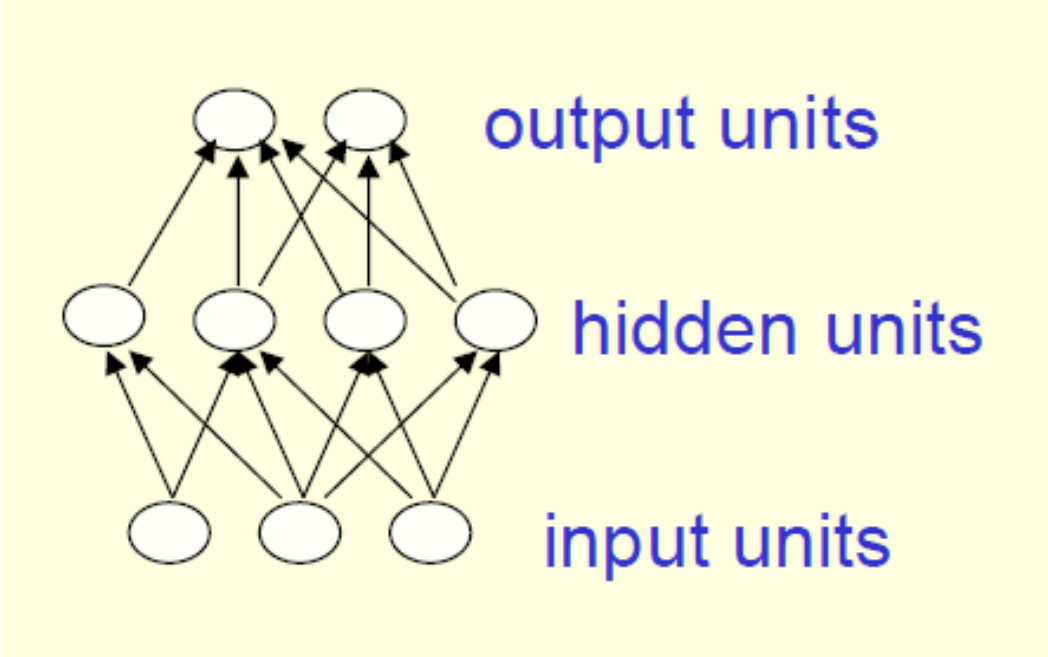
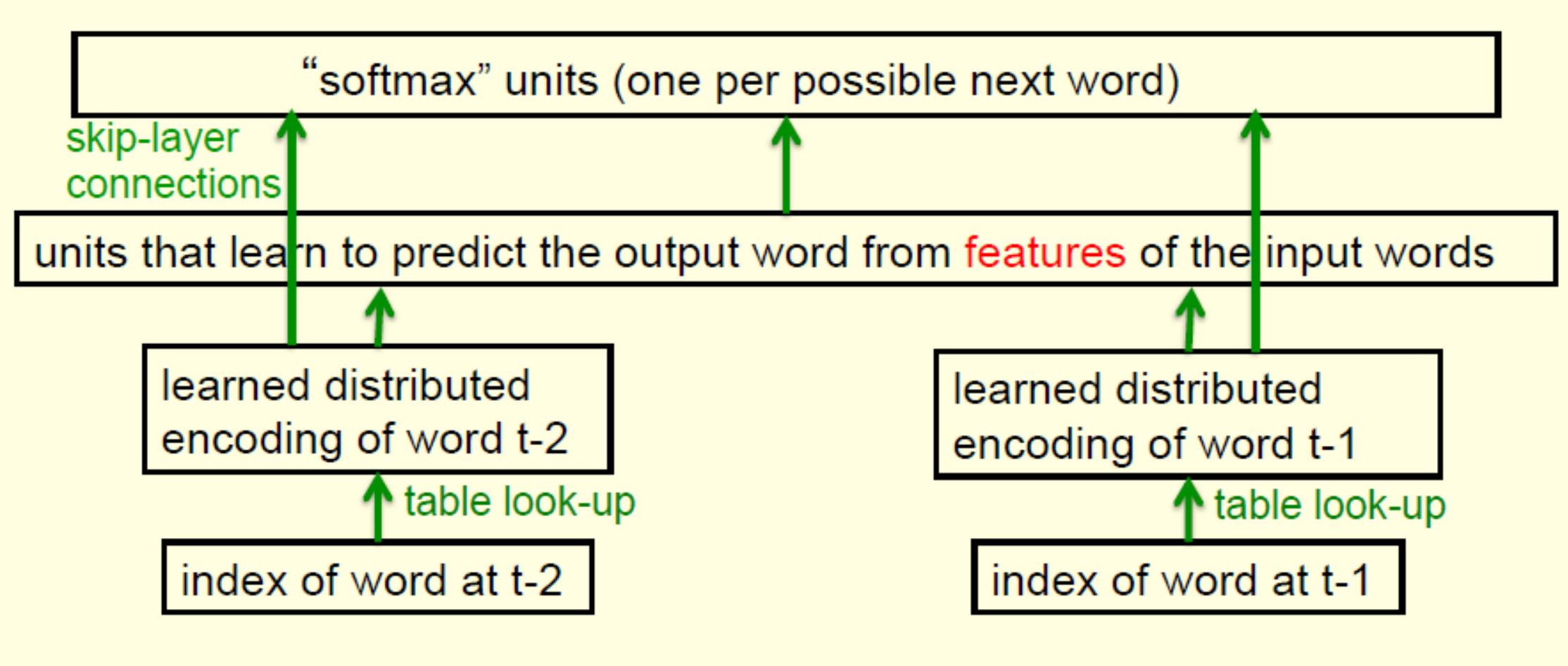
Neural LM

Feed-forward neural network

- This is the simplest type of NN:
  - The first layer is the input and the last layer is the output.
  - If there is more than one hidden layer, we call them deep NN.
- Training: learn the weights on the arcs, using backpropagation.



Bengio's neural LM: "trigram" model

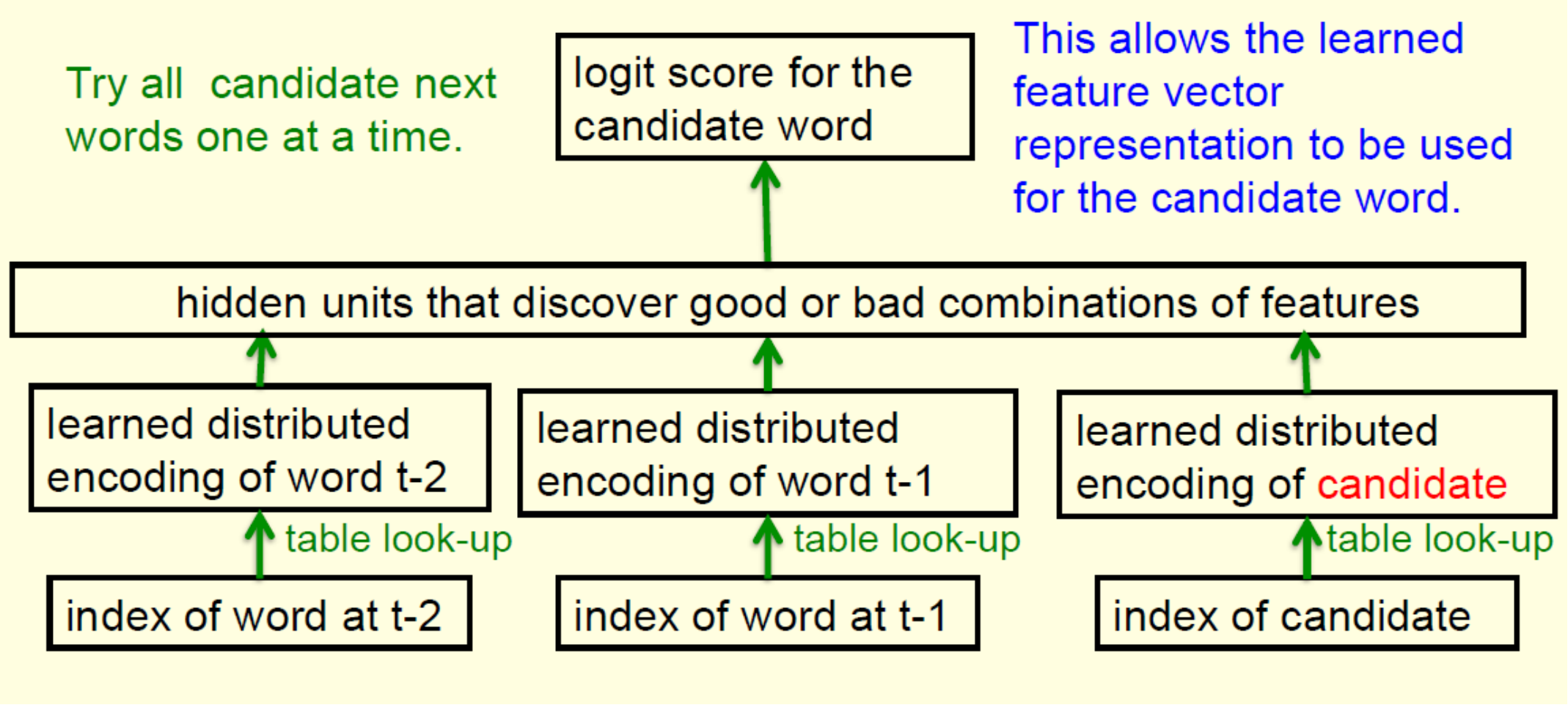


A problem with having 100K output words

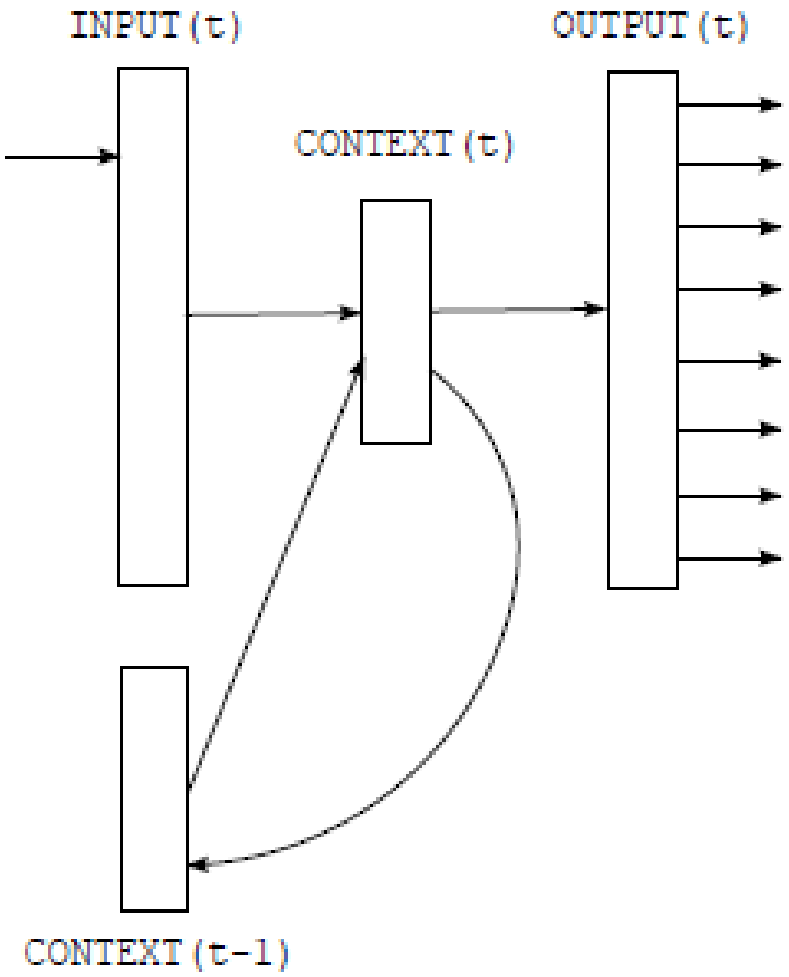
- Each unit in the last hidden layer has 100K outgoing weights:
  - If the number of units in the hidden layer is big, we need a huge number of training instances.
  - If the number of units in the hidden layer is small, it's hard to get the 100K probabilities right.

- Is there a better way to deal with a large number of outputs?

A serial architecture



Recurrent Neural Network LM (Mikolov et al., 2010)

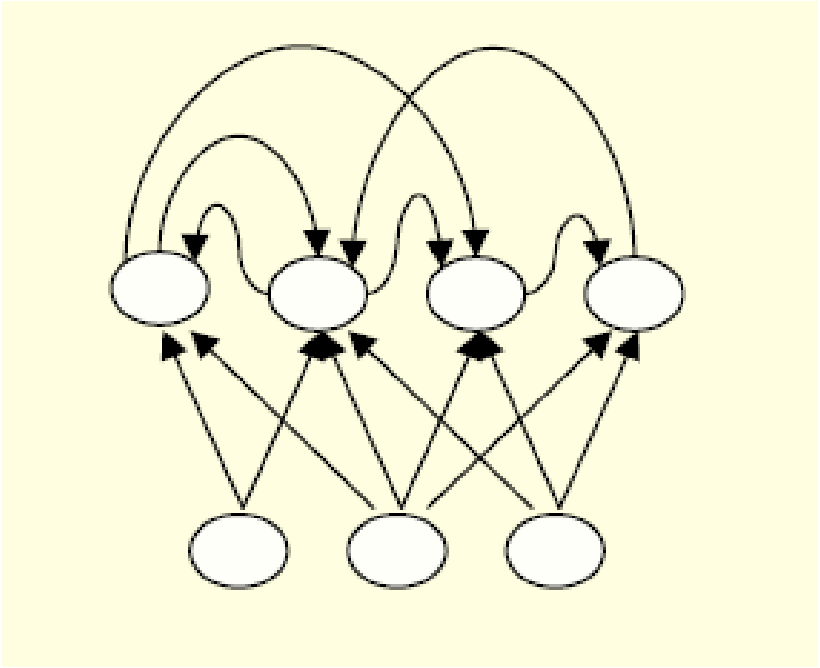


Input layer is w(t) and s(t-1)
Hidden layer represents context s(t)
Output layer is the prob distribution of w(t+1)

Training can be slow.

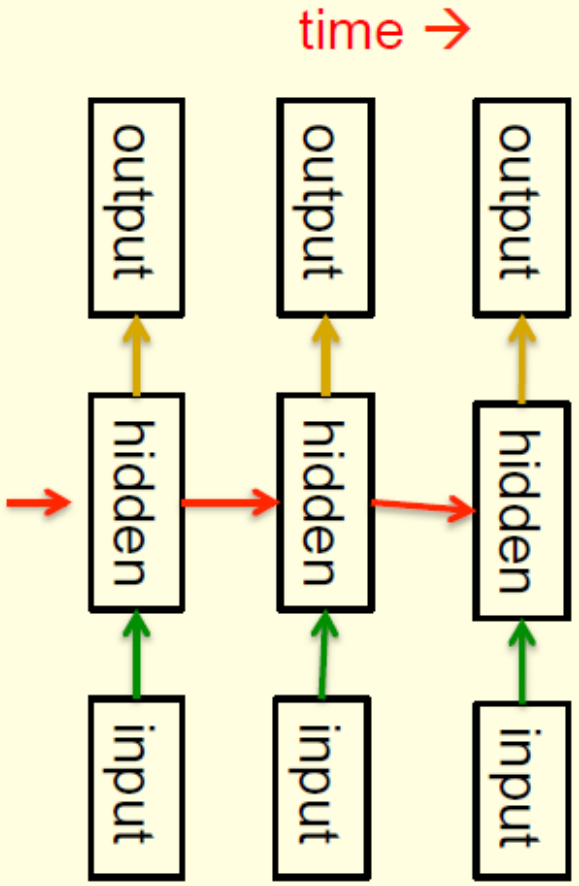
Recurrent neural network (RNN)

- There are directed cycles in the network.
- The training becomes more difficult.
- One of the most commonly used NN types in the NLP field right now.



RNNs for modeling sequences

- Recurrent neural networks are a very natural way to model sequential data:
  - They are equivalent to very deep nets with one hidden layer per time slice.
  - Except that they use the same weights at every time slice and they get input at every time slice.
- They have the ability to remember information in their hidden state for a long time.
  - But its very hard to train them to use this potential.



Summary

- Compared with n-gram LM, Neural LMs can use bigger context and word embedding.
- There are many ways of building neural LMs: e.g., feed-forward, RNN, transformer-based.
- Using large pre-trained LMs has become the norm in NLP:
  - Building a large LM takes a tremendous amount of resource.
  - Common ways of using LMs: fine-tuning, zero-shot, one-shot, few-shot, etc.