True/False: Suppose you learn a word embedding for a vocabulary of 20000 words. Then the embedding vectors could be 1000 dimensional, so as to capture the full range of variation and meaning in those words.

0 / 1 point

- (•) False
- ( ) True

Expand

Incorrect
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. True/False: t-SNE is a linear transformation that allows us to solve analogies on word vectors.

- (•) False
- ( ) True

Expand

Correct
tr-SNE is a non-linear dimensionality reduction technique.

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.

0 / 1 point

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=1
- (•) y=0

Incorrect
No, 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 "0".

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

- [x] $e_{boy} - e_{girl} \approx e_{brother} - e_{sister}$

  ✓ **Correct**
  Yes!

- [ ] $e_{boy} - e_{brother} \approx e_{girl} - e_{sister}$
- [ ] $e_{boy} - e_{brother} \approx e_{sister} - e_{girl}$
- [ ] $e_{boy} - e_{girl} \approx e_{sister} - e_{brother}$

Expand

✗ **Incorrect**
You didn't select all the correct answers

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? 1 / 1

- ( ) This doesn't handle unknown words (<UNK>).
- ( ) None of the answers are correct: calling the Python snippet as described above is fine.
- (•) It is computationally wasteful.
- ( ) The correct formula is $A^T * o_{4567}$

Expand

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

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

- (●) True
- ( ) False

Expand

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

True/False: In the word2vec algorithm, you estimate $P(t \mid c)$, where t is the target word and c is a context word. t and c are chosen from the training set using $c$as the sequence of all the words in the sentence before t. 0 / 1 poi

- (●) True
- ( ) False

Expand

✗ **Incorrect**
No, and are chosen from the training set to be nearby words.

8. Suppose you have a 10000 word vocabulary, and are learning 100-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.

- [ ] After training, we should expect θ_t to be very close to e_c when t and c are the same word.
- [ ] θ_t and e_c are both 10000 dimensional vectors.
- [x] θ_t and e_c are both trained with an optimization algorithm.

✓ **Correct**
To review this concept watch the *Word2Vec* lecture.

- [x] θ_t and e_c are both 100 dimensional vectors.

✓ **Correct**

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

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

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

- [ ] Theoretically, the weighting function $f(.)$ must satisfy $f(0) = 0$
- [x] θ_i and e_j should be initialized to 0 at the beginning of training.

! **This should not be selected**
The variables should not be initialized to 0 at the beginning of training.

- [x] X_{ij} is the number of times word j appears in the context of word i.

✓ **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 t_1 words. You are considering using these word embeddings for a language task, for which you have a separate labeled dataset of t_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?

- ○ When t_1 is equal to t_2
- ◉ When t_1 is smaller than t_2
- ○ When t_1 is larger than t_2

Expand

True/False: Suppose you learn a word embedding for a vocabulary of 20000 words. Then the embedding vectors could be 1000 dimensional, so as to capture the full range of variation and meaning in those words.

1 / 1 point

- ◉ True
- ○ False

Expand

✓ **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?

- (●) A non-linear dimensionality reduction technique
- ( ) A linear transformation that allows us to solve analogies on word vectors
- ( ) An open-source sequence modeling library
- ( ) A supervised learning algorithm for learning word embeddings

Expand

Correct
Yes

5. True/False: The most computationally efficient formula for Python to get the embedding of word 1021, if C is an embedding matrix, and o_{1021} is a one-hot vector corresponding to word 1021, is $C^T * o_{1021}$.

- (●) False
- ( ) True

Expand

Correct
It is computationally wasteful because the element-wise multiplication will be extremely inefficient.

6. When learning word embeddings, we create an artificial task of estimating $P(target \mid context)$. It is okay if we do poorly on this artificial prediction task; the more important by-product of this task is that we learn a useful set of word embeddings. **1 / 1 point**

- (●) True
- ( ) False

Expand

Correct