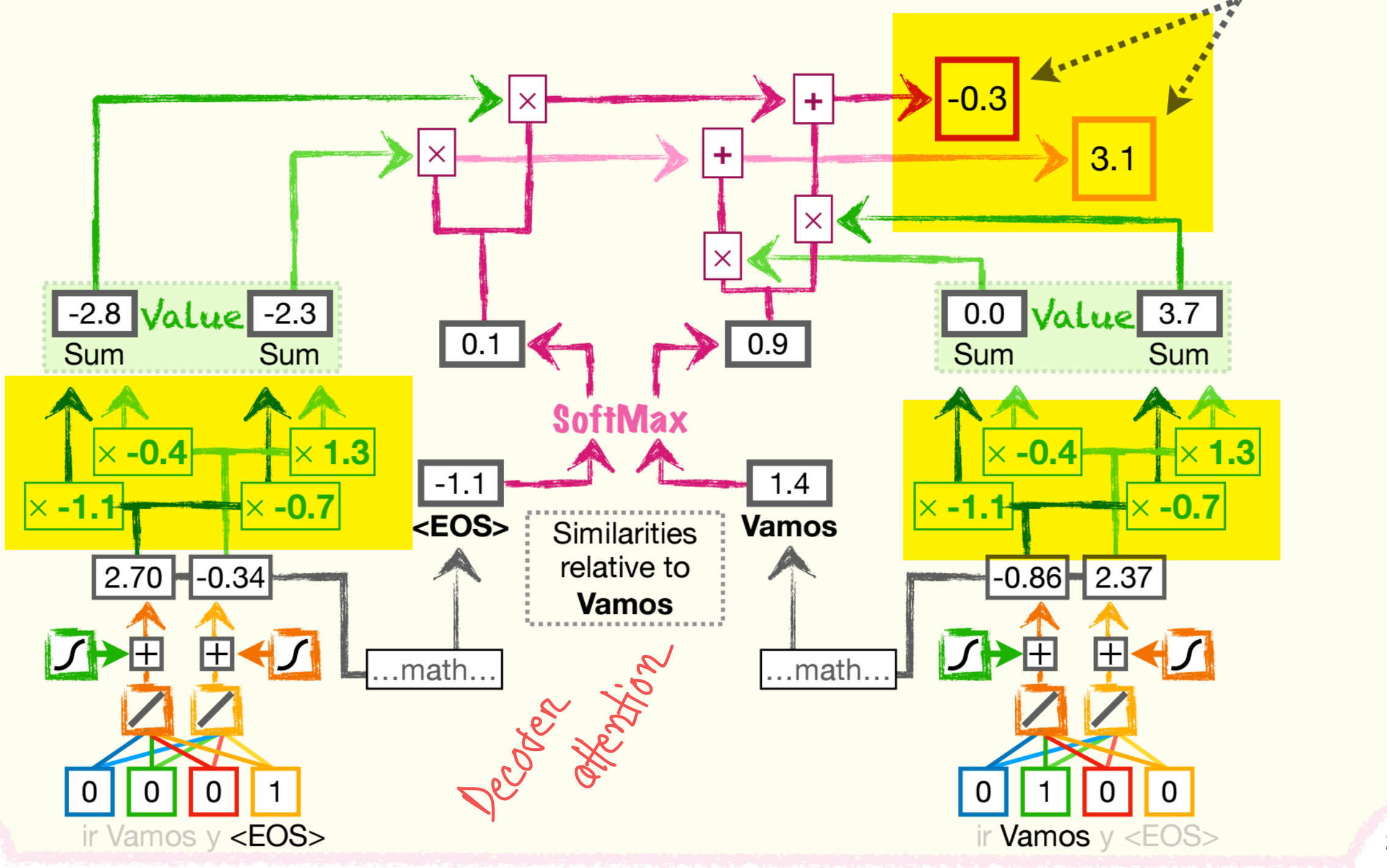


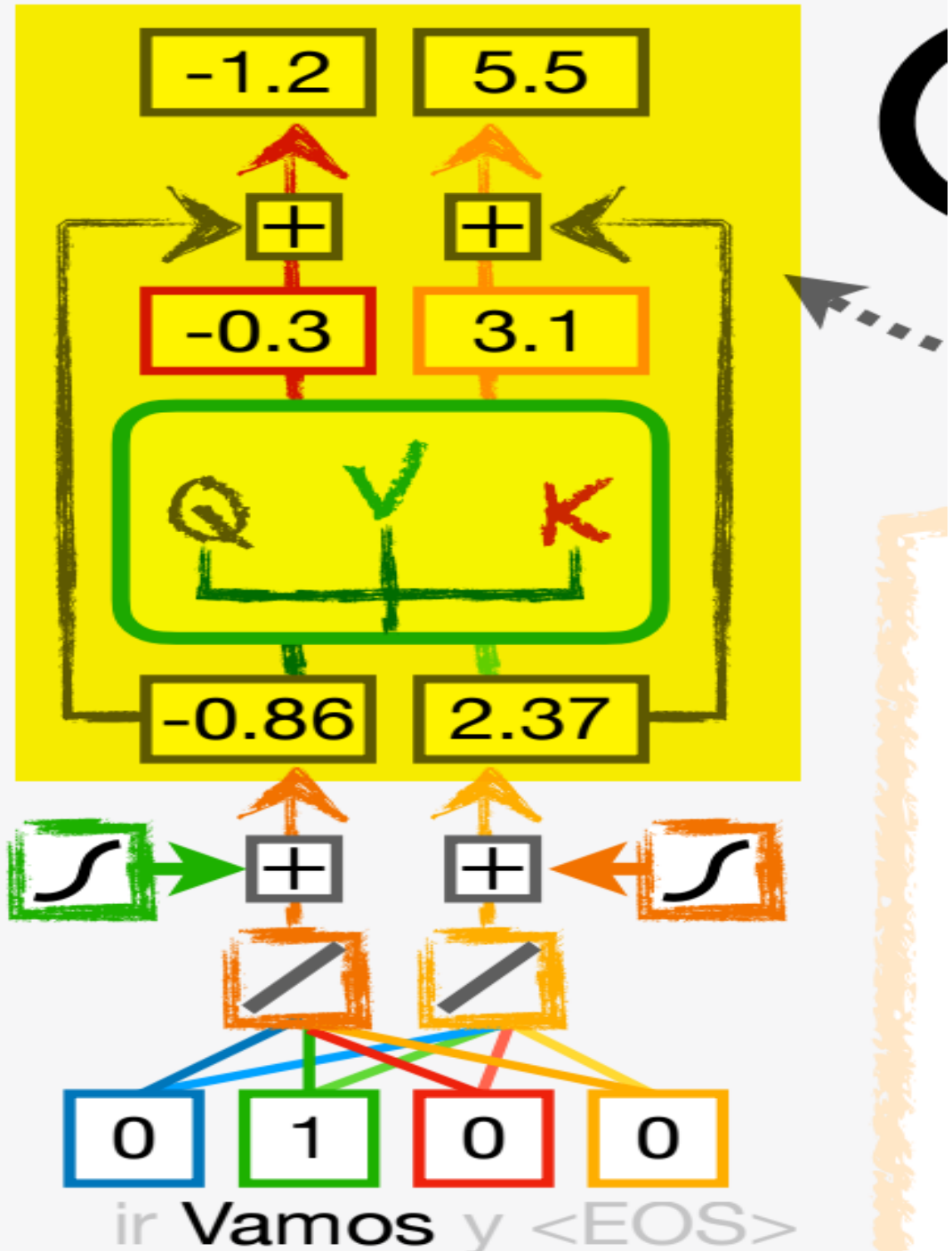
15

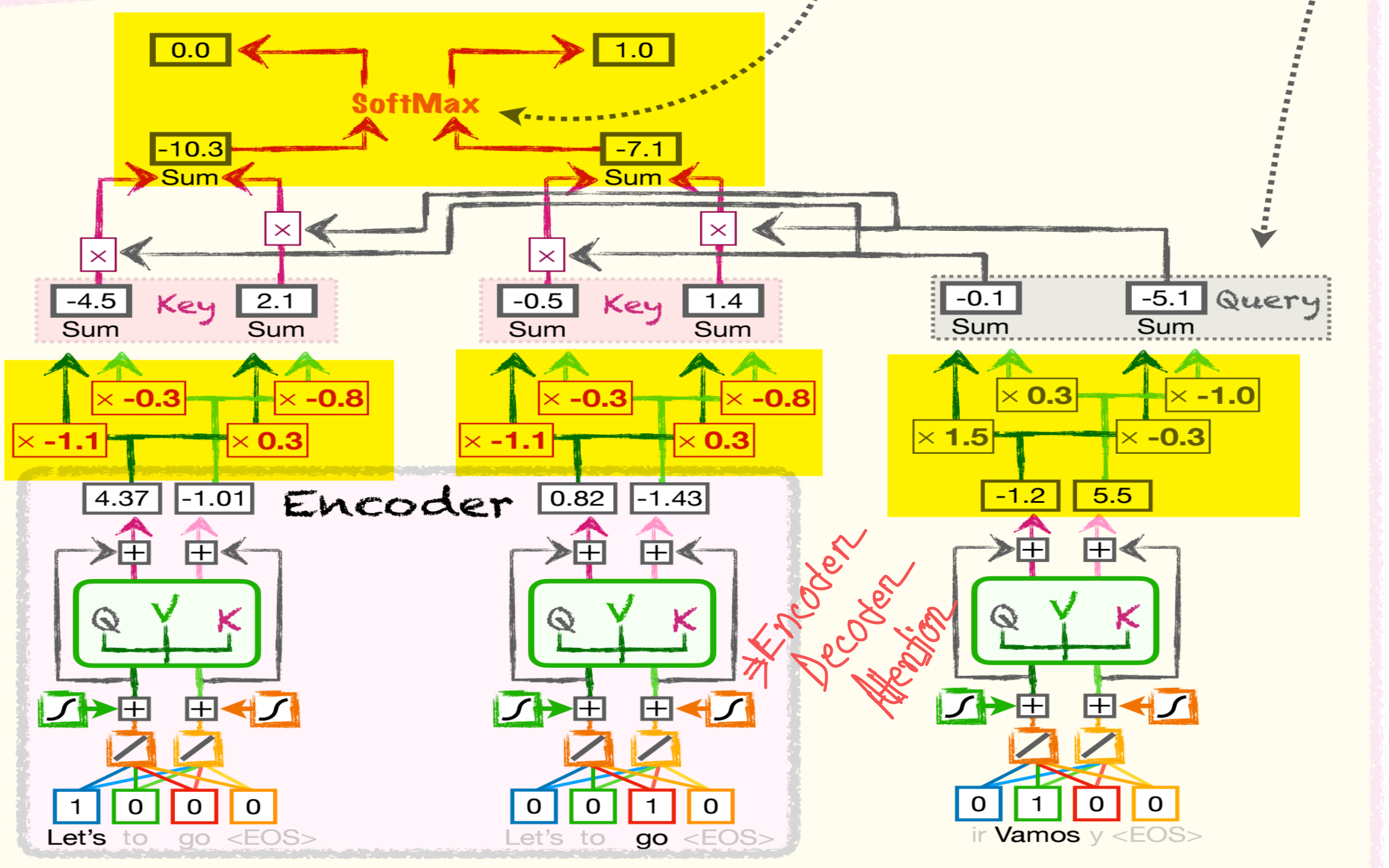
…then scali

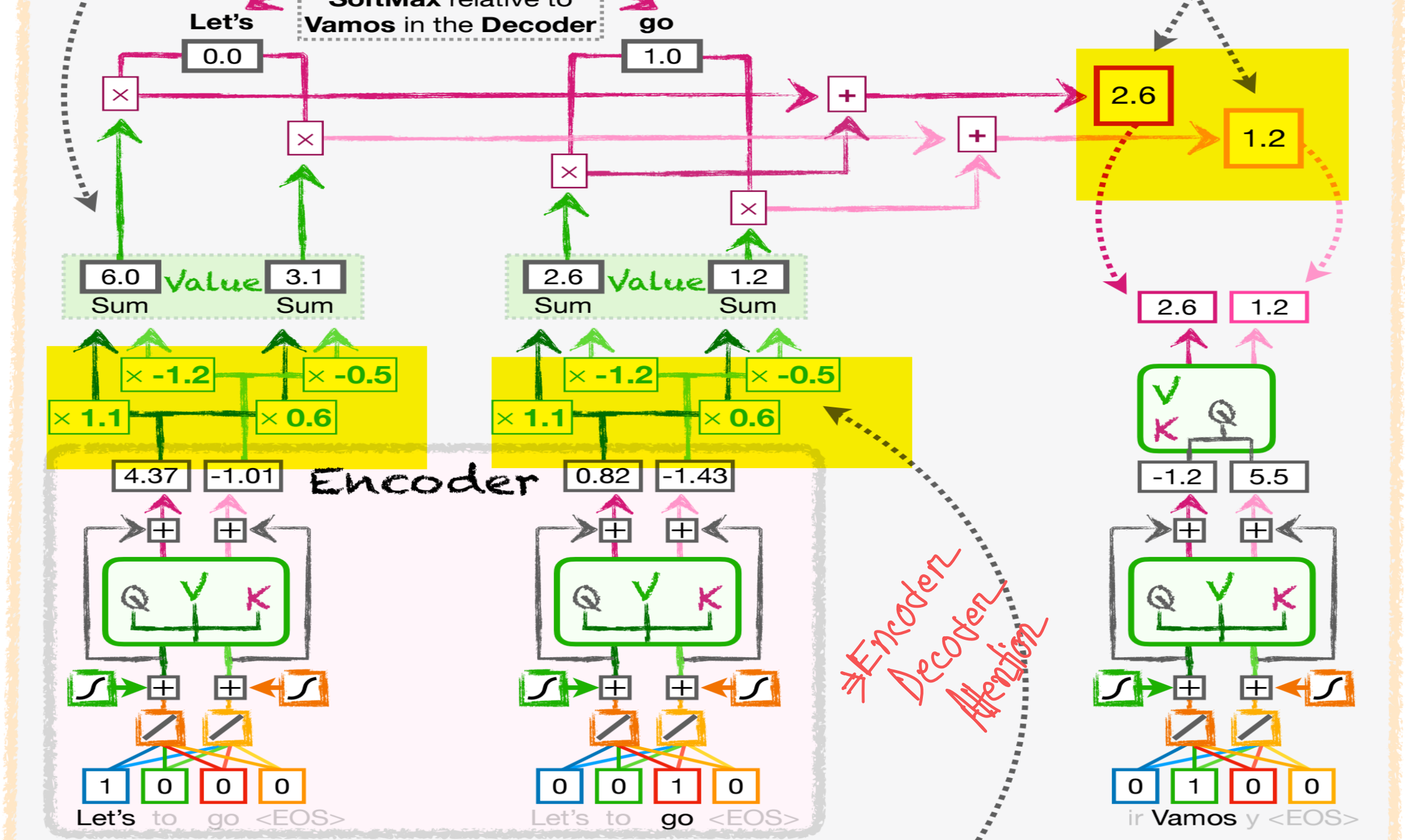
Values, and

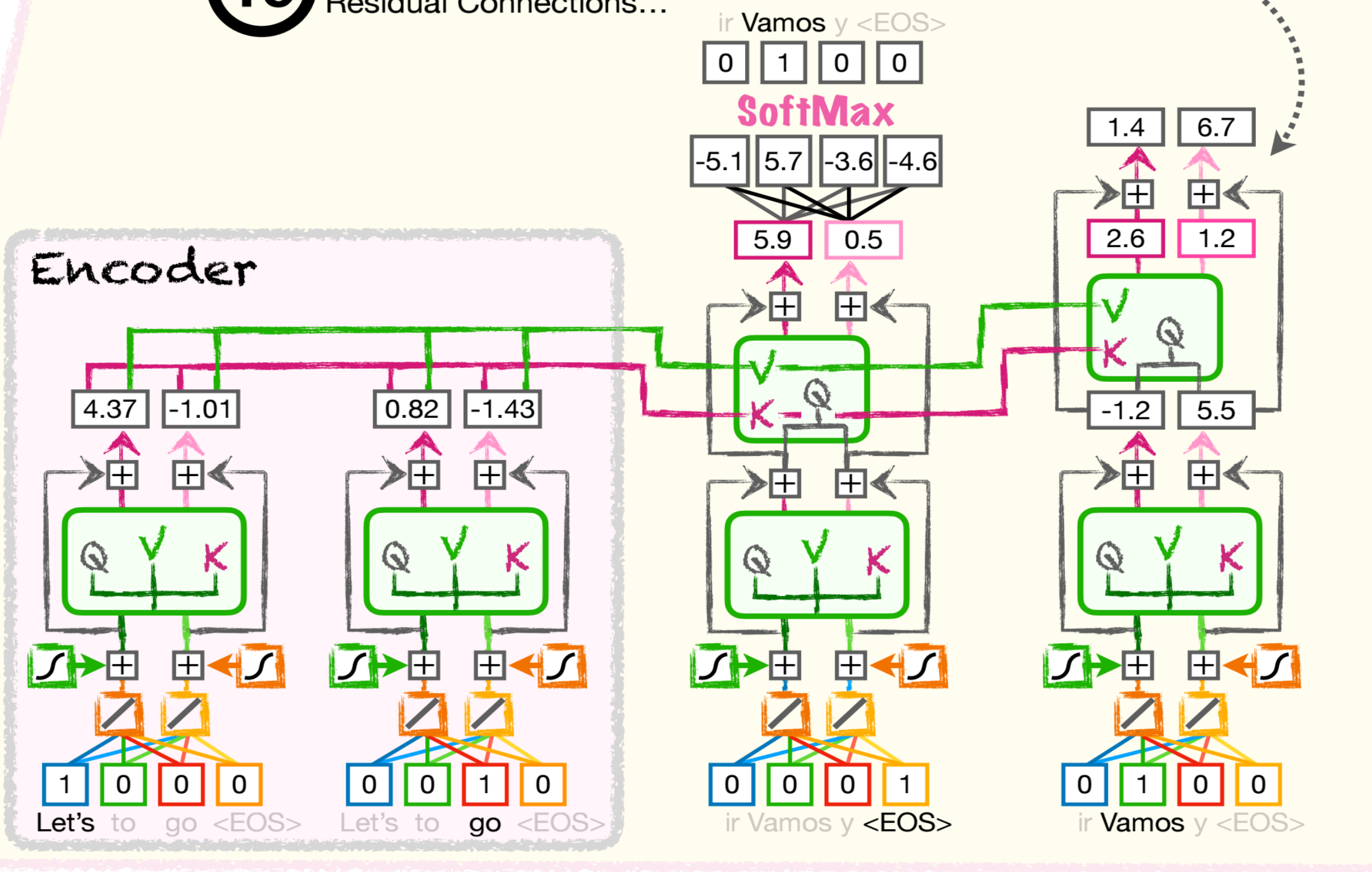
adding the p

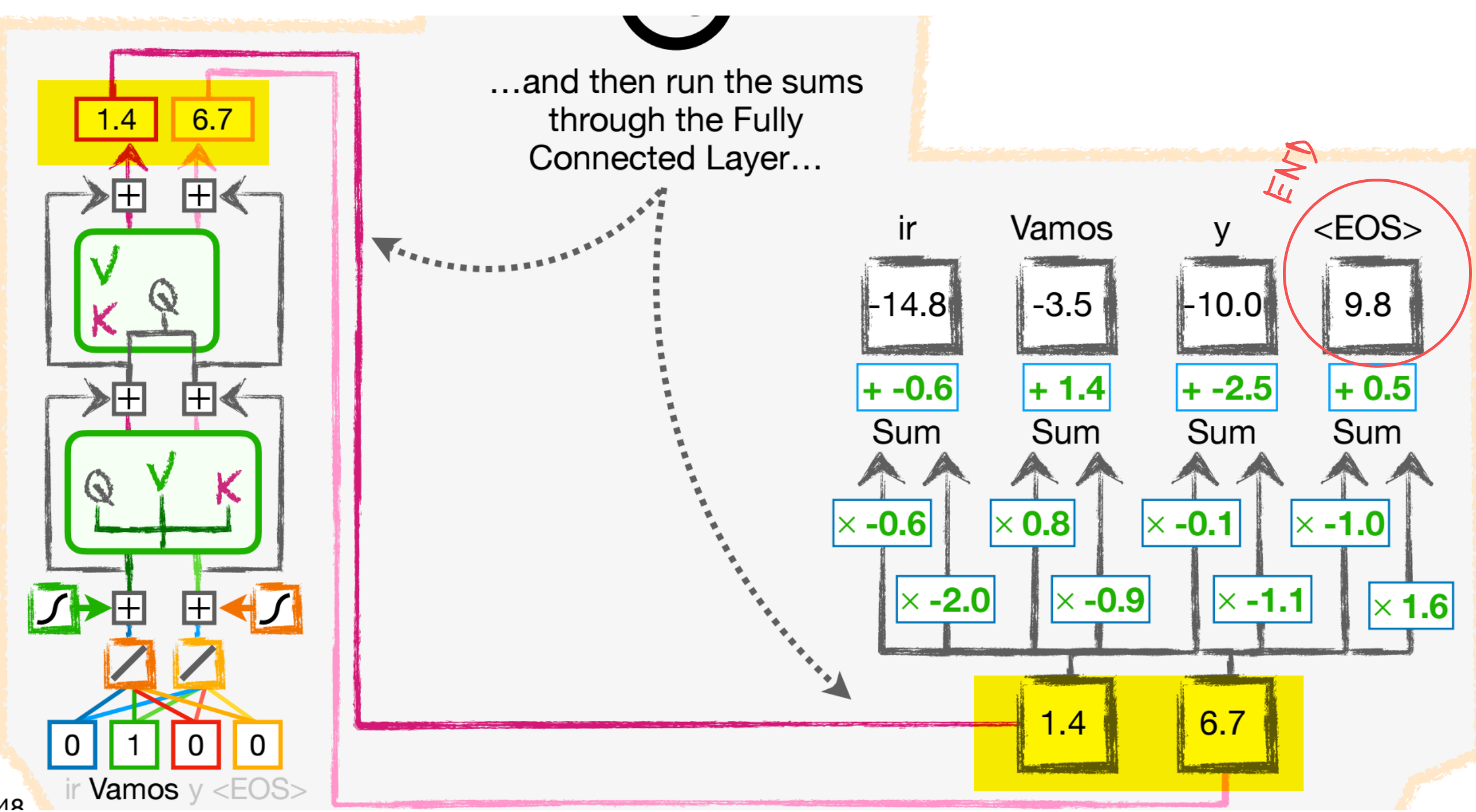

-0.3
3.1
-2.8
Value
-2.3
Sum
Sum
0.0
Value
3.7
Sum
Sum
0.1
0.9
SoftMax
× -0.4
× 1.3
× -1.1
× -0.7
-1.1
<EOS>
Similarities relative to Vamos
1.4
Vamos
× -0.4
× 1.3
× -1.1
× -0.7
2.70
-0.34
-0.86
2.37
...math...
...math...
0
0
0
1
0
1
0
0
Decoder attention
ir Vamos y <EOS>
ir Vamos y <EOS>

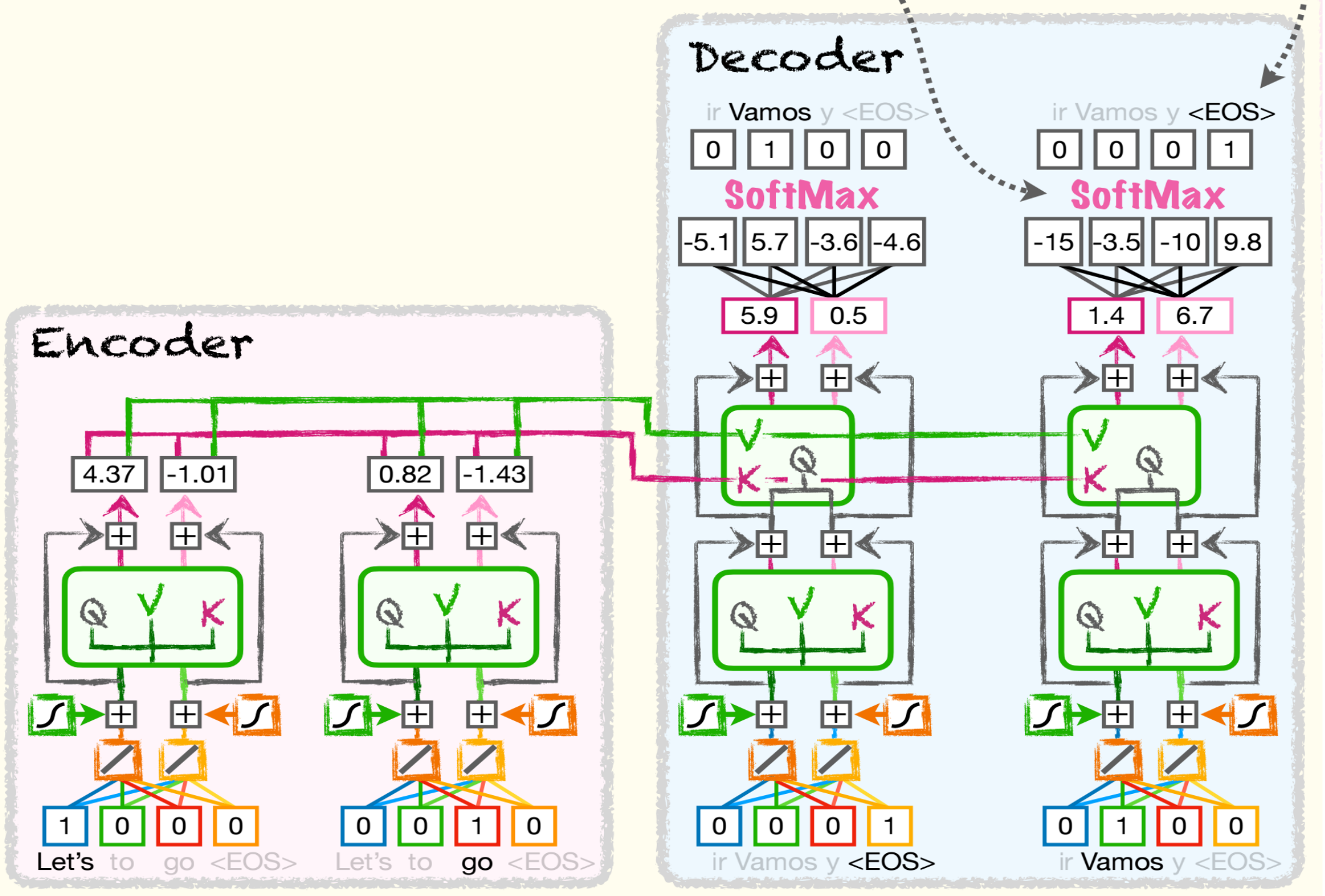

-1.2
5.5
-0.3
3.1
Q
V
K
-0.86
2.37
0
1
0
0
ir Vamos y <EOS>


0.0
1.0
SoftMax
-10.3
Sum
-7.1
Sum
-4.5
Key
2.1
Sum
-0.5
Key
1.4
Sum
-0.1
Sum
-5.1
Query
Sum
× -0.3
× -0.8
× -1.1
× 0.3
× -0.3
× -0.8
× -1.1
× 0.3
× 0.3
× -1.0
× 1.5
× -0.3
-1.2
5.5
4.37
-1.01
Encoder
0.82
-1.43
Q
V
K
Encoder Decoder Attention
1
0
0
0
Let's
to
go
<EOS>
0
0
1
0
Let's
to
go
<EOS>
0
1
0
0
ir
Vamos
y
<EOS>


SoftMax relative to
Vamos in the Decoder
Let's
go
0.0
1.0
2.6
1.2
6.0
Value
3.1
Sum
Sum
2.6
Value
1.2
Sum
Sum
× -1.2
× -0.5
× 1.1
× 0.6
× -1.2
× -0.5
× 1.1
× 0.6
4.37
-1.01
0.82
-1.43
2.6
1.2
-1.2
5.5
Encoder
Q
V
K
1
0
0
0
0
0
1
0
0
1
0
0
Let's to go <EOS>
Let's to go <EOS>
ir Vamos y <EOS>
Encoder Decoder Attention


Residual Connections…
ir Vamos y <EOS>
0 1 0 0
SoftMax
-5.1 5.7 -3.6 -4.6
5.9 0.5
1.4 6.7
2.6 1.2
-1.2 5.5
Encoder
4.37 -1.01
0.82 -1.43
V K Q
1 0 0 0
Let's to go <EOS>
0 0 1 0
Let's to go <EOS>
0 0 0 1
ir Vamos y <EOS>
0 1 0 0
ir Vamos y <EOS>


1.4
6.7
V
K
Q
0
1
0
0
ir Vamos y <EOS>
...and then run the sums through the Fully Connected Layer...
END
ir
Vamos
y
<EOS>
-14.8
-3.5
-10.0
9.8
+ -0.6
+ 1.4
+ -2.5
+ 0.5
Sum
Sum
Sum
Sum
× -0.6
× -2.0
× 0.8
× -0.9
× -0.1
× -1.1
× -1.0
× 1.6
1.4
6.7


Decoder
ir Vamos y <EOS>
0 1 0 0
SoftMax
-5.1 5.7 -3.6 -4.6
5.9 0.5
ir Vamos y <EOS>
0 0 0 1
SoftMax
-15 -3.5 -10 9.8
1.4 6.7
Encoder
4.37 -1.01
0.82 -1.43
V K Q
1 0 0 0
Let's to go <EOS>
0 0 1 0
Let's to go <EOS>
0 0 0 1
ir Vamos y <EOS>
0 1 0 0
ir Vamos y <EOS>

Encoder

Lets	go
[1.87, 0.09]	[-0.78, 0.27]
Positional encoding	
+[0.0,1,0]	+[-0.9,0.4]
[1.87,1.09]	[-1.68,0.67] ……(1)
Attention calculation in encoder (Lets go)	
Q=1.1 0.6 -2.8 2.4	K = -1.7 0.5 -1.5 0.9
V = 1.5 -1.0 -0.3 -0.2	
QL = Lets X Q KL = Lets X K VL = Lets X V QG = go X Q KG = go X K VG = go X V	
QL = [-1.0, 3.7] KL = [-4.7,1.9] VL = [2.5,-2.1] QG =[-3.7, 0.6] KG = [1.9,-0.2] VG = [-2.7,1.5]	
QLxKL = 11.7 QLxKG = -2.6 QGxKL = 18.5 QGxKG = -7.2	
Softmax(11.7,-2.6) 1.0, 0.0	Softmax(18.5.-7.2) 1.0, 0.0
VLx1.0 + VGx0 [2.5, -2.1]	VLx1.0 + VGx0.0 [2,5, -2.1]

From (1) [1.87,1.09]	[-1.68,0.67]
+ [2.5, -2.1]	+ [2.5, -2.1]
[4.37, -1.01]	[0.82, -1.43]

Decoder

Attention calculation in decoder (EOS)
EOS [2.70, -1.34] + positional encoding [0.0, 1.0] = [2.70,-0.34]
Q=0.4 0.4 -0.3 0.1
K = 0.4 -0.7 -0.4 0.3
V = -1.1 -0.7 -0.4 1.3
QE = EOSXQ KE = EOSXK VE = EOSXV QE = [1,2,1.0] KE = [1.2, -1.8] VE = [-2.8, -2.3]
QEXKE = -0.4 Softmax(-0.4) = 1.0
EOS[2.70, -0.34] + VE X 1.0 = [-0.1, -2.6]……(2)

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

$$PE(pos, 2i_{+1}) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

Decoder Encoder cross attention calculation (EOS, Lets, go)

Q=1.5 -0.3 K = -1.1 0.3 V = 1.1 0.6
0.3 -1.0 -0.3 -0.8 -1.2 -0.5

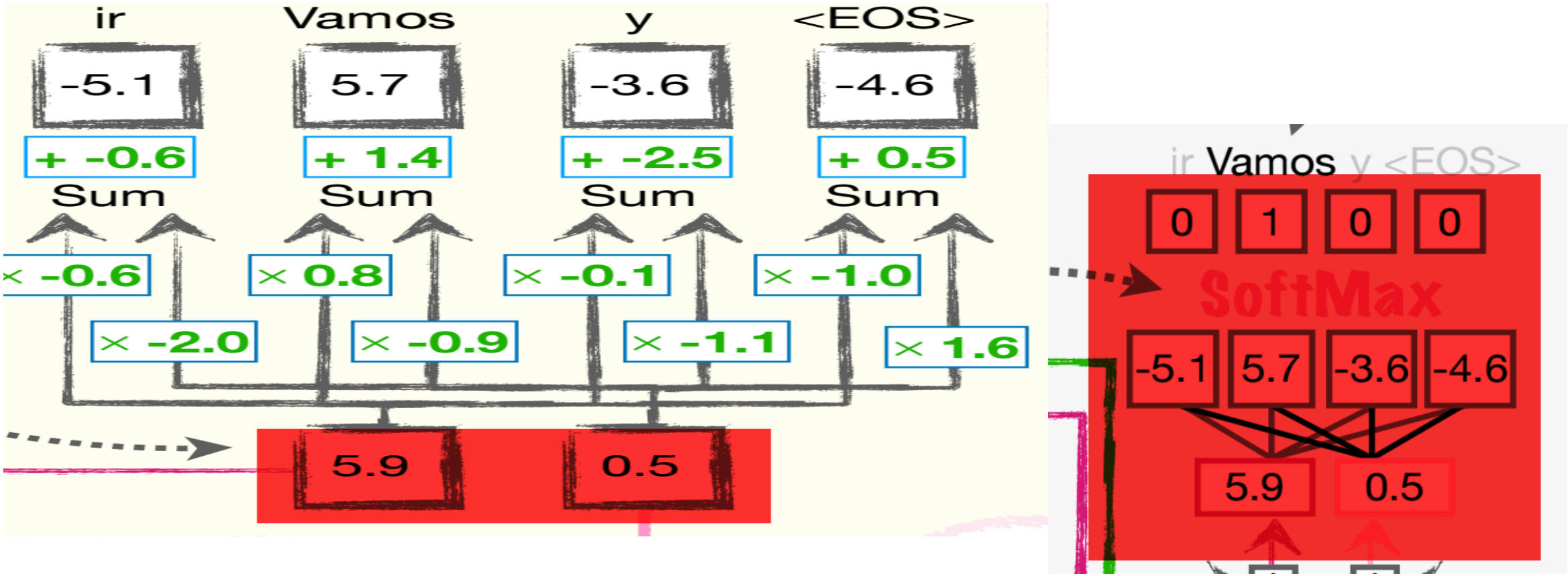
QE = EOS X Q KL = Lets X K KG = go X K
VL = Lets X V VG = go X V

QE = [-0.9, 2.6] KL = [-4.5,2.1] VL = [6.0, 3.1]
KG = [-0.5,1.4] VG = [2.6,1.2]

QExKL = 9.5 QExKG = 4.1
Softmax(9.5,4.1)
1.0, 0.0
VLx1.0 + VGx0
[6.0, 3.1]

From (2) EOS [-0.1, -2.6] + [6.0.3.1] = [5.9,0.5] ……(3)

Generating first token in decoder after processing EOS



Decoder

Attention calculation in decoder (Vamos, EOS)

Vamos [-0.86,2.37] + positional encoding [-0.9,0.4] = [-0.86,2.37]

Q=0.4 0.4 K = 0.4 -0.7 V = -1.1 -0.7
-0.3 0.1 -0.4 0.3 -0.4 1.3

KE = EOSXK VE = EOSXV QV = Vamos X Q VV = vamos X V
KE = [1.2, -1.8] VE = [-2.8, -2.3] QV = [-1.3, -0.1] VV = [0.0, 3.7]

QVXKE = -1.1 QVXKV = 1.4
Softmax(-1.1, -1.4) = 0.1,0.9
varmos[-0.86, 2.37] + VE X 0.1 + VVX0.9 = [-1.2, 5.5]
…….(4)

Decoder Encoder cross attention (Vamos, Lets, go)

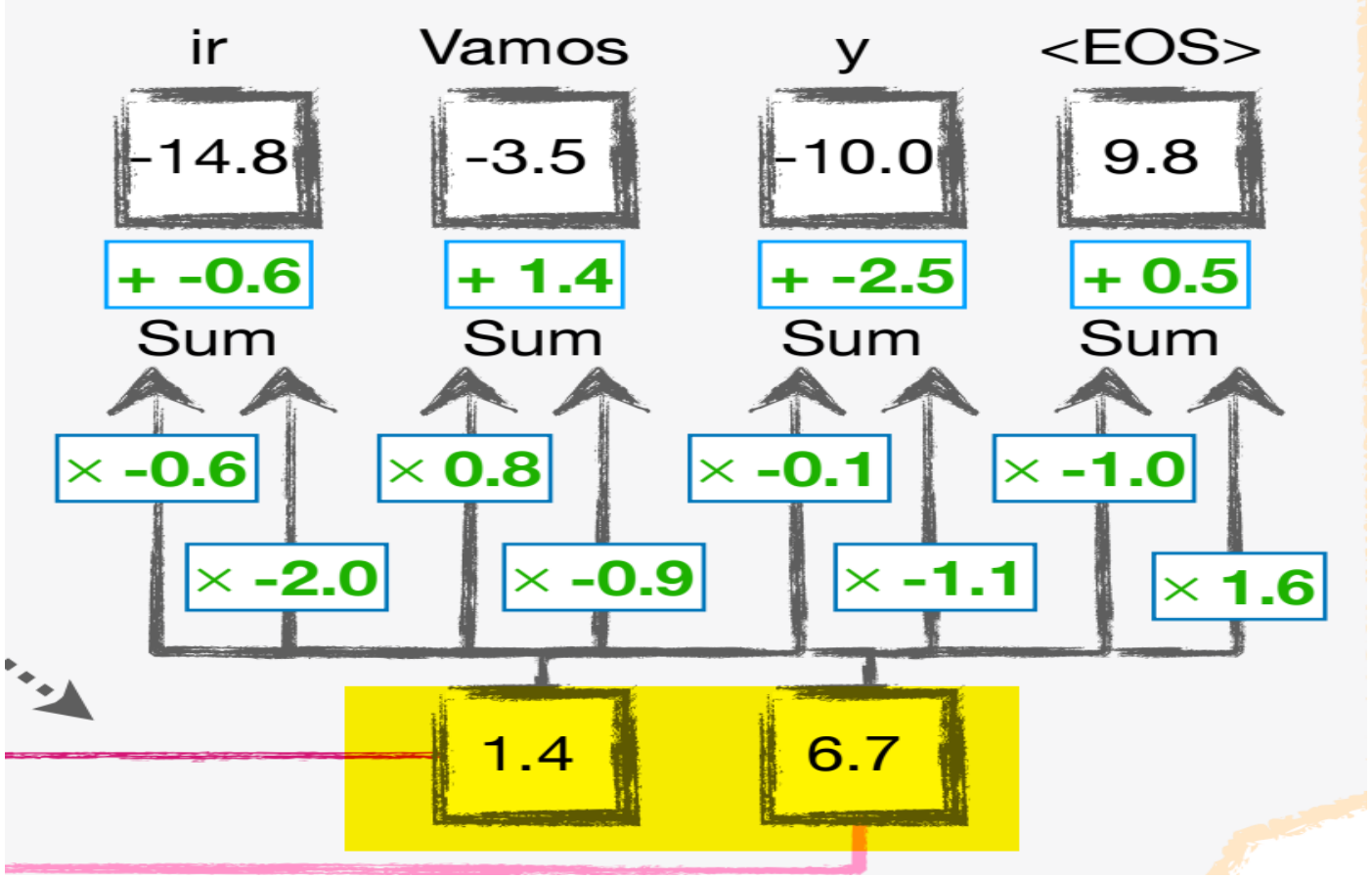
Q=1.5 -0.3 K = -1.1 0.3 V = 1.1 0.6
0.3 -1.0 -0.3 -0.8 -1.2 -0.5

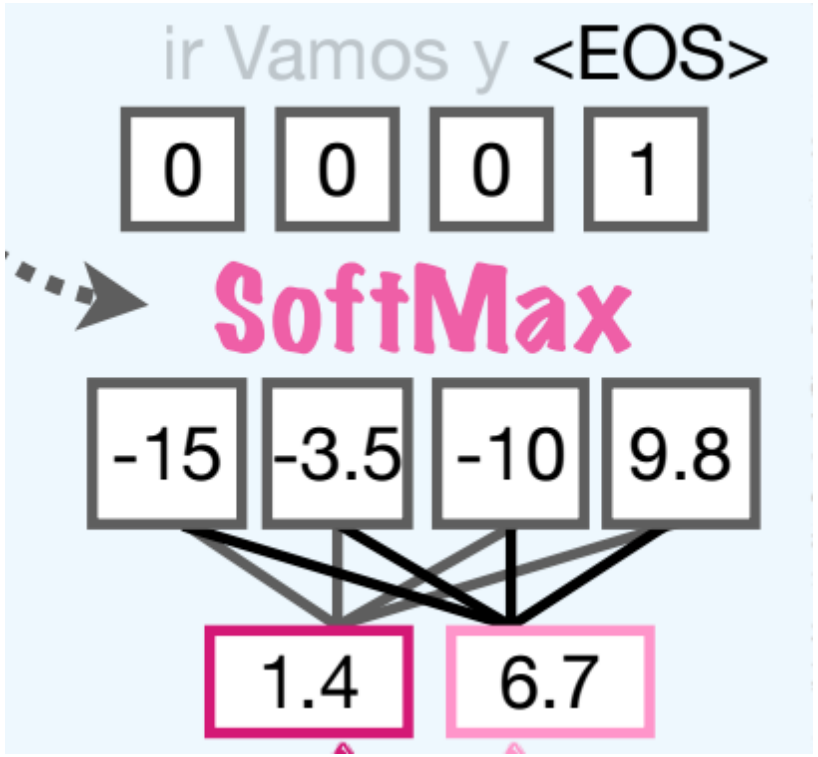
QV = vamos X Q KL = Lets X K KG = go X K
VL = Lets X V VG = go X V

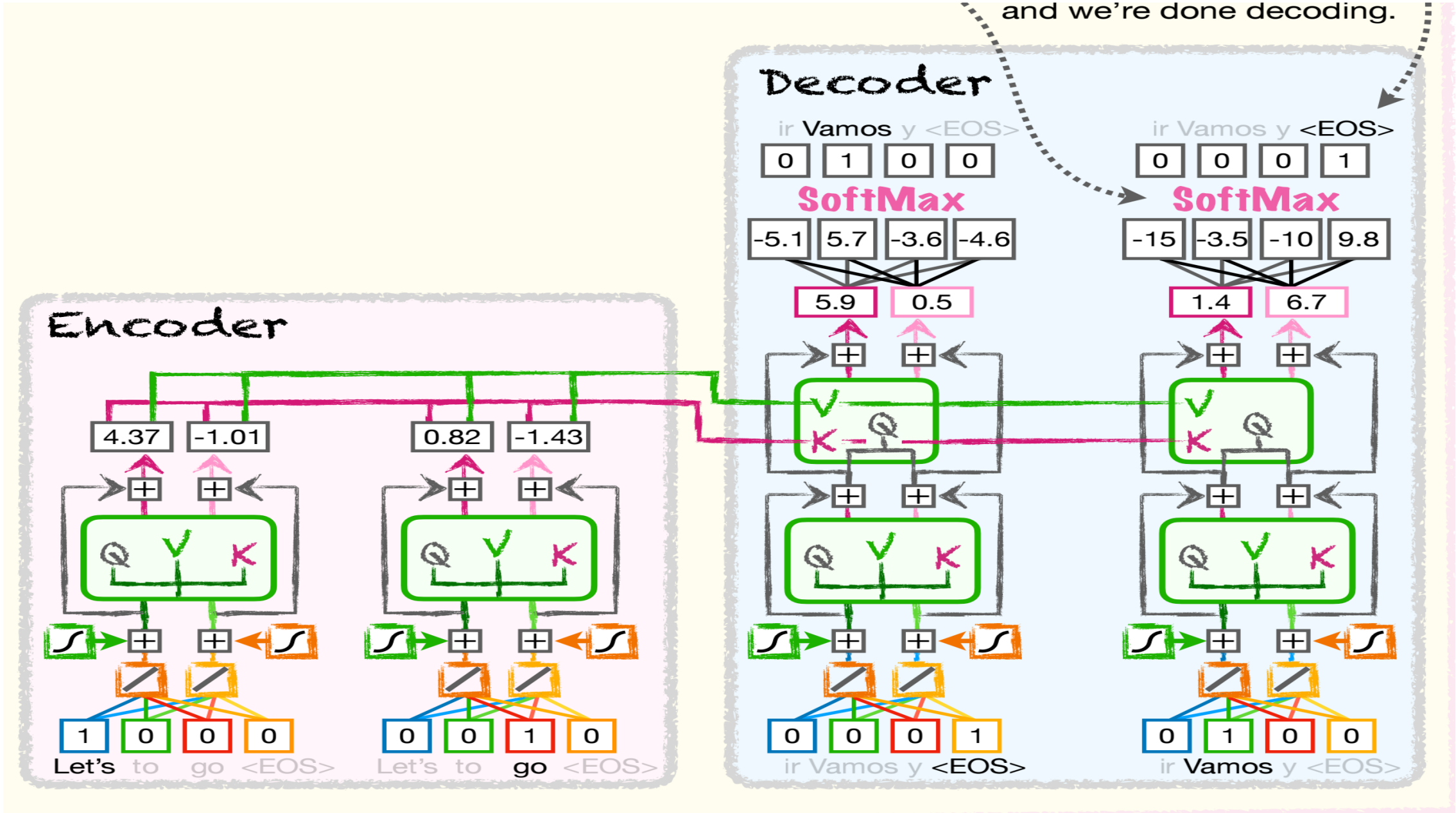
QV = [-0.1, -5.1] KL = [-4.5,2.1] VL = [6.0, 3.1]
KG = [-0.5,1.4] VG = [2.6,1.2]

QVxKL = -10.3 QVxKG = -7.1
Softmax(-10.3,-7.1)
0.0, 1.0
VLx0.0 + VGx1.0
[2.6,1.2]

From (4) Vamos [-1.2,5.5] + [2.6,1.2] = [1.4,6.7]
……(5)


ir
Vamos
y
<EOS>
-14.8
-3.5
-10.0
9.8
+ -0.6
+ 1.4
+ -2.5
+ 0.5
Sum
Sum
Sum
Sum
× -0.6
× -2.0
× 0.8
× -0.9
× -0.1
× -1.1
× -1.0
× 1.6
1.4
6.7


ir Vamos y <EOS>
0
0
0
1
SoftMax
-15
-3.5
-10
9.8
1.4
6.7


and we're done decoding.
Decoder
ir Vamos y <EOS>
0 1 0 0
SoftMax
-5.1 5.7 -3.6 -4.6
5.9 0.5
ir Vamos y <EOS>
0 0 0 1
SoftMax
-15 -3.5 -10 9.8
1.4 6.7
Encoder
4.37 -1.01
0.82 -1.43
V K Q
1 0 0 0
Let's to go <EOS>
0 0 1 0
Let's to go <EOS>
0 0 0 1
ir Vamos y <EOS>
0 1 0 0
ir Vamos y <EOS>