Question Duplicates

How old are you? = What is your age?

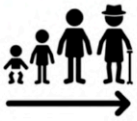
Where are you from? ≠ Where are you going?

What do Siamese Networks learn?

I am happy because I am learning 😃 🙁

Classification: categorize things

Siamese Networks: Identify similarity between things

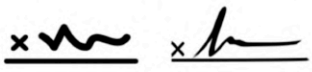


What is your age?
How old are you?
} Difference or Similarity

Siamese Networks in NLP

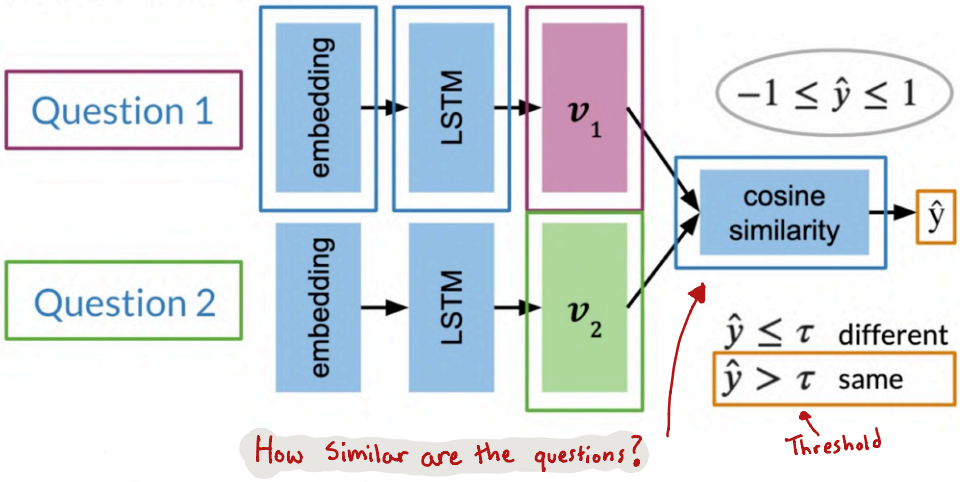
What is your age?
How old are you?

Handwritten checks | Question duplicates | Queries

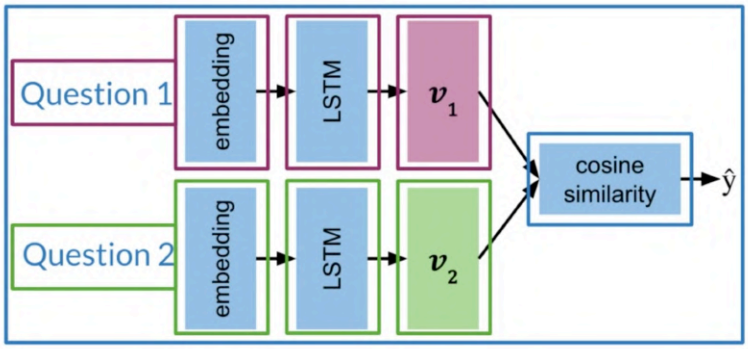


An important note here is that the sub-networks share identical parameters. That is the learned parameters of each sub-network are exactly the same. So you actually only need to train one sets of weights, not two.

Model Architecture

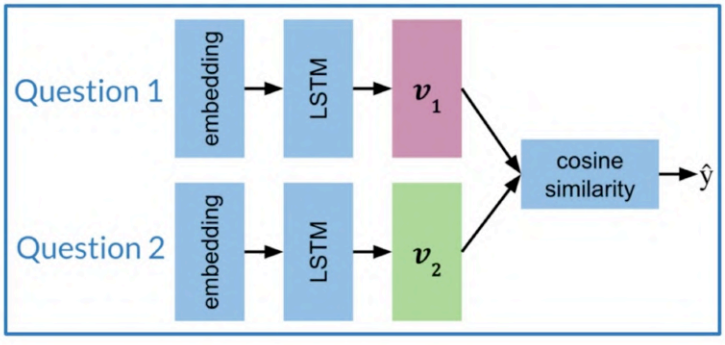


Model Architecture



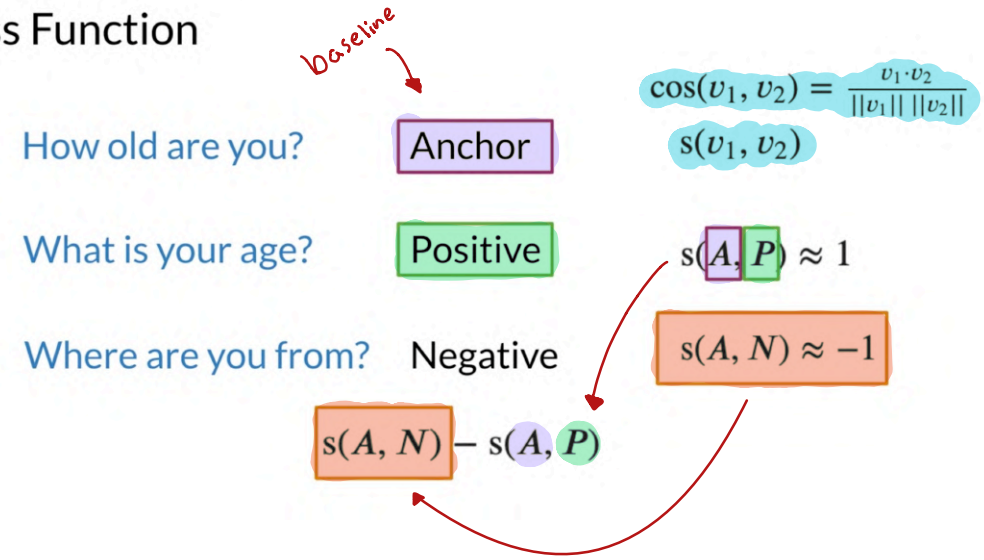
1) Inputs
2) Embedding
3) LSTM
4) Vectors
5) Cosine Similarity

Loss Function

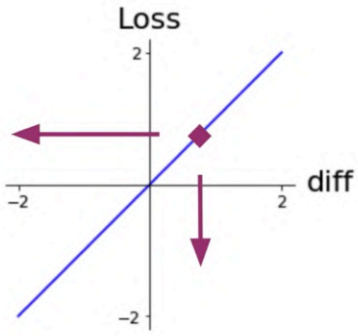


$$\hat{y} = s(v_1, v_2)$$

Loss Function



Loss Function



$$\text{diff} = s(A, N) - s(A, P)$$

Triplets

How old are you? A

What is your age? P

Where are you from? N

Triplets !!!

Triplet Loss

How old are you? A

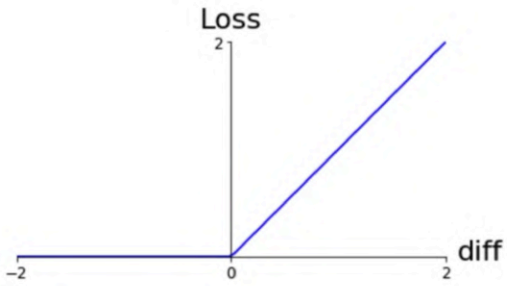
What is your age? P

Where are you from? N

Simple loss:

$$\text{diff} = \text{s}(A, N) - \text{s}(A, P)$$

Triplet Loss



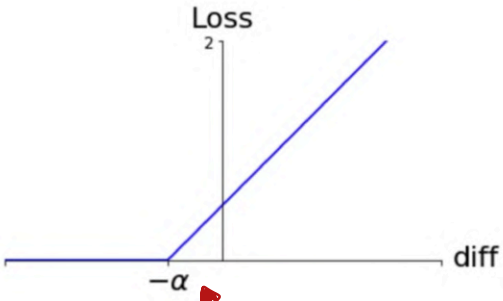
Simple loss:

$$\text{diff} = \text{s}(A, N) - \text{s}(A, P)$$

Non linearity:

$$\mathcal{L} = \begin{cases} 0; & \text{if } diff \leq 0 \\ diff; & \text{if } diff > 0 \end{cases}$$

Triplet Loss



Simple loss:

$$\text{diff} = \text{s}(A, N) - \text{s}(A, P)$$

Non linearity:

$$\mathcal{L} = \begin{cases} 0; & \text{if } diff \leq 0 \\ diff; & \text{if } diff > 0 \end{cases}$$

Alpha margin:

$$\mathcal{L} = \begin{cases} 0; & \text{if } diff + \alpha \leq 0 \\ diff + \alpha; & \text{if } diff + \alpha > 0 \end{cases}$$

Triplet Loss

Similarity

$s(v_1, v_2)$ $d(v_1, v_2)$ 👍

$$\mathcal{L} = \begin{cases} 0; & \text{if } diff + \alpha \leq 0 \\ diff; & \text{if } diff + \alpha > 0 \end{cases}$$

$$\mathcal{L}(A, P, N) = \max(diff + \alpha, 0)$$

A distance metric is the mirror image of a similarity metric, and a similarity metric can be derived from a distance metric.

Triplet Selection

Hard triplets are better for training !

Triplet A, P, N: { duplicate set: A, P; non-duplicate set: A, N

Random:

$$\mathcal{L} = \max(diff + \alpha, 0)$$

$$\text{diff} = s(A, N) - s(A, P)$$

Easy to satisfy. Little to learn ❌

Hard:

$$s(A, N) \approx s(A, P)$$

Harder to train. More to learn ✓

Hard triplets are those where the similarity between anchor and negative is very close to, but still smaller than the similarity between anchor and positive. When the model encounters a hard triplet, the learning algorithm needs to adjust its weight, so that's it's going to yield similarities that line up with the real-world labels. So by selecting hard triplets, focusing the training on doing better, on the difficult cases, that it's predicting incorrectly.

Computing The Cost

Prepare the batches as follows: ❌

Duplicates among rows

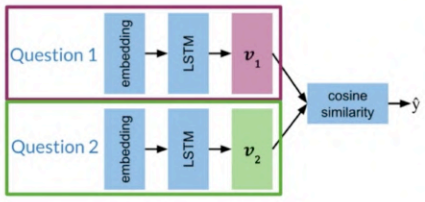
What is your age?	How old are you?
Can you see me?	Are you seeing me?
Where are thou?	Where are you?
When is the game?	What time is the game?

✅

No duplicates among columns

b = 4

Computing The Cost



Each question in the batch

v_1 = (1, d_model)

Batch 1

- What is your age?
- Can you see me?
- Where are thou?
- When is the game?

4 rows

v_{1_1} v_{1_2} v_{1_3} v_{1_4}

Batch 2

- How old are you?
- Are you seeing me?
- Where are you?
- What time is the game?

v_2

v_{2_1} v_{2_2} v_{2_3} v_{2_4}

5

Computing The Cost

The diagonal is a key feature here. These values are the similarities for all your positive examples, the question duplicates. Notice that all the values are generally greater than the numbers in the off diagonals. So the model is performing as you would expect for duplicates questions, because you would expect that the question duplicates to have higher similarity compared to the non-duplicates.

$s(v_1, v_2)$

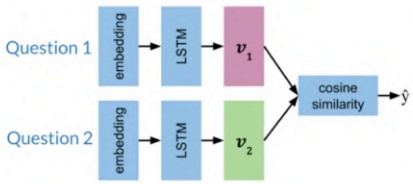
v_2 \ v_1	_1	_2	_3	_4
_1	0.9	-0.8	0.3	-0.5
_2	-0.8	0.5	0.1	-0.2
_3	0.3	0.1	0.7	-0.8
_4	-0.5	-0.2	-0.8	1.0

$$\mathcal{L}(A, P, N) = \max(diff + \alpha, 0)$$
$$\text{diff} = s(A, N) - s(A, P)$$

$$\mathcal{J} = \sum_{i=1}^{m} \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)})$$

In the upper right and lower left, you have the similarities for all the negative examples. These are the results for the non-duplicates pairs. Notice that most of these numbers are lower than the similarities that's are along the diagonal. Also notice that you can have negative example question pairs that still have a similarity greater than zero. The range of similarity ranges from negative 1 to positive 1, but there isn't any special requirements that a similarity greater than zero indicates duplicates or that's a similarity less than zero indicates non-duplicates. What's matters for a properly functioning model is that it generally finds that duplicates have a higher similarity relative to non-duplicates. Creating non-duplicates pairs like this removes the need for additional non-duplicate examples and the input data, which turns out to be a big deal. Instead of needing to sets up specific batches with negative examples, your model can learn from them in the existing question duplicates batches.

Computing The Cost



v_1 = (1, d_model)

Batch 1

- What is your age?
- Can you see me?
- Where are thou?
- When is the game?

v_{1_1} v_{1_2} v_{1_3} v_{1_4}

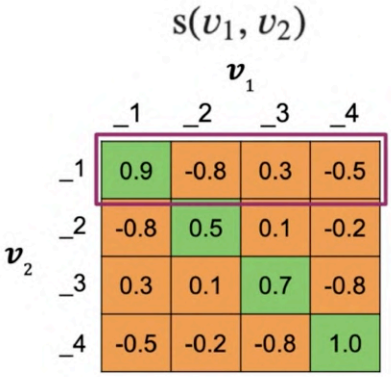
Batch 2

- How old are you?
- Are you seeing me?
- Where are you?
- What time is the game?

v_2

v_{2_1} v_{2_2} v_{2_3} v_{2_4}

Hard Negative Mining



mean negative:
mean of off-diagonal values in each row

closest negative:
off-diagonal value closest to (but less than) the value on diagonal in each row

Hard Negative Mining

mean negative: mean of off-diagonal values

closest negative: closest off-diagonal value

$$\mathcal{L}_{\text{Original}} = \max(\underbrace{s(A,N) - s(A,P) + \alpha}_{\text{diff}}, 0)$$

$$\mathcal{L}_1 = \max(mean\_neg - s(A,P) + \alpha, 0)$$

$$\mathcal{L}_2 = \max(closest\_neg - s(A,P) + \alpha, 0)$$

$$\mathcal{L}_{\text{Full}} = \mathcal{L}_1 + \mathcal{L}_2$$

I'll introduce loss 1 to be the max of the mean negative minus the similarity of A and P plus alpha and 0. The change between the formulas for triplet loss and loss 1 is the replacement of similarity of A and N. With the mean negative, this helps the model converge faster during training by reducing noise. It reduces noise by training on just the average of several observations, rather than training the model on each of these off-diagonal examples. So why does taking the average of several observations usually reduce noise? Well, we define noise to be a small value that comes from a distribution that is centered around 0. So in other words, the average of several noise values is usually 0. So if we took the average of several examples, this has the effect of cancelling out the individual noise from those observations.

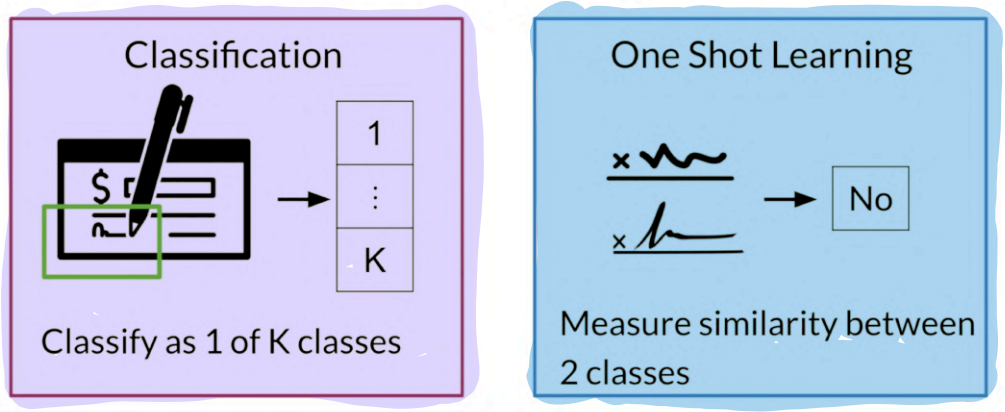
Then loss 2 will be the max of the closest negative minus the similarity of A and B plus alpha and 0. The difference between the formulas this time is the replacement of the cosine of A and N. With the closest negative, this helps create a slightly larger penalty by diminishing the effects of the otherwise more negative similarity of A and N that it replaces. You can think of the closest negative as finding the negative example that results in the smallest difference between the two cosine similarities. If you had that small difference to alpha, then you're able to generate the largest loss among all of the other examples in that row. By focusing the training on the examples that produce higher loss values, you make the model update its weights more.

Hard Negative Mining

$$\mathcal{L}_{\text{Full}}(A,P,N) = \mathcal{L}_1 + \mathcal{L}_2$$

$$\mathcal{J} = \sum_{i=1}^{m} \mathcal{L}_{\text{Full}}(A^{(i)}, P^{(i)}, N^{(i)})$$

Classification vs One Shot Learning



One-shot learning makes use of Siamese networks.

One Shot Learning

No need for retraining !

Learn a similarity score!

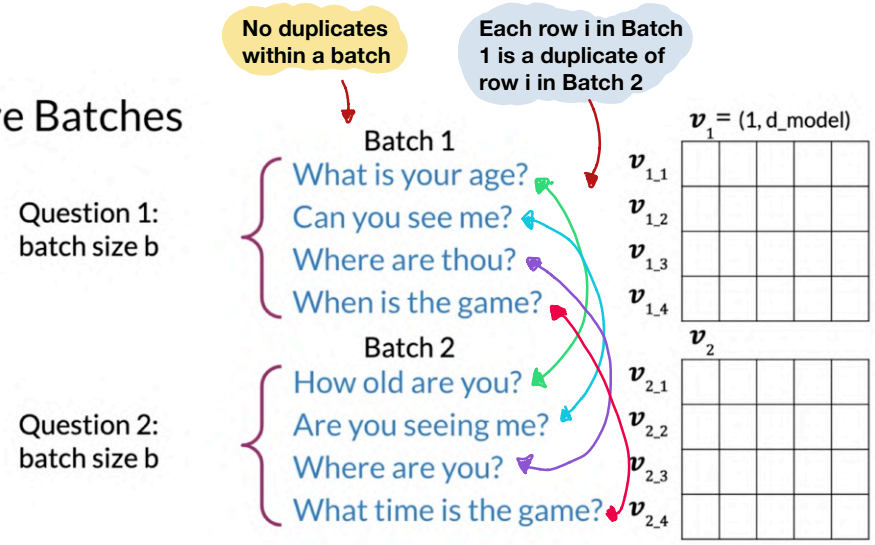
$s(sig1, sig2) > \tau$ ✅

$s(sig1, sig2) \leq \tau$

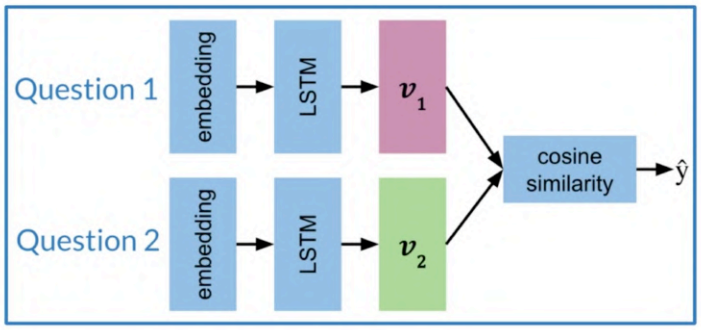
Dataset

Question 1	Question 2	is_duplicate
What is your age?	How old are you?	true
Where are you from?	Where are you going?	false
⋮	⋮	⋮

Prepare Batches



Siamese Model



Create a subnetwork:

1) Embedding
2) LSTM
3) Vectors
4) Cosine Similarity

Testing

1. Convert each input into an array of numbers
2. Feed arrays into your model
3. Compare v_1, v_2 using cosine similarity
4. Test against a threshold τ