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 point**

- [ ] True
- [x] False

2. True/False: t-SNE is a linear transformation that allows us to solve analogies on word vectors. **1 point**

- [x] False
- [ ] True

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. **1 point**

x (input text)	y (happy?)
I'm feeling wonderful today!	1
I'm bummed that my cat is ill.	0
Really enjoying this!	1

True/False: Then even if the word "upset" does not appear in your small training set, your RNN might reasonably be expected to recognize "I'm upset" as deserving a label y = 0.

- [ ] False
- [x] True

4. Which of these equations do you think should hold for a good word embedding? (Check all that apply) **1 point**

- [x] $e_{man} - e_{uncle} \approx e_{woman} - e_{aunt}$
- [x] $e_{man} - e_{woman} \approx e_{uncle} - e_{aunt}$
- [ ] $e_{man} - e_{woman} \approx e_{aunt} - e_{uncle}$
- [ ] $e_{man} - e_{aunt} \approx e_{woman} - e_{uncle}$

5. Let E be an embedding matrix, and let o_{1234} be a one-hot vector corresponding to word 1234. Then to get the embedding of word 1234, why don't we call $E * o_{1234}$ in Python? **1 point**

- [ ] None of the above: calling the Python snippet as described above is fine.
- [x] It is computationally wasteful.
- [ ] The correct formula is $E^T * o_{1234}$.
- [ ] This doesn't handle unknown words (<UNK>).

6. When learning word embeddings, we pick a given word and try to predict its surrounding words or vice versa. **1 point**

- [x] True
- [ ] False

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. **1 point**

- [x] True
- [ ] False

8. Suppose you have a 10000 word vocabulary, and are learning 500-dimensional word embeddings. The word2vec model uses the following softmax function: **1 point**

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

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

- [x] θ_t and e_c are both 10000 dimensional vectors.
- [x] θ_t and e_c are both trained with an optimization algorithm such as Adam or gradient descent.
- [ ] 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 500 dimensional vectors.

9. Suppose you have a 10000 word vocabulary, and are learning 500-dimensional word embeddings. The GloVe model minimizes this objective: **1 point**

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

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

- [x] X_{ij} is the number of times word j appears in the context of word i.
- [x] θ_i and e_j should be initialized randomly at the beginning of training.
- [ ] θ_i and e_j should be initialized to 0 at the beginning of training.
- [x] Theoretically, the weighting function $f(.)$ must satisfy $f(0) = 0$.

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? **1 point**

- [x] $s_1 >> s_2$
- [ ] $s_1 << s_2$