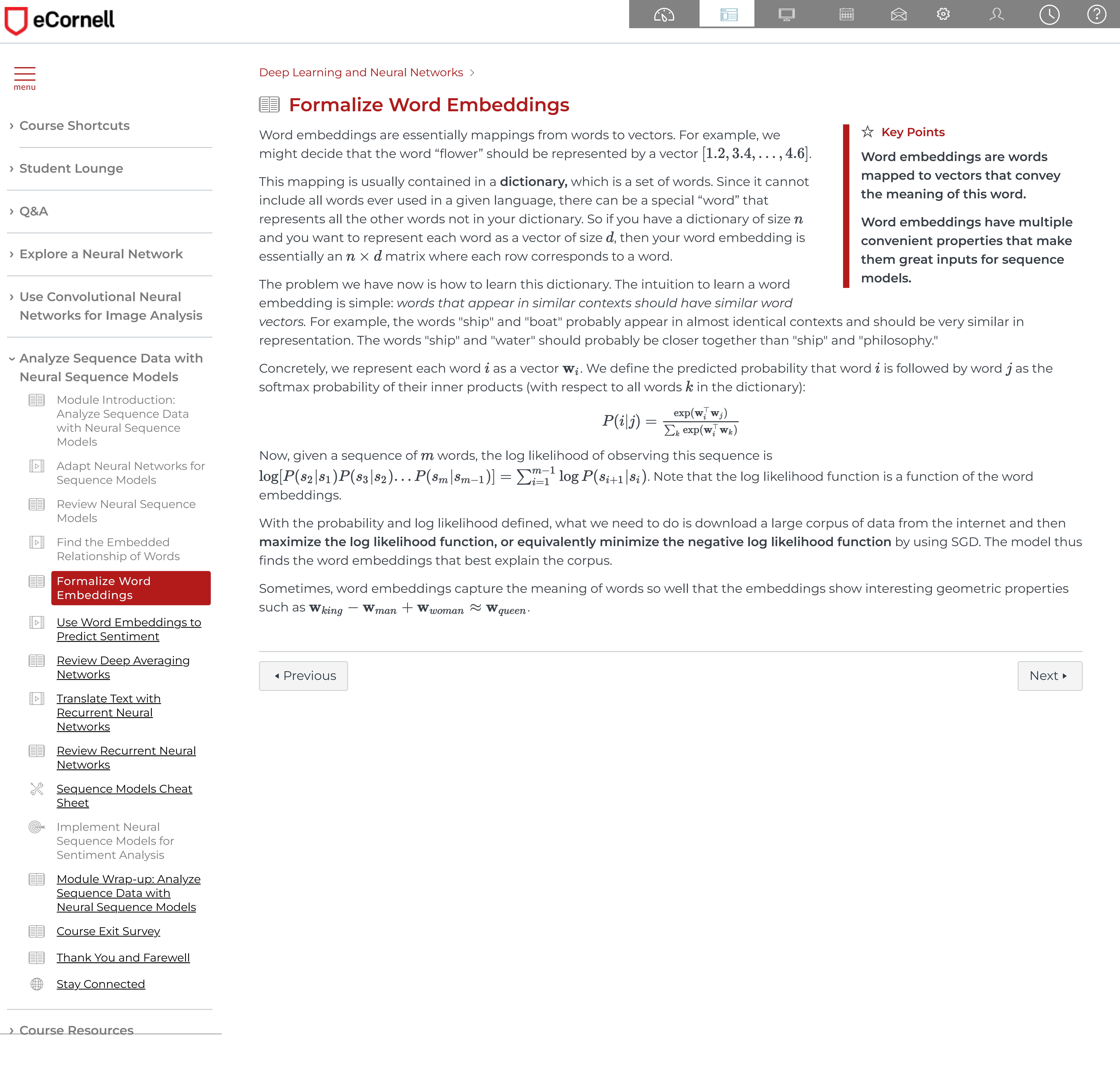

› Course Shortcuts

› Student Lounge

› Q&A

› Explore a Neural Network

› Use Convolutional Neural Networks for Image Analysis

⌄ Analyze Sequence Data with Neural Sequence Models

Module Introduction: Analyze Sequence Data with Neural Sequence Models

Adapt Neural Networks for Sequence Models

Review Neural Sequence Models

Find the Embedded Relationship of Words

Formalize Word Embeddings

Use Word Embeddings to Predict Sentiment

Review Deep Averaging Networks

Translate Text with Recurrent Neural Networks

Review Recurrent Neural Networks

Sequence Models Cheat Sheet

Implement Neural Sequence Models for Sentiment Analysis

Module Wrap-up: Analyze Sequence Data with Neural Sequence Models

Course Exit Survey

Thank You and Farewell

Stay Connected

› Course Resources

Deep Learning and Neural Networks ›

Formalize Word Embeddings

☆ **Key Points**

Word embeddings are words mapped to vectors that convey the meaning of this word.

Word embeddings have multiple convenient properties that make them great inputs for sequence models.

Word embeddings are essentially mappings from words to vectors. For example, we might decide that the word “flower” should be represented by a vector $[1.2, 3.4, \ldots, 4.6]$.

This mapping is usually contained in a **dictionary,** which is a set of words. Since it cannot include all words ever used in a given language, there can be a special “word” that represents all the other words not in your dictionary. So if you have a dictionary of size n and you want to represent each word as a vector of size d, then your word embedding is essentially an $n \times d$ matrix where each row corresponds to a word.

The problem we have now is how to learn this dictionary. The intuition to learn a word embedding is simple: *words that appear in similar contexts should have similar word vectors*. For example, the words "ship" and "boat" probably appear in almost identical contexts and should be very similar in representation. The words "ship" and "water" should probably be closer together than "ship" and "philosophy."

Concretely, we represent each word i as a vector $\mathbf{w}_i$. We define the predicted probability that word i is followed by word j as the softmax probability of their inner products (with respect to all words k in the dictionary):

$$P(i|j) = \frac{\exp(\mathbf{w}_i^\top \mathbf{w}_j)}{\sum_k \exp(\mathbf{w}_i^\top \mathbf{w}_k)}$$

Now, given a sequence of m words, the log likelihood of observing this sequence is $\log[P(s_2|s_1)P(s_3|s_2)\ldots P(s_m|s_{m-1})] = \sum_{i=1}^{m-1} \log P(s_{i+1}|s_i)$. Note that the log likelihood function is a function of the word embeddings.

With the probability and log likelihood defined, what we need to do is download a large corpus of data from the internet and then **maximize the log likelihood function, or equivalently minimize the negative log likelihood function** by using SGD. The model thus finds the word embeddings that best explain the corpus.

Sometimes, word embeddings capture the meaning of words so well that the embeddings show interesting geometric properties such as $\mathbf{w}_{king} - \mathbf{w}_{man} + \mathbf{w}_{woman} \approx \mathbf{w}_{queen}$.

◂ Previous

Next ▸