Document and Sequence Classification

Sentiment (Refresher)

“I bought an iPhone a few days ago. It was such a nice phone. The touch screen was really cool. The voice quality was clear too. Although the battery life was not long, that is ok for me. ..." (Liu, 2010)

I like the movie.

👍

The movie is like terrible.

👎

Text Classification (Refresher)

The Buccaneers win it!

President Biden vetoed bill

Twitter to be acquired by Apple

→

Let’s start with a sentiment example

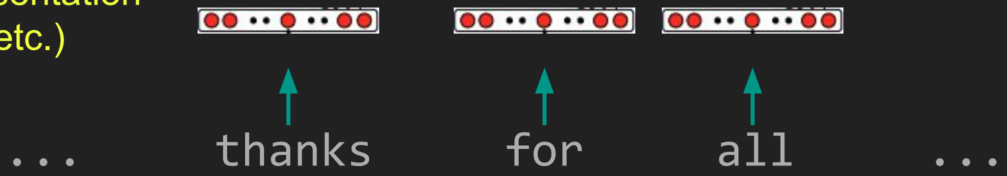
```
So long, and thanks for all the fish!
                       -The Dolphins
```

Representing sequences - Tokens

```
<s> so long <,> and thanks for all the fish <!> </s>
```

Representing sequences - Embeddings

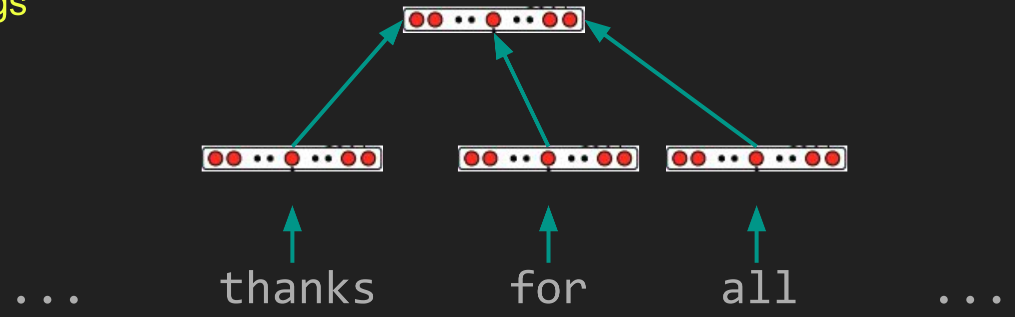
Word embedding representation
(one-hot, PCA, ELMO, etc.)



Representing sequences - Average all embeddings

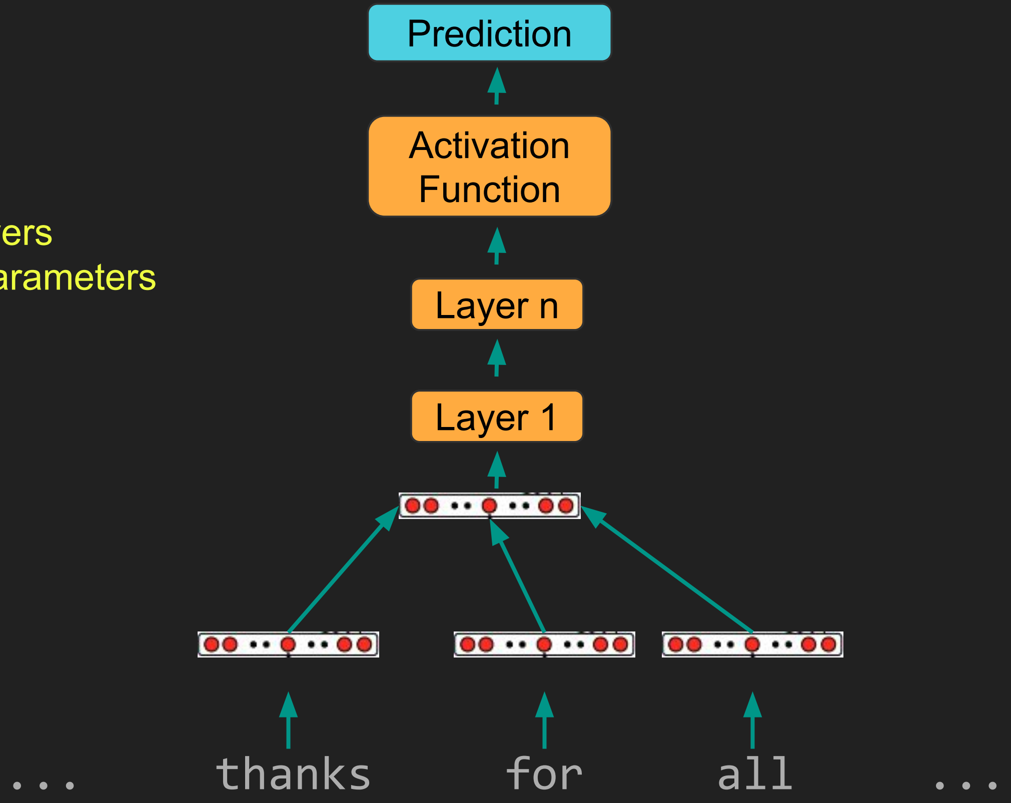
A naive detour

Average the embeddings elementwise

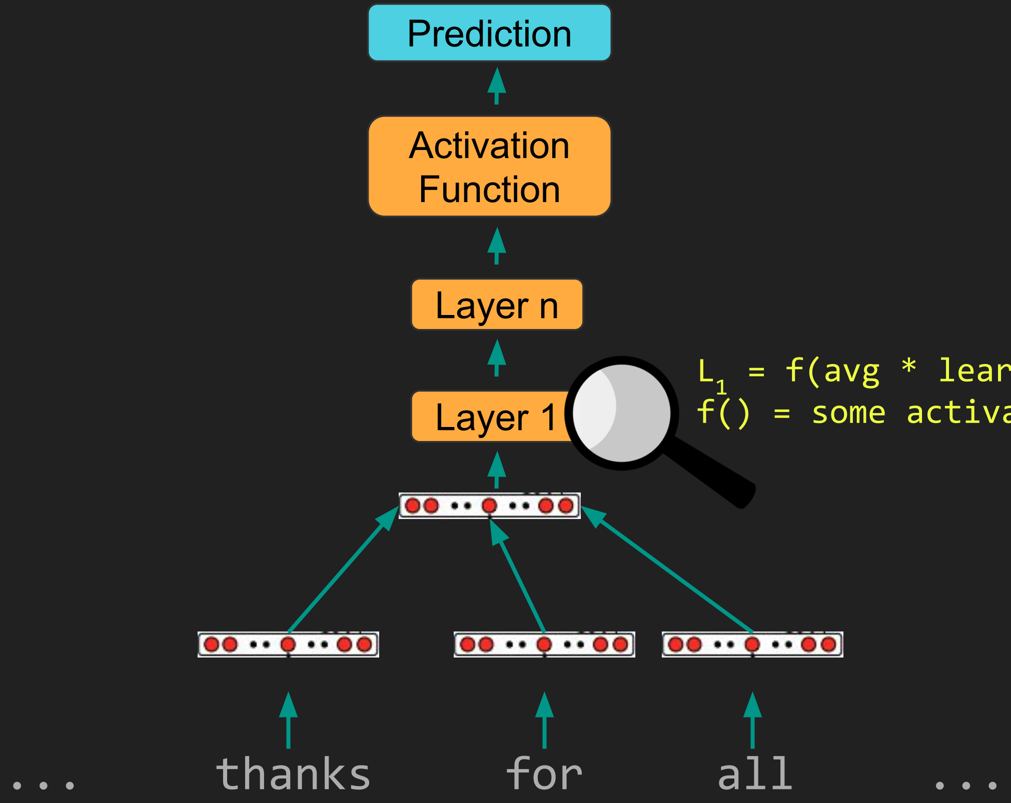


Representing sequences - Average all embeddings

Pass through neural layers
with learnable weight parameters
for each layer

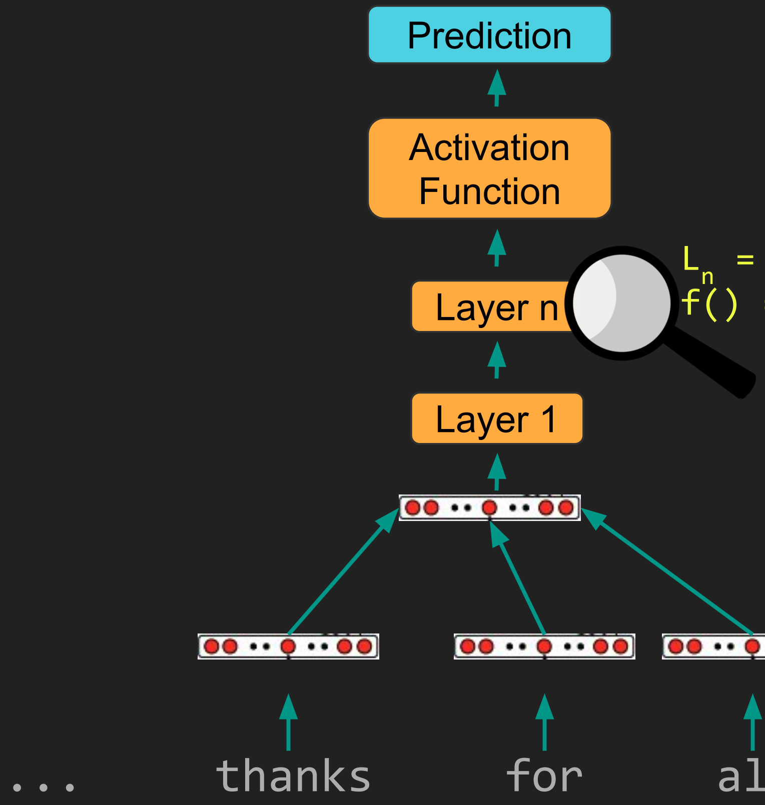


Representing sequences - Average all embeddings



L_1 = f(avg * learned_matrix$_1$ + (bias$_1$))
f() = some activation function

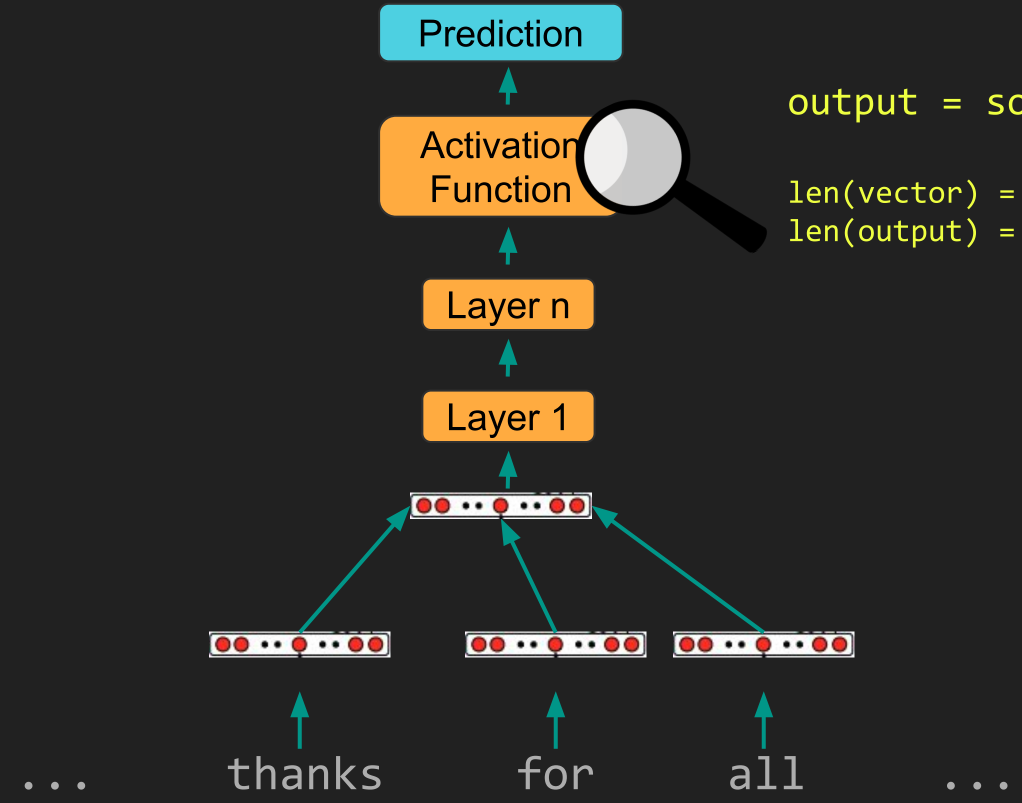
Representing sequences - Average all embeddings



$L_n = f(L_{n-1} * \text{learned\_matrix}_n + \text{bias}_n)$

$f()$ = some activation function

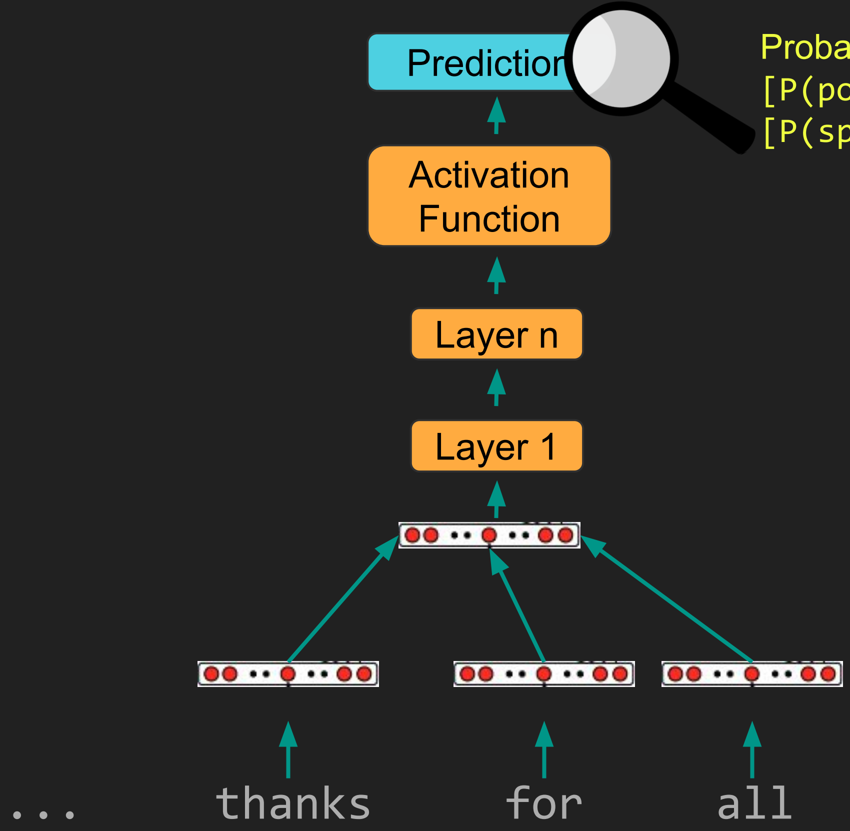
Representing sequences - Average all embeddings



```
output = softmax(vector)

len(vector) = number of classes
len(output) = number of classes
```

Representing sequences - Average all embeddings



Probability for each class
`[P(positive),P(negative)]`
`[P(sports),P(tech),P(politics),...]`

Limitations of Averaging All Embeddings

Hint: *What did we not encode?*

The order of the words in the sequence!

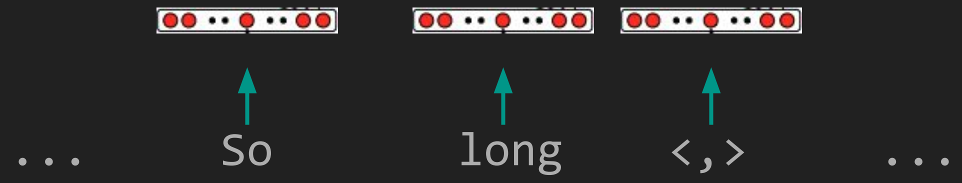
We want to understand text as a sequence

Our models should likewise represent text sequentially

Adding context is the motivation behind using RNNs

By using RNNs we can use information from the representation of previous words to improve the representation of the current word

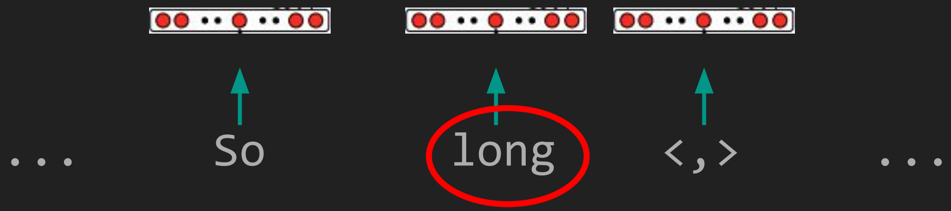
Representing sequences - RNN Layer



Does order matter?

Representing sequences - RNN Layer

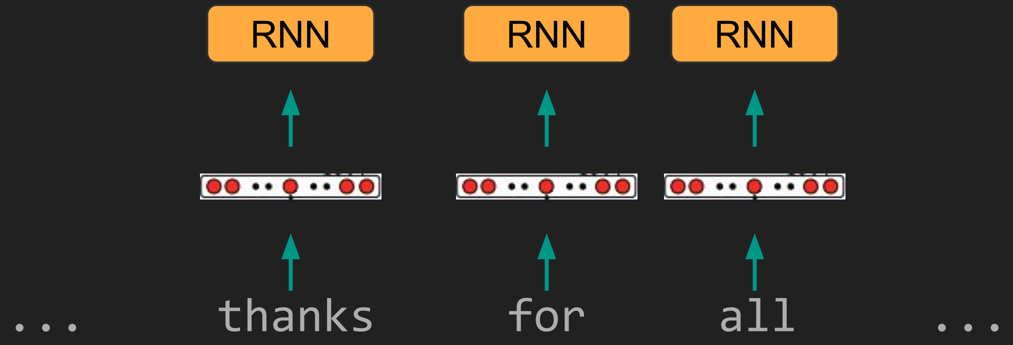
Yes!



... So long <,> ...

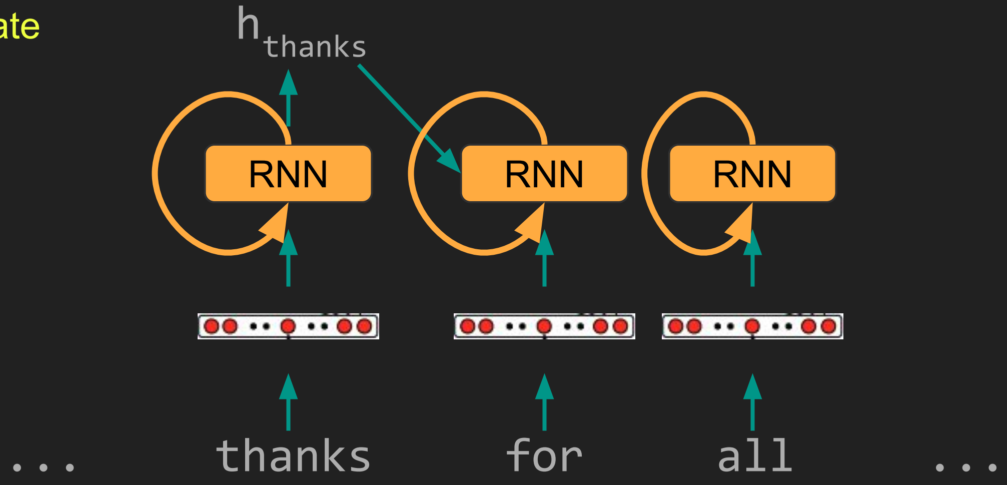
Representing sequences - RNN Layer

Pass through an RNN



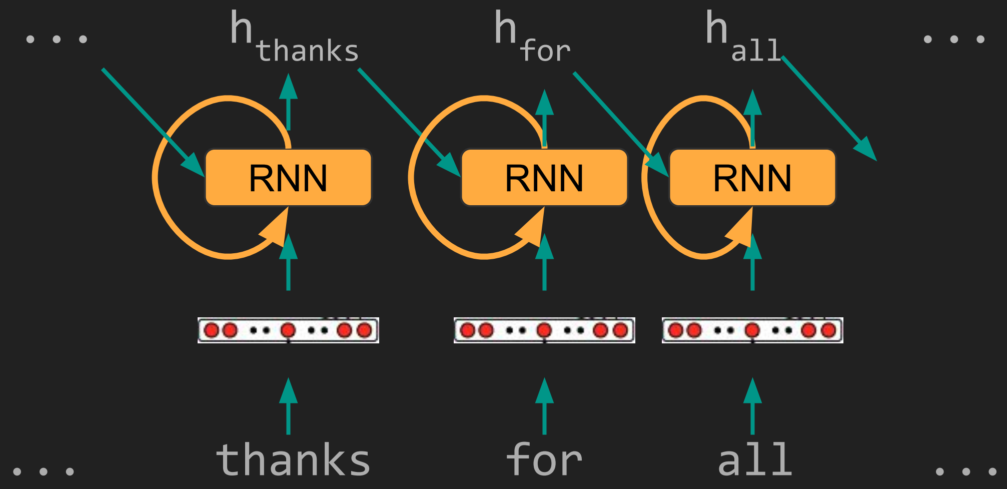
Representing sequences - Hidden State

Calculate the hidden state



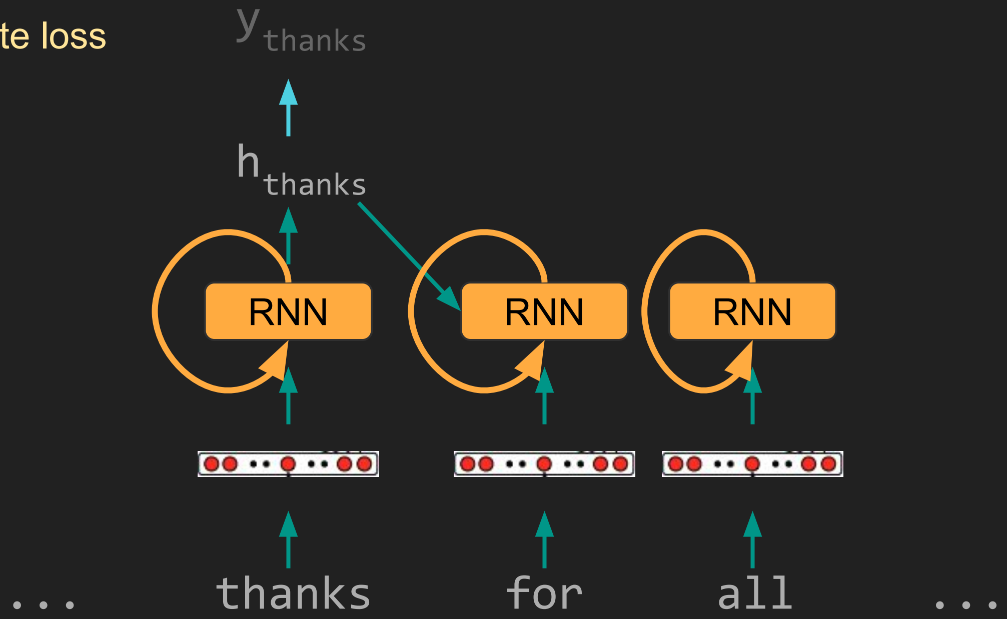
Representing sequences - Hidden States

Repeat



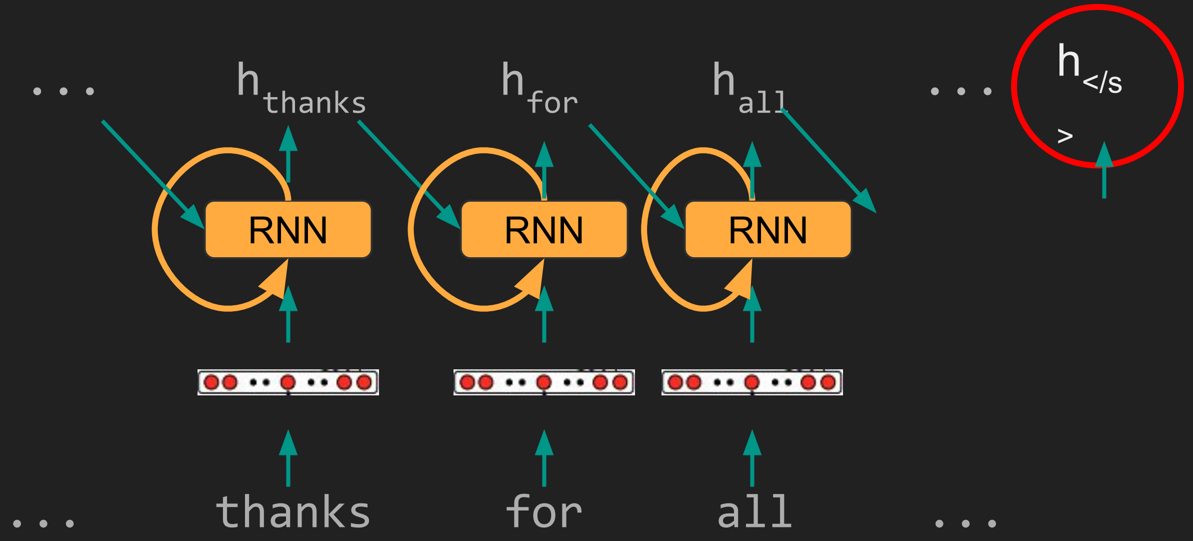
Representing sequences - Outputs

(When training) Calculate loss



Representing sequences in an RNN - Final state

A representation of the entire sequence in order

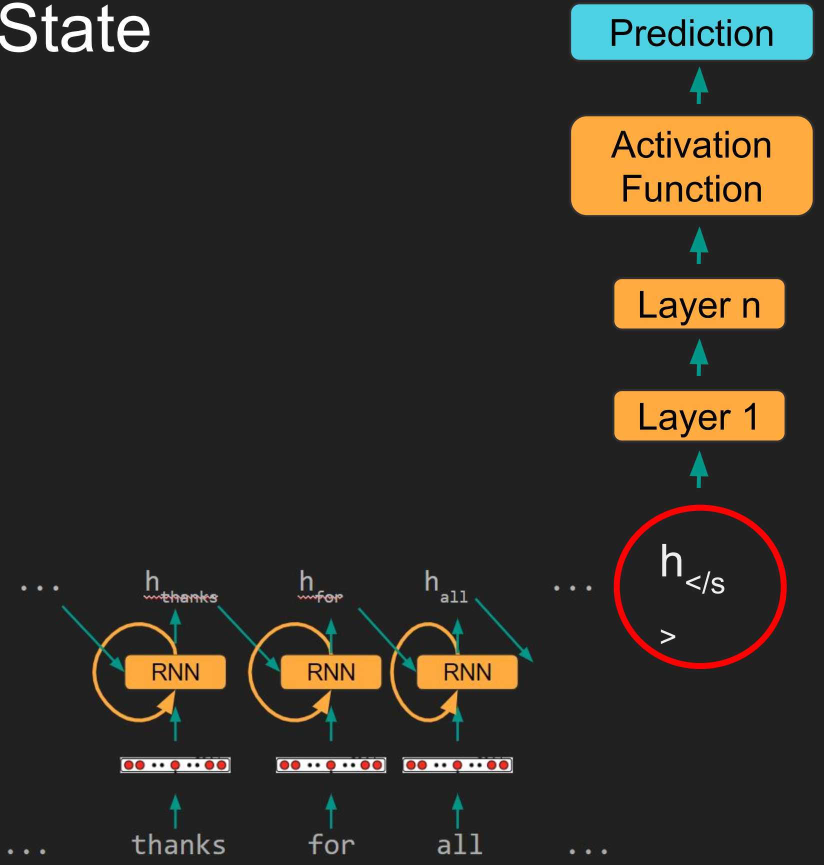


Classification Using the Hidden State

Our final state is a fixed length vector representation

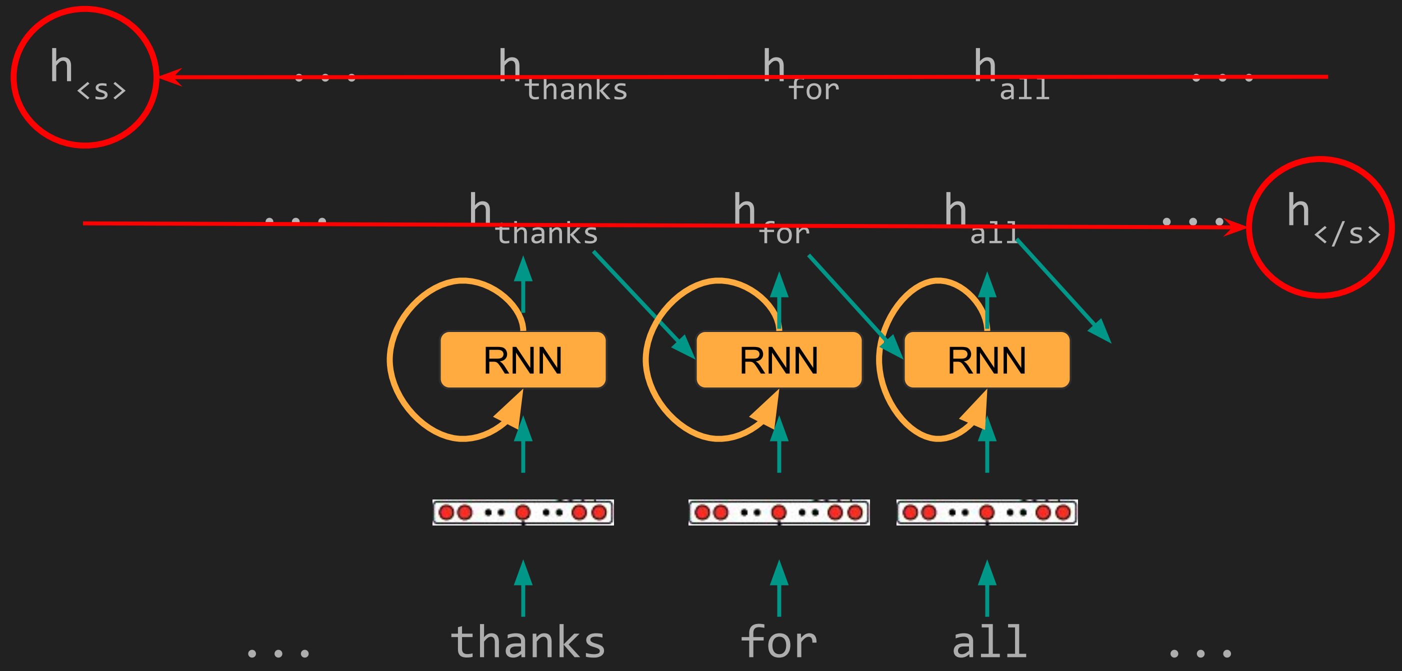
We can pass the final state we just calculated through a neural network classifier

Same as before, each neural layer is composed of transformations of the data using learned parameters

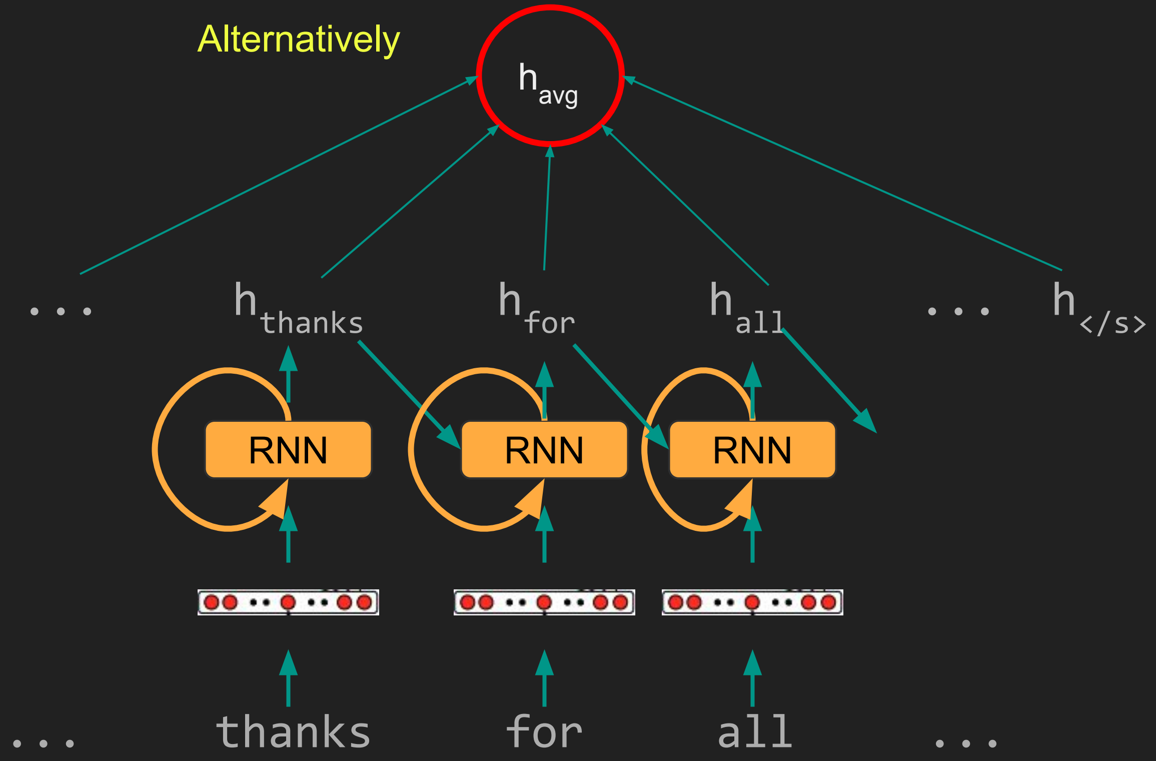


Representing sequences - Bidirectional

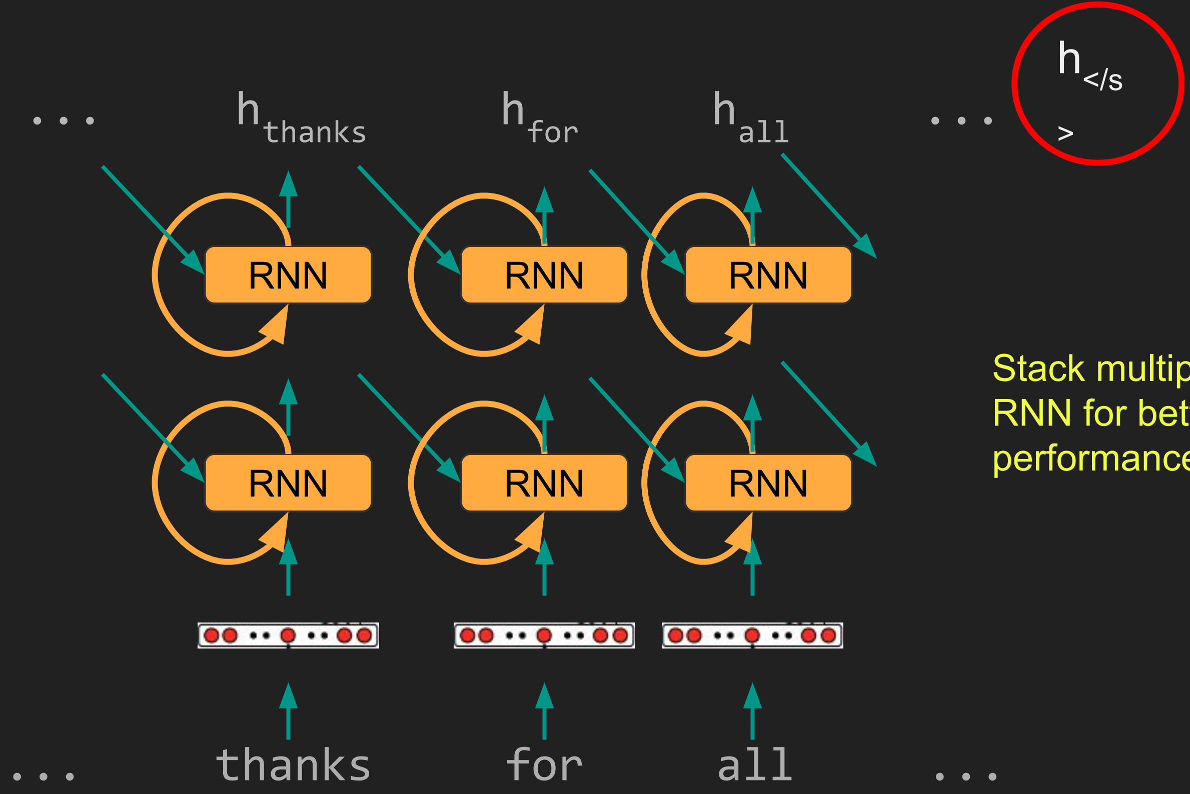
Now go backwards then concat($h_{<s>}$, $h_{</s>}$)



Representing sequences - Average all states



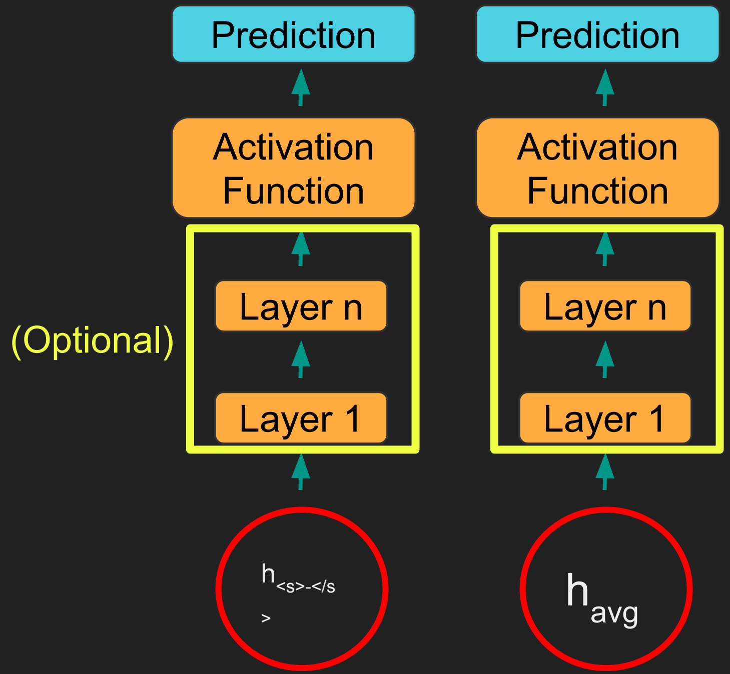
Representing sequences - Stacked RNN



Stack multiple layers of RNN for better performance

Classification Using the Hidden State

Use the same procedure as before!



Limitations

The horse which was raced past the bar tripped.

We will STILL struggle to connect distant words

Hint: *Time complexity*

Slow to run since this process is sequential

Limitations - Fixes

The horse which was raced past the bar tripped.

We will STILL struggle to connect distant words

We can mitigate this with LSTMs and GRU units instead of vanilla RNNs

Hint: *Time complexity*

Slow to run since this process is sequential

NLP recently adopted a much more efficient network - Transformers

Named Entities

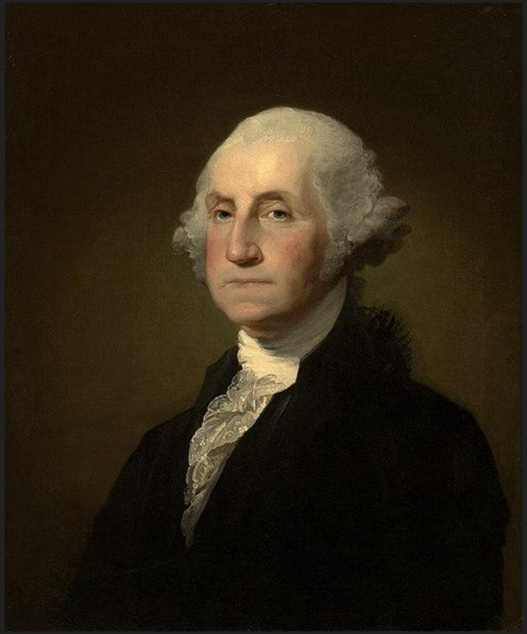
The Task - Named Entity Recognition

Washington

NNP (Proper noun, singular)

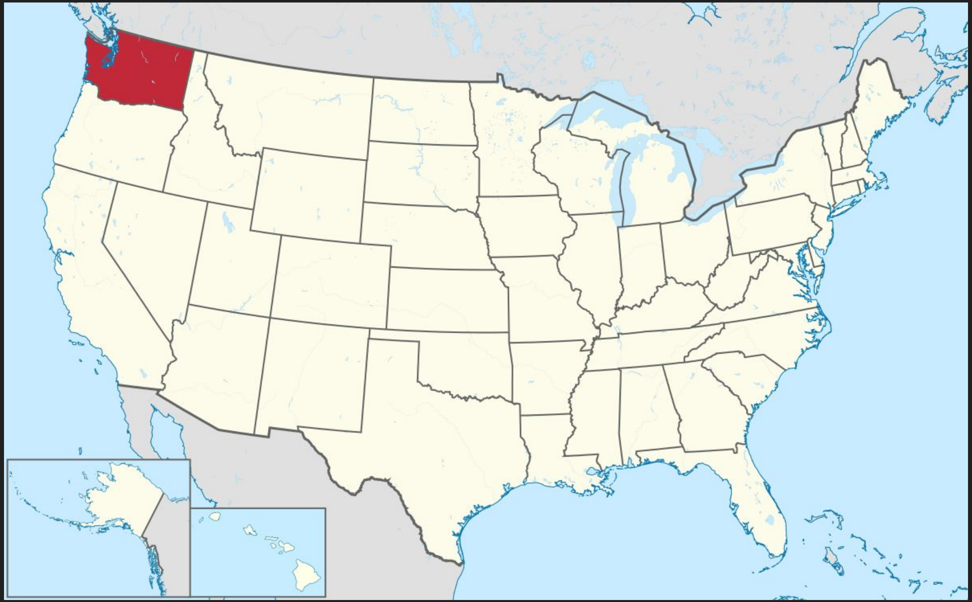
Person

Washington?



Place

Washington?



Organization

Washington?

NER with RNNs

When RNNs were first applied to NER they gave state of art results

This technique is called Sequence Labeling

- Assign each element of a sequence with a label from a fixed set of labels

Examples of Named Entities

Common:

- PER (person) - *Professor Schwartz*
- LOC (location) - *New York City*
- ORG (organization) - *Stony Brook University*
- GPE (geo-political entity) - *United States of America*

Rare:

- TIME (temporal objects) - *Thursday*
- MONEY (prices) - *$600*
- ...

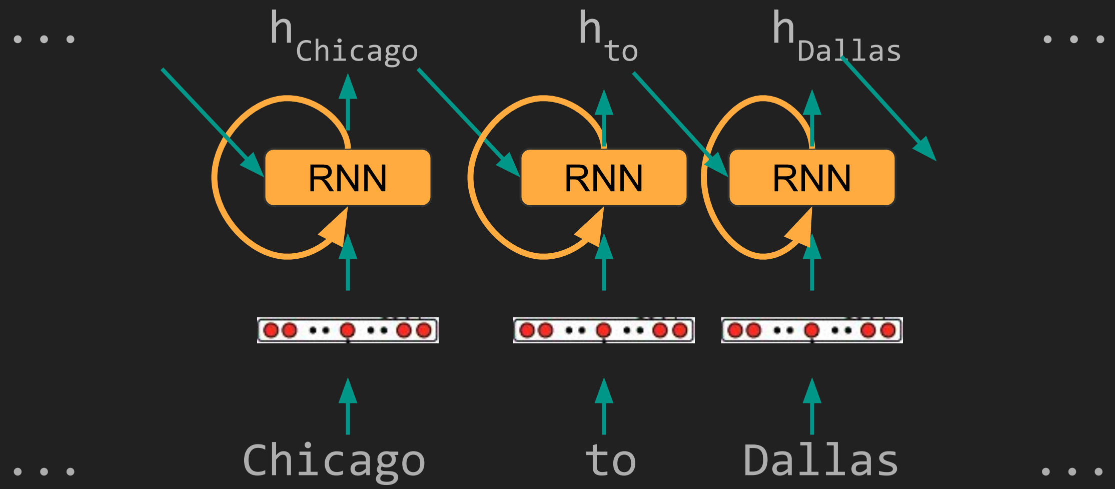
Pieces of NER

(Find spans) Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers, such as Chicago to Dallas and Denver to San Francisco.

(Tag spans) Citing high fuel prices, [ORG United Airlines] said [TIME Friday] it has increased fares by [MONEY $6] per round trip on flights to some cities also served by lower-cost carriers, such as [LOC Chicago] to [LOC Dallas] and [LOC Denver] to [LOC San Francisco].

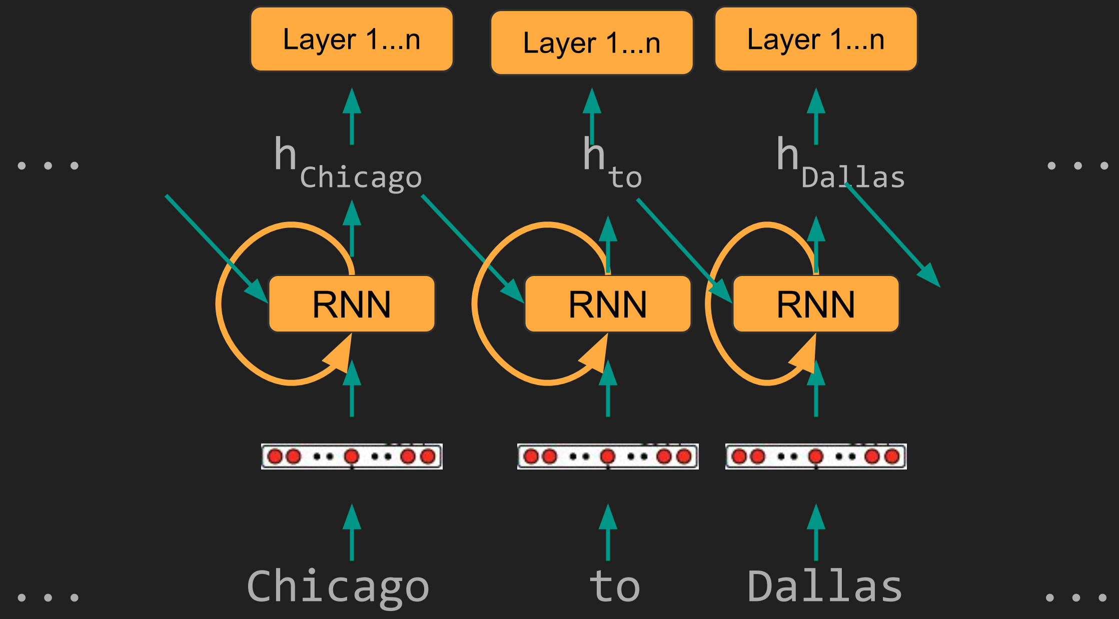
Sequence Labeling

Generate hidden states



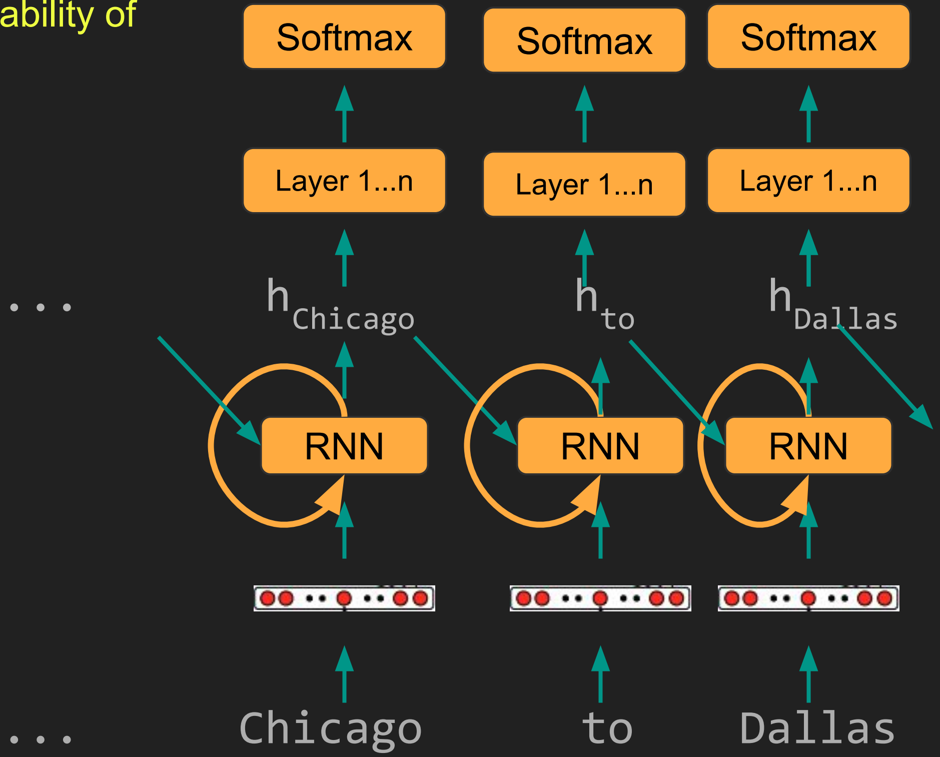
Sequence Labeling

Pass hidden states through linear layer to make them into vectors with the length of the number of classes



Sequence Labeling

Find the probability of each class



Softmax

Softmax

Softmax

Layer 1...n

Layer 1...n

Layer 1...n

...

$h_{Chicago}$

h_{to}

h_{Dallas}

...

RNN

RNN

RNN

...

Chicago

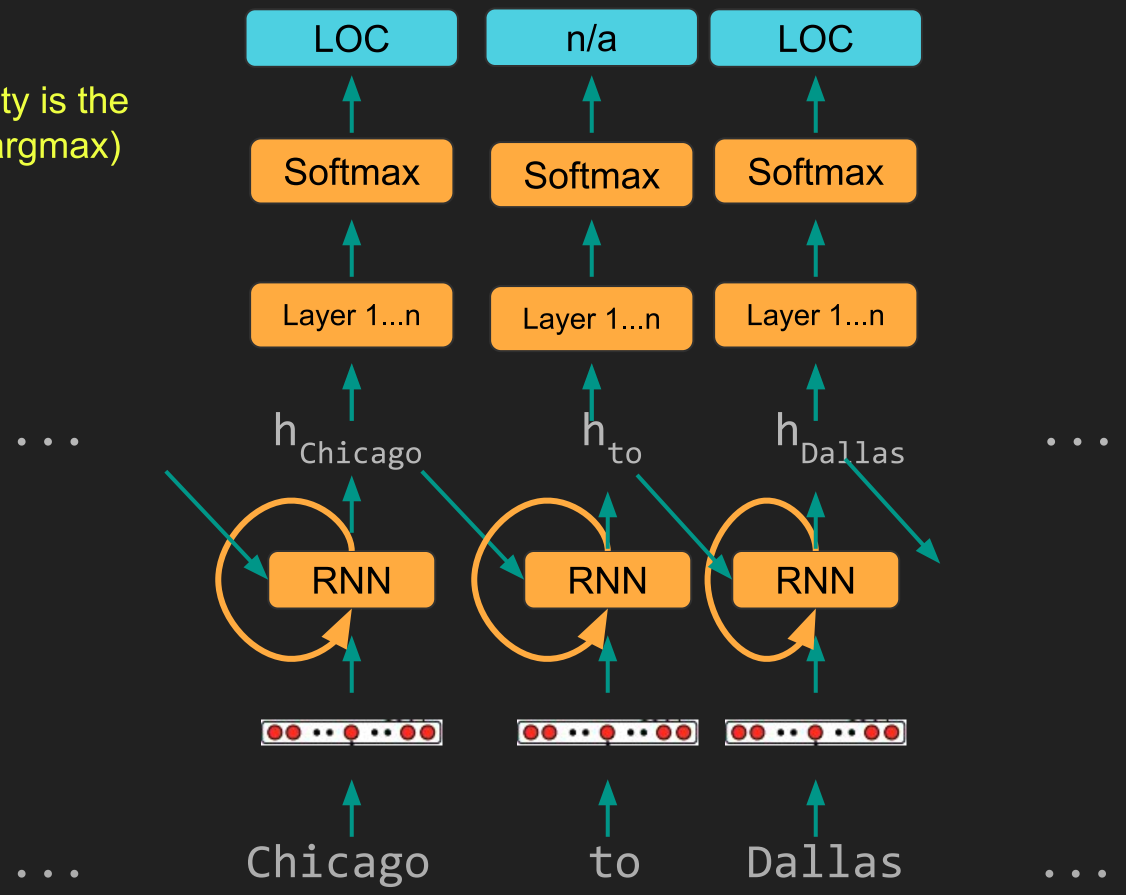
to

Dallas

...

Sequence Labeling

Highest probability is the assigned label (argmax)



Sequence Labeling - PyTorch Model

```
import torch.nn as nn

class NER_RNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, lstm_hidden_dim, number_of_tags):
        super(NER_RNN, self).__init__()

    # Map each token in the vocab to a vector of length embedding_dim
    self.embedding = nn.Embedding(vocab_size, embedding_dim)

    # Get LSTM hidden states from embeddings
    self.lstm = nn.LSTM(embedding_dim, lstm_hidden_dim)

    # Transforms the embedding to a vector of length number_of_tags
    self.classifier = nn.Linear(lstm_hidden_dim, number_of_tags)
```

Sequence Labeling - PyTorch Forward Pass

```python
import torch.nn.functional as F

def forward(self, s):

    # Word embeddings from tokens
    s = self.embedding(s)

    # LSTM embedded sequence
    s, _ = self.lstm(s)

    # Reshaped data, such that there is one token per row
    s = s.view(-1, s.shape[2])

    # Obtain scores for each possible named entity
    s = self.classifier(s)

    # Change scores to probabilities
    return F.log_softmax(s, dim=1)
```

Named Entity Recognition (NER) Uses

Named Entity tags are important for:

- Question answering
    - Responding to questions with relevant information
- Stance detection
    - Understanding the subject of an argument
- Information extraction
    - Extracting the correct sense of a word

Can we improve even further?

There must be a better way

- We would like to encode context information without having to move through the sequence token by token