## Natural Language Processing & Word Embeddings

1. 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	
∠ <sup>¬</sup> Expand	
Norrect  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.	1 / 1 point
False	
○ True	
∠ <sup>7</sup> Expand	
Correct tr-SNE is a non-linear dimensionality reduction technique.	

## 3. Question 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?)
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.



✓ Correct

Yes, word vectors empower your model with an incredible ability to generalize. The vector for "upset" would contain a negative/unhappy connotation which will probably make your model classify the sentence as a "0".

1/1 point

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?	1/1 point
	None of the above: calling the Python snippet as described above is fine.	
	$\bigcirc$ The correct formula is $E^T*o_{1234}$	
	This doesn't handle unknown words ( <unk>).</unk>	
	It is computationally wasteful.	
	∠ <sup>™</sup> Expand	
	Correct Yes, the element-wise multiplication will be extremely inefficient.	
(	5. When learning word embeddings, words are automatically generated along with the surrounding words.	1/1 point
	False	
	○ True	
	∠ <sup>¬</sup> Expand	
	<ul> <li>Correct</li> <li>We pick a given word and try to predict its surrounding words or vice versa.</li> </ul>	
7.	True/False: In the word2vec algorithm, you estimate $P(t \mid c)$ , where $t$ is the target word and $c$ is a context word and $c$ are chosen from the training set using $c$ as the sequence of all the words in the sentence before $t$ .	1. <i>t</i> 1/1 point
	○ True	
	False	
	∠ <sup>≯</sup> Expand	
	Correct and are chosen from the training set to be nearby words.	

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

0 / 1 point

$$P(t \mid c) = rac{e^{ heta_t^T e_c}}{\sum_{t'=1}^{10000} e^{ heta_t^T e_c}}$$

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

- $hinspace{0.95\textwidth}{0.95\textw$
- $oxed{\Box}$  After training, we should expect  $heta_t$  to be very close to  $e_c$  when t and c are the same word.
- $\ensuremath{igsplus}$   $\theta_t$  and  $e_c$  are both 500 dimensional vectors.

✓ Correc



## **⊗** Incorrect

You didn't select all the correct answers

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

0 / 1 point

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

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

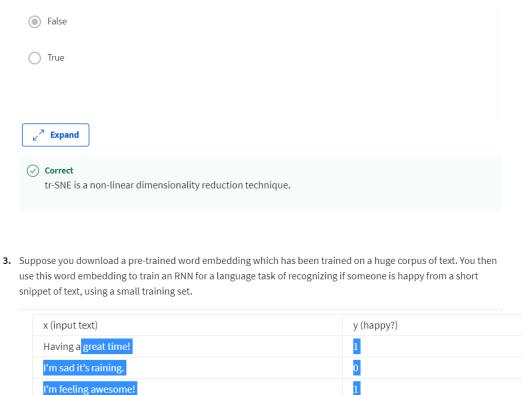
- True
- False



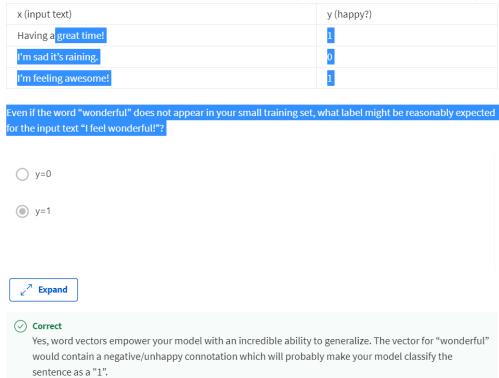
## igotimes Incorrect

No,  $heta_i$  and  $e_j$  should be initialized randomly at the beginning of training.

	$\textbf{10.} \ \ \text{You have trained word embeddings using a text dataset of } t_1 \ \text{words.} \ \ \text{You are considering using these word} \\ \text{embeddings for a language task, for which you have a separate labeled dataset of } t_2 \ \text{words.} \ \ \text{Keeping in mind that} \\ \text{using word embeddings is a form of transfer learning, under which of these circumstances would you expect the} \\ \text{word embeddings to be helpful?}$	1/1 point
	$igcup $ When $t_1$ is smaller than $t_2$ $igcup $ When $t_1$ is larger than $t_2$	
	<ul> <li>✓ Correct         Transfer embeddings to new tasks with smaller training sets.     </li> </ul>	
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
	<ul><li>True</li><li>● False</li></ul>	
	∠ <sup>™</sup> 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.	True/False: t-SNE is a linear transformation that allows us to solve analogies on word vectors.	1/1 point



1/1 point



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  $C^T*o_{1021}$ .

0 / 1 point

1/1 point

TrueFalse

Great, you got all the right answers.

∠<sup>7</sup> Expand

 $\bigotimes$  Incorrect

No, 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
	○ False	
	∠ <sup>¬</sup> Expand  ⊙ Correct	
	In the word2vec algorithm, you estimate $P(t\mid c)$ , where $t$ is the target word and $c$ is a context word. How are $t$ and $c$ chosen from the training set? Pick the best answer.	1/1 point
	$\bigcirc$ $c$ and $t$ are chosen to be nearby words.	
	$\bigcirc \ c$ is the sequence of all the words in the sentence before $t$	
	$\bigcirc \ c$ is the one word that comes immediately before $t$	
	$\bigcirc \ c$ is a sequence of several words immediately before $t$	
	∠ <sup>¬</sup> Expand	
	<b>⊘</b> Correct	

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

1/1 point

$$P(t \mid c) = rac{e^{ heta_t^T e_c}}{\sum_{t'=1}^{10000} e^{ heta_t^T e_c}}$$

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

 $\ensuremath{\nabla}$   $\theta_t$  and  $e_c$  are both 500 dimensional vectors.

✓ Correct

- $\theta_t$  and  $\theta_c$  are both trained with an optimization algorithm such as Adam or gradient

✓ Correct

- $\square$  After training, we should expect  $\theta_t$  to be very close to  $e_c$  when t and c are the same word.
- Suppose you have a 10000 word vocabulary, and are learning 500-dimensional word embeddings. The GloVe model minimizes this objective:

1/1 point

$$\min \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij}) (\theta_i^T e_j + b_i + b_j) - \log X_{ij})^2$$

True/False:  $X_{ij}$  is the number of times word j appears in the context of word i.

- True
- False

∠<sup>7</sup> Expand

**⊘** Correct

 $X_{ij}$  is the number of times word j appears in the context of word i.

10. You have trained word embeddings using a text dataset of $t_1$ words. You are considering using these embeddings for a language task, for which you have a separate labeled dataset of $t_2$ words. Keepin using word embeddings is a form of transfer learning, under which of these circumstances would you word embeddings to be helpful?	g in mind that
$igcup$ When $t_1$ is equal to $t_2$	<u> </u>
$lacktriangledown$ When $t_1$ is larger than $t_2$	- 1
$igcup$ When $t_1$ is smaller than $t_2$	•
∠ <sup>7</sup> Expand	
Correct     Transfer embeddings to new tasks with smaller training sets.	

1 / 1 point