Example Short Answer Questions

1. **The embeddings for the following tokens are shown below. Compute the bag of words document embedding for the following document:**

I love learning

you	**1.0**	**2.5**	**-1.0**	**0.5**
I	**1.0**	**1.0**	**1.5**	**1.0**
car	**-1.0**	**-2.0**	**1.0**	**5.0**
learn	**0.5**	**1.5**	**-2.0**	**-2.5**
love	**0.0**	**-1.0**	**2.0**	**1.5**
hate	**0.5**	**1.0**	**0.5**	**0.0**

When we convert the raw text into tokens and look up their embeddings, we get

I	1.0	1.0	1.5	1.0
Love	0	-1.0	2.0	1.5
Learning	0.5	1.5	-2.0	-2.5
Sum	*1.5*	*1.5*	*1.5*	*0*
Average	*0.5*	*0.5*	*0.5*	*0*

The final document embedding using a bag of words assumption is

0.5 1.0 0.5 0

Study Resources

- Word2vec theory
- Intro to Spacy (see the final cells for an example computation of the document vector)

2. **For a given input document, your transformer model has generated the following attention matrix:**

0.8	**0.1**	**0.05**	**0.05**
0.2	**0.75**	**0.05**	**0.0**
0.0	**0.1**	**0.9**	**0**
0	**0.44**	**0.1**	**0.46**

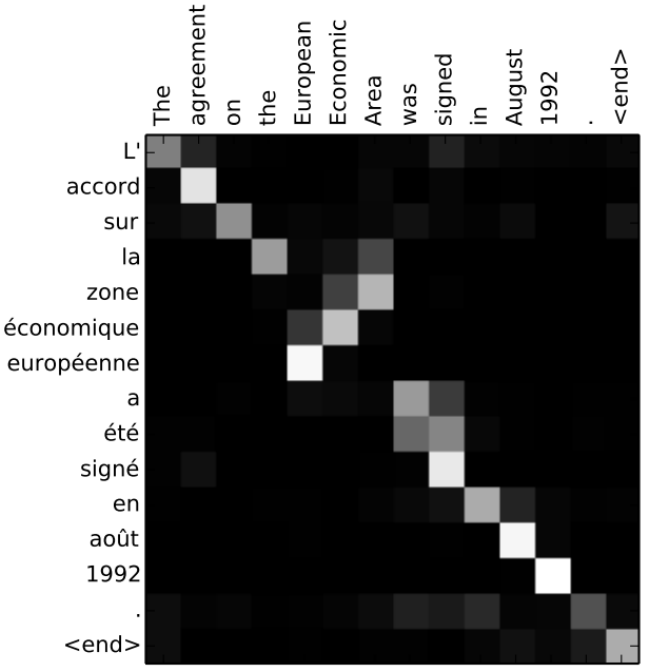
When your transformer attempts to make a prediction for the 3rd sequence step? Which input sequence step would it take most of its signal from?

It would use the 3rd input sequence step, since it has an attention value of 0.9. The information that your transformer would use would be a weighted average combination of inputs from sequence step 2 (10%) and sequence step 3 (90%).

What about for the 4th sequence step?

The model would focus its attention on the 2nd, 3rd, and 4th sequence steps. In particular, it would focus attention on the 2nd and 4th sequence steps.

Another example - in the below example, when making the prediction for the word **européenne**, the model will consider almost exclusively the input from the 5th sequence step (4th index position) – the word "European". However, when it makes its prediction for the word **la**, it will consider the input from the 4th word in the sequence (the) and the 7th word in the sequence (area). This is likely because it needs to understand the word we are translating – the – but also the gender of the word (area) the article refers to.



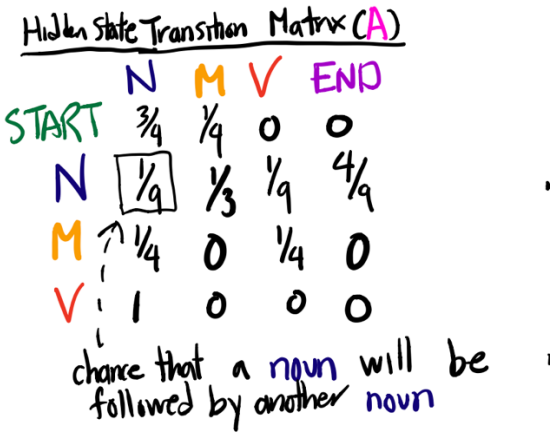
Study Resources

- [Attention and Transformers]
- [Understanding Positional Encodings]
- [Jay Alammar Blog Walkthrough] **(see the final visual of attention matrix for sequence to sequence translation from English to French)**

3. **You are employing a Hidden Markov Model to tag part of speech in your dataset. There are 30,000 unique tokens in your vocabulary and 3 distinct parts of speech (NOUN, VERB, ADJECTIVE).**

What is the shape of your transition matrix?

Your transition matrix would be of shape 3 x 3 (you could also say 4 x 4, to include the START/END states). Each row should sum to 1 and the value in matrix[i][j] represents the probability that starting at hidden state i, you transition next to the hidden state j.



What is the shape of your HMM's emission matrix?

Your emission matrix should be of shape 30,000 x 3. Each column should sum to 1, and the value in matrix[i][j] represents the probability of observing(emitting) the i-th word given we are currently in hidden state j.

The below example is if we had 3 hidden states to model and a unique vocabulary size of 4.

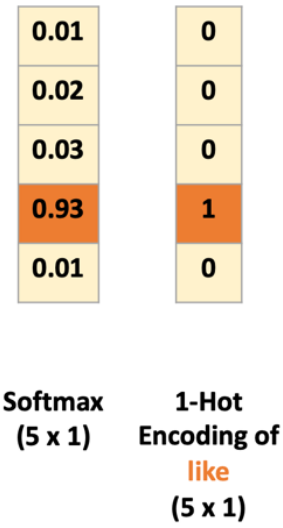
Study Resources

- HMMs for Part of Speech Tagging
- Viterbi Algorithm Example

4. **Imagine that you are using the skipgram architecture to train your own word2vec embeddings. There are 40,000 distinct tokens in your vocabulary. You've chosen to use an embedding size of 50.**
    a. **What is the shape of the final output layer of the skipgram architecture?**

The final output layer should be 40,000 x 1. The values of this matrix should sum up to 1, and represent the probability of seeing a particular context word j given a specific target word i. This output is then compared to the ground truth vector, which will be one-hot encoded with 0s in every element except for the jth entry.

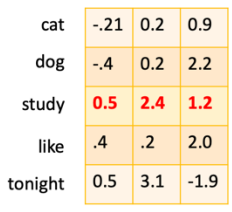
In the example below, there are 5 unique tokens in the vocabulary, and the context word (***like***) is in the 3rd index position of the vocabulary:



b. What is the shape of the trained embeddings matrix?

The shape of the trained embeddings matrix is 40,000 x 50. For each token in our vocabulary, we will have a 50-element vector that represents this word. In the example below, our vocabulary size is 5 and embedding size is 3:

V = 5
M = 3 (embedding size)



This matrix now becomes our "word embeddings". Each word in our vocabulary is now represented as a vector of numbers!

Study Resources

- Word2vec Skipgram Architecture
- Word2vec theory

5. **You have performed PCA (Principal Component Analysis) on your document dataset, choosing the number of components you keep to be 5. You have 20,000 documents and after TF-IDF vectorization, find there are 3,000 features.**

   What is the shape of your new dataset?

The shape of your new dataset should be 20,000 x 5 (# of documents x number of components you selected to keep).

Suppose that you observe the following eigenvalues after PCA. Your goal is to keep around 90% of the original information from your dataset. How many components would you select to keep?

0.50	**0.21**	**0.12**	**0.11**	**0.05**	**0.01**

You would select 4 components, since the total variance captured by the first 4 components is 0.50 + 0.21 + 0.12 + 0.11 = 0.94, or 94%.

If you wanted to retain only 80% of the original variance, you could select only the first 3 components: 0.5 + 0.21 + 0.12 = 0.83 or 83%.

Study Resources

- Dimensionality Reduction

6. **You have used Non-Negative Matrix Factorization to perform topic modelling on a dataset. You hypothesize that there are two main topics in your corpus. Your original dataset looks like this after applying text preprocessing and count vectorization:**

	Cat	Eat	Treat	I	House	Run
Doc #1	**0**	**1**	**2**	**0**	**0**	**0**
Doc #2	**1**	**0**	**1**	**0**	**0**	**1**
Doc #3	**1.5**	**0**	**3**	**1**	**1**	**0**
Doc #4	**2**	**0**	**1**	**0**	**0**	**0**

Your decomposed W matrix looks like this:

1.07	**0**
0.17	**0.77**
1.23	**1.02**

0	**1.28**

Your decomposed H matrix looks like this:

0	**0.40**	**1.83**	**.38**	**.38**	**0**
1.49	**0**	**0.79**	**.15**	**.15**	**.24**

Which two documents are most likely to be about Topic 1?

Document 1 and Document 3 would be most likely to be about Topic 1. You can calculate the percentage topic distributions of each document by dividing the values in each row by the sum of the row. This results in the following modified W matrix:

1.0	**0**
0.18	**0.82**
0.55	**.45**
0	**1**

We can clearly see that Document 1 and Document 3 have the highest distributions of Topic 1.

Which two tokens are most likely to related to Topic 2?

Tokens 1 (cat) and 6 (run) are most likely related to Topic 2. Calculating the distribution of topics for each token, we receive the following modified H matrix:

0	**1**	**0.70**	**.72**	**.72**	**0**
1	**0**	**0.30**	**.28**	**.28**	**1**

We can see that Tokens 1 and 6 are entirely distributed towards Topic 2.

Study Resources

- Topic Modeling

Example True/False Questions

1. **If you load in pre-trained embeddings to use for your embedding layer in a neural network, you should make sure to set that layer's parameter trainable=False.**

True. This is extremely important to remember to do. Otherwise, during backpropagation, the pre-trained embeddings in your embedding matrix will be updated. This defeats the purpose of using pre-trained embeddings.

Study Resources

- Training Your Own Embeddings Notebook and Video Lecture

2. **When performing sentiment classification (ie. returning a score from 0 to 1, 1 being completely positive and 0 being completely negative to represent the final sentiment of the document) using a sequence model such as an RNN, you would only care about the first sequence step's output value and disregard all the other sequence steps' y value.**

False. In this use case, we would likely only care about the final sequence step's y output value, because this represents the predicted sentiment the RNN has generated after observing all of the input tokens (the y output at sequence step 1 would represent the RNN's predicted sentiment after observing only the first token).

Study Resources

- RNNs and LSTMS Theory
- RNNs In Action Slides and Video Lecture

3. **One of the advantages of performing PCA on your vectorized text dataset is that the new features after reducing dimensions are highly correlated and can therefore help improve the predictive signal your model learns from.**

False. The principal components produced from PCA are not correlated with each other. In fact, PCA is often used both to reduce dimensionality, but also to decorrelate highly correlated datasets (for example, if you use ngrams = 2 in a count vectorizer, you'll end up with features like "in drive", "drive through", "through lane" that are highly correlated with each other because they are all decompositions of the phrase "in the drive through lane"

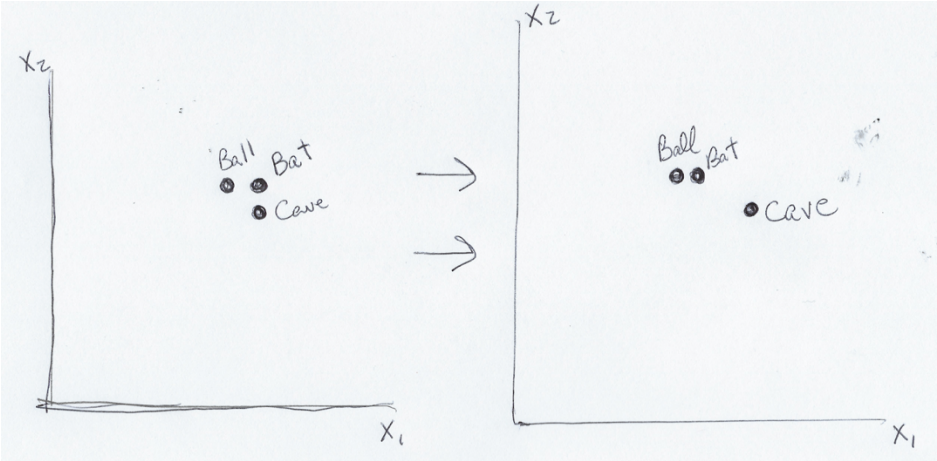
Study Resources

- Dimensionality Reduction

4. **One of the limitations of BERT is that it generates static word embeddings that are unable to incorporate the context that the token is used in in creating its embedded representation.**

False. Perhaps the biggest differentiator between word2vec and traditional word embedding approaches and BERT is that BERT generates dynamic token embeddings that take into consideration the context (the words surrounding the target word, and the index position the target word is found in the document sequence). For instance, while word2vec will produce the same static embedding representation for the word "bat", BERT would theoretically produce different representations based on how the word "bat" is used in the document.

For example, below, the left-hand side represents the static representations from word2vec. The right-hand side represents BERT dynamic embeddings generated from the document "The bat did not make contact with the ball and the batter struck out" – "bat" is clearly used here in a sports context, instead of referring to an animal.



Study Resources

- BERT and Transformers Lecture

5. **If we represent the positional encoding of a token at index position i as PE(i), then**

   cos_similarity(PE(1), PE(3)) > cos_similarity(PE(3, 50)) > cos_similarity(PE(1), PE(50))

True. Positional encodings for index positions that are closer to each other (like 1 and 3) should be significantly more similar than positional encodings for index positions that are farther away (like 1 and 50).

Study Resources

- Understanding Positional Encodings

6. **One of the limitations of BERT is that it cannot be used to generate new text responses from a given input.**

False. You can apply BERT to a variety of business use cases, including generating text responses (both in response to a given input, and unsupervised – ie. predicted the next token after observing a sequence of tokens).

Study Resources

- Huggingface Demo