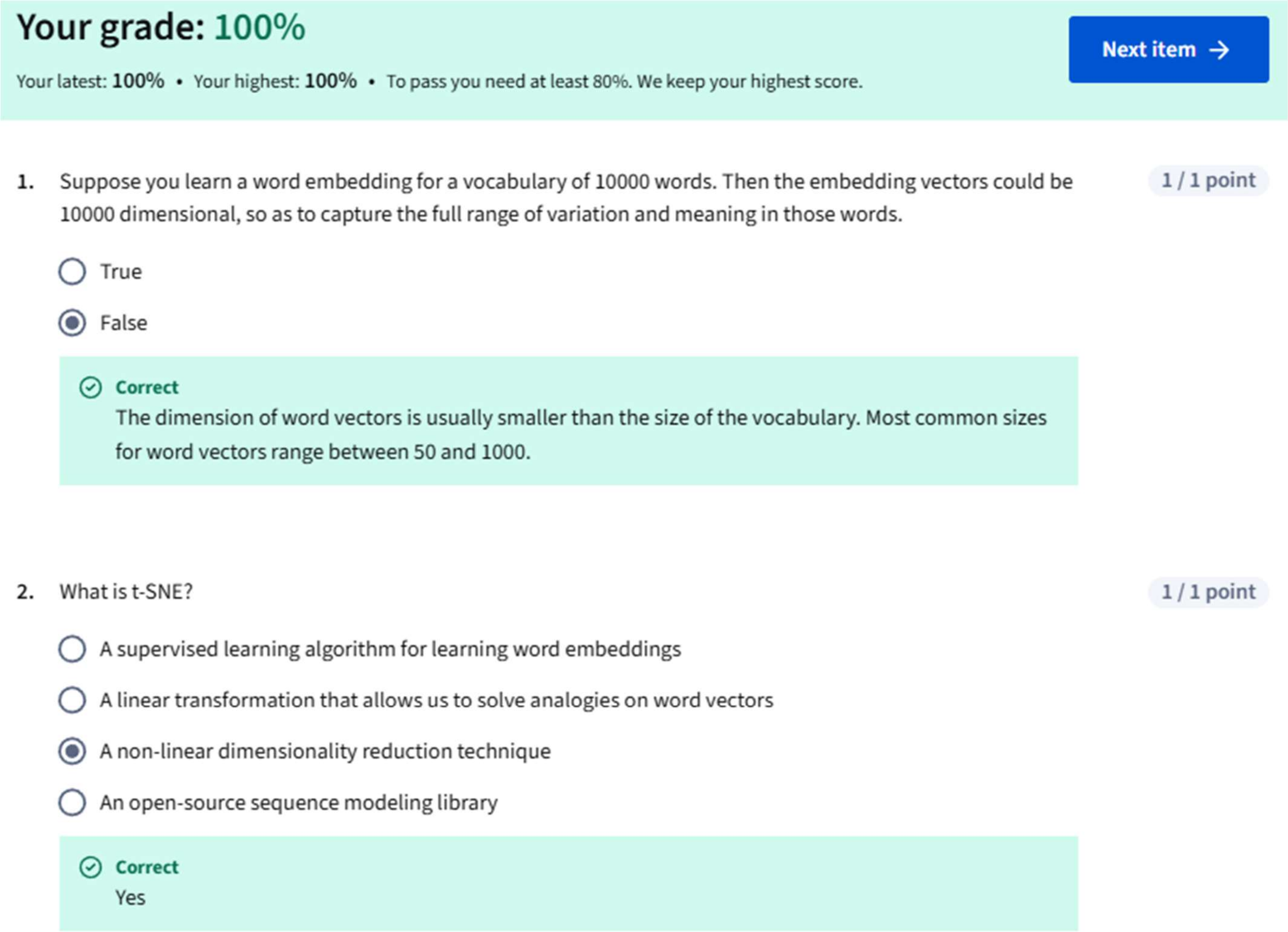


Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 80%. We keep your highest score.

1. Suppose you learn a word embedding for a vocabulary of 10000 words. Then the embedding vectors could be 10000 dimensional, so as to capture the full range of variation and meaning in those words.

1 / 1 point

○ True

● False

✓ **Correct**
The dimension of word vectors is usually smaller than the size of the vocabulary. Most common sizes for word vectors range between 50 and 1000.

2. What is t-SNE?

1 / 1 point

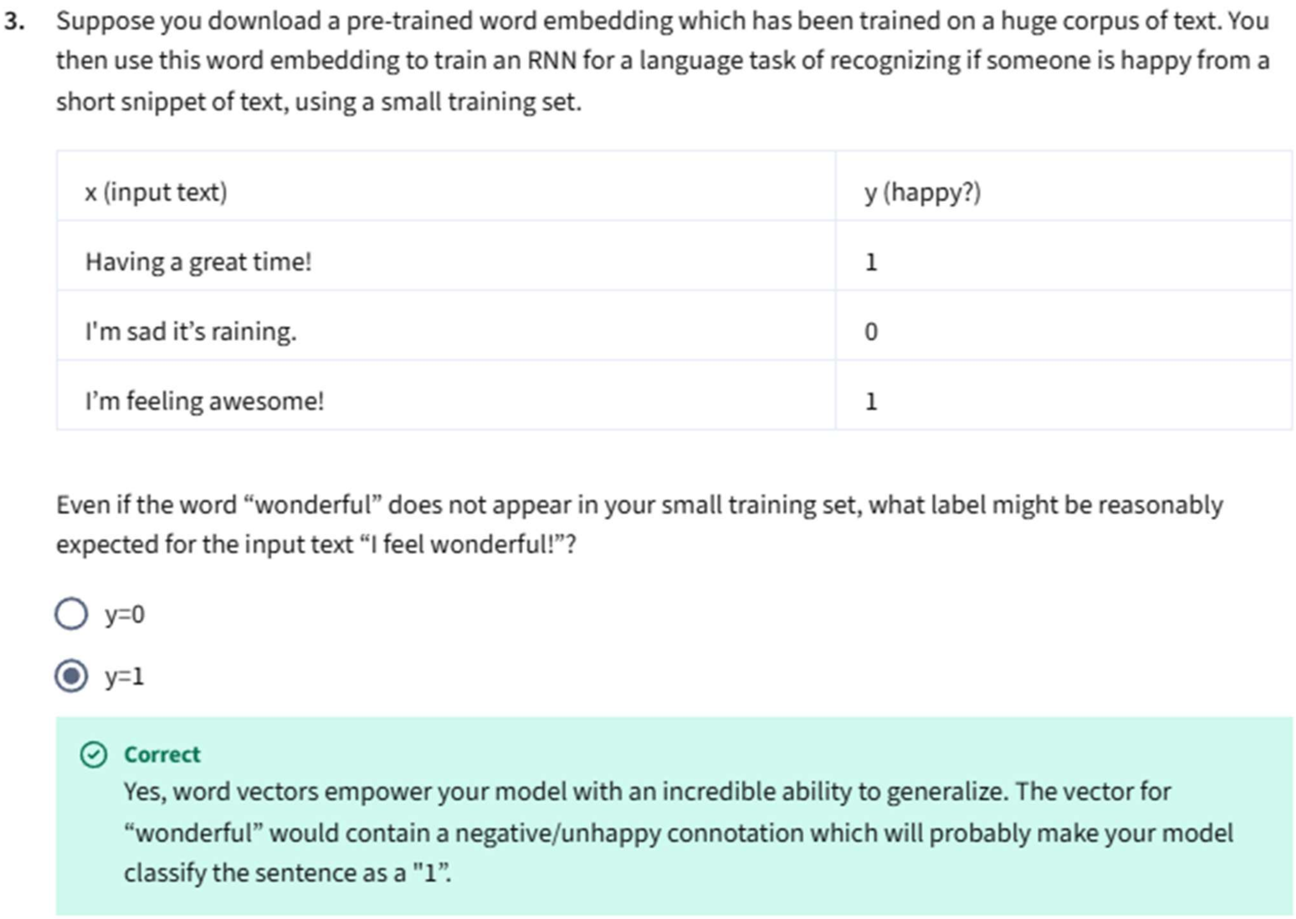
○ A supervised learning algorithm for learning word embeddings

○ A linear transformation that allows us to solve analogies on word vectors

● A non-linear dimensionality reduction technique

○ An open-source sequence modeling library

✓ **Correct**
Yes



3. Suppose you download a pre-trained word embedding which has been trained on a huge corpus of text. You then use this word embedding to train an RNN for a language task of recognizing if someone is happy from a short snippet of text, using a small training set.

x (input text)	y (happy?)
Having a great time!	1
I'm sad it’s raining.	0
I’m feeling awesome!	1

Even if the word “wonderful” does not appear in your small training set, what label might be reasonably expected for the input text “I feel wonderful!”?

○ y=0

● y=1

✓ **Correct**
Yes, word vectors empower your model with an incredible ability to generalize. The vector for “wonderful” would contain a negative/unhappy connotation which will probably make your model classify the sentence as a "1”.

4. Which of these equations do you think should hold for a good word embedding? (Check all that apply)

☑ $e_{boy} - e_{girl} \approx e_{brother} - e_{sister}$

Correct
Yes!

☑ $e_{boy} - e_{brother} \approx e_{girl} - e_{sister}$

Correct
Yes!

□ $e_{boy} - e_{girl} \approx e_{sister} - e_{brother}$

□ $e_{boy} - e_{brother} \approx e_{sister} - e_{girl}$

5. Let A be an embedding matrix, and let o_{4567} be a one-hot vector corresponding to word 4567. Then to get the embedding of word 4567, why don't we call $A * o_{4567}$ in Python?

● It is computationally wasteful.

○ This doesn't handle unknown words (<UNK>).

○ The correct formula is $A^T * o_{4567}$.

○ None of the answers are correct: calling the Python snippet as described above is fine.

Correct
Yes, the element-wise multiplication will be extremely inefficient.

6. When learning word embeddings, we pick a given word and try to predict its surrounding words or vice versa.

● True

○ False

Correct
Word embeddings are learned by picking a given word and trying to predict its surrounding words or vice versa.

7. True/False: In the word2vec algorithm, you estimate *P(t | c)*, where *t* is the target word and *c* is a context word. *t* and *c* are chosen from the training set to be nearby words.

○ False

● True

Correct
Yes, *t* and *c* are chosen from the training set to be nearby words.

8. Suppose you have a 10000 word vocabulary, and are learning 500-dimensional word embeddings. The word2vec model uses the following softmax function:

$$P(t \mid c) = \frac{e^{\theta_t^T e_c}}{\sum_{t'=1}^{10000} e^{\theta_{t'}^T e_c}}$$

Which of these statements are correct? Check all that apply.

☐ θ_t and e_c are both 10000 dimensional vectors.

☑ θ_t and e_c are both trained with an optimization algorithm such as Adam or gradient descent.

Correct

☑ θ_t and e_c are both 500 dimensional vectors.

Correct

☐ After training, we should expect θ_t to be very close to e_c when t and c are the same word.

9. Suppose you have a 10000 word vocabulary, and are learning 500-dimensional word embeddings.The GloVe model minimizes this objective:

$$\min \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})(\theta_i^T e_j + b_i + b_j\text{'} - log X_{ij})^2$$

True/False: θ_i and e_j should be initialized to 0 at the beginning of training.

○ True

● False

Correct

θ_i and e_j should be initialized randomly at the beginning of training.

10. You have trained word embeddings using a text dataset of s_1 words. You are considering using these word embeddings for a language task, for which you have a separate labeled dataset of s_2 words. Keeping in mind that using word embeddings is a form of transfer learning, under which of these circumstances would you expect the word embeddings to be helpful?

○ $s_1 << s_2$

● $s_1 >> s_2$

Correct

s_1 should transfer to s_2