Natural Language Processing

Tutorial 7 (Week 10): Word Vectors / LM

Question 1

You are working with a small corpus of text from a children's storybook. Here's a sample sentence from the book: "The cat sat on the mat." You decide to use the Skip-Gram model of Word2Vec to learn word vectors for this text.

1) Given the **target word "sat"**, and a **window size of 1**, identify the (target, context) word pairs you would use to train the Skip-Gram model.

Solution 1 (1)

Sentence: *"The cat **sat** on the mat."*

Target word: sat

Window size: 1

context words for sat: *cat, on*.

(target, context) pairs:

- (sat, cat)
- (sat, on)

Question 1

You are working with a small corpus of text from a children's storybook. Here's a sample sentence from the book: "The cat sat on the mat." You decide to use the Skip-Gram model of Word2Vec to learn word vectors for this text.

2) Write the objective function to be maximized with respect to this **target word "sat"** and **context window 1**, ignoring negative sampling.

Solution 1 (2)

Sentence: *"The cat **sat** on the mat."*

Target word: sat

Window size: 1

context words for sat: *cat, on*.

(target, context) pairs:

- (sat, cat)
- (sat, on)

$$\log P(cat|sat) = \log \frac{\exp(u_{cat}^{\top} v_{sat})}{\sum_{w \in V} \exp(u_w^{\top} v_{sat})}$$

$$+$$

$$\log P(on|sat) = \log \frac{\exp(u_{on}^{\top} v_{sat})}{\sum_{w \in V} \exp(u_w^{\top} v_{sat})}$$

Question 1

You are working with a small corpus of text from a children's storybook. Here's a sample sentence from the book: "The cat sat on the mat." You decide to use the Skip-Gram model of Word2Vec to learn word vectors for this text.

3) Write the objective function to be maximized with respect to this **target word "sat"** and **context window 1**, with negative sampling.

Solution 1 (2)

Sentence: *"The cat **sat** on the mat."*

Target word: sat

Window size: 1

context words for sat: *cat, on*.

(target, context) pairs:

- (sat, cat)
- (sat, on)

$$\log P(t=1|cat, sat) = \log \sigma(u_{cat}^\top v_{sat})$$
$$+$$
$$\log P(t=1|on, sat) = \log \sigma(u_{on}^\top v_{sat})$$
$$+$$
$$\sum_k \log P(t=0|w_k, sat) = \sum_k \log \sigma(-u_{w_k}^\top v_{sat})$$

Question 1

You are working with a small corpus of text from a children's storybook. Here's a sample sentence from the book: "The cat sat on the mat." You decide to use the Skip-Gram model of Word2Vec to learn word vectors for this text.

4) Suppose after training, you obtain the following word vectors. Which word is most similar to "cat" based on cosine similarity? Are these vectors well learned?

Question 1

Word	Vector Representation	L2-norm
cat	[0.5, 1.0, -0.3]	1.16
mat	[0.6, 0.8, -0.2]	1.02
sat	[-0.1, 0.9, 0.3]	0.95
hat	[0.6, 1.0, -0.4]	1.23
dog	[-1.0, -0.5, 0.8]	1.46

Solution 1 (4)

Word	Vector Representation	L2-norm
cat	[0.5, 1.0, -0.3]	1.16
mat	[0.6, 0.8, -0.2]	1.02
sat	[-0.1, 0.9, 0.3]	0.95
hat	[0.6, 1.0, -0.4]	1.23
dog	[-1.0, -0.5, 0.8]	1.46

The cosine similarity between two vectors A and B is given by

$$cossim(A, B) = \frac{AB}{||A||||B||}$$

$$cossim(cat, mat) = \frac{0.5\times0.6+1.0\times0.8+0.3\times0.2}{1.16\times1.02} = 0.98$$

$$cossim(cat, sat) = \frac{-0.5\times0.1+1.0\times0.9-0.3\times0.3}{1.16\times0.95} = 0.67$$

$$cossim(cat, hat) = \frac{0.5\times0.6+1.0\times1.0+0.3\times0.4}{1.16\times1.23} = 1.00$$

$$cossim(cat, dog) = \frac{-0.5\times1.0-1.0\times0.5-0.3\times0.8}{1.16\times1.46} = -0.73$$

Solution 1 (4)

Word	Vector Representation	L2-norm
cat	[0.5, 1.0, -0.3]	1.16
mat	[0.6, 0.8, -0.2]	1.02
sat	[-0.1, 0.9, 0.3]	0.95
hat	[0.6, 1.0, -0.4]	1.23
dog	[-1.0, -0.5, 0.8]	1.46

The cosine similarity between two vectors A and B is given by

$$cossim(A, B) = \frac{AB}{||A||||B||}$$

$$cossim(cat, mat) = \frac{0.5\times0.6+1.0\times0.8+0.3\times0.2}{1.16\times1.02} = 0.98$$

$$cossim(cat, sat) = \frac{-0.5\times0.1+1.0\times0.9-0.3\times0.3}{1.16\times0.95} = 0.67$$

$$cossim(cat, hat) = \frac{0.5\times0.6+1.0\times1.0+0.3\times0.4}{1.16\times1.23} = 1.00$$

$$cossim(cat, dog) = \frac{-0.5\times1.0-1.0\times0.5-0.3\times0.8}{1.16\times1.46} = -0.73$$

No. These vectors are not well learned because cat should be more similar to dog.

Question 1

You are working with a small corpus of text from a children's storybook. Here's a sample sentence from the book: "The cat sat on the mat." You decide to use the Skip-Gram model of Word2Vec to learn word vectors for this text.

5) Explain whether cosine similarity is used in the training objective function, and discuss why.

Solution 1 (5)

No, the object function uses dot product instead of cosine similarity. Possible reasons include

- Computing the dot product is computationally simpler than computing cosine similarity, especially when dealing with large vocabularies.
- Cosine similarity values are bounded between -1 and 1. This constraint can influence the dynamics of training, potentially causing saturation or making optimization more challenging.
- Common optimization algorithms used for training deep learning models (like stochastic gradient descent and its variants) work well with the kind of loss surfaces and gradients produced when using dot products. Changing to cosine similarity could alter these dynamics.

Question 2

Assume we have simplified vectors for the words: happy, joyful, sad, angry, calm, peaceful, tense, and frustrated. Their vectors are:

- “happy” = [6, 4]
- “joyful” = [5, 5]
- “sad” = [1, 2]
- “angry” = [2, 6]
- “calm” = [4, 1]
- “peaceful” = [4, 2]
- “tense” = [1, 5]
- “frustrated” = [2, 5]

1) Use vector arithmetic to find a word that should be as opposite to “happy” as “angry” is to “calm”.

Solution 2 (1)

1) Use vector arithmetic to find a word that should be as opposite to “happy” as “angry” is to “calm”.

$$x - happy = angry - calm$$
$$x = happy + angry - calm$$
$$x = [6, 4] + [2, 6] - [4, 1] = [4, 9]$$

Solution 2 (1)

1) Use vector arithmetic to find a word that should be as opposite to "happy" as "angry" is to "calm".

$$x - happy = angry - calm$$
$$x = happy + angry - calm$$
$$x = [6, 4] + [2, 6] - [4, 1] = [4, 9]$$

- "happy" = [6, 4] —> 0.84
- "joyful" = [5, 5] —> 0.93
- "sad" = [1, 2] —> 0.9989
- "angry" = [2, 6] —> 0.996

- "calm" = [4, 1] —> 0.62
- "peaceful" = [4, 2] —> 0.77
- "tense" = [1, 5] —> 0.975
- "**frustrated**" = [2, 5] —> 0.9992

Question 2

Assume we have simplified vectors for the words: happy, joyful, sad, angry, calm, peaceful, tense, and frustrated. Their vectors are:

- “happy” = [6, 4]
- “joyful” = [5, 5]
- “sad” = [1, 2]
- “angry” = [2, 6]
- “calm” = [4, 1]
- “peaceful” = [4, 2]
- “tense” = [1, 5]
- “frustrated” = [2, 5]

2) Use vector arithmetic to find a word that is to “happy” as “peaceful” is to “calm”.

Solution 2 (2)

2) Use vector arithmetic to find a word that is to “happy” as “peaceful” is to “calm”.

$$x - happy = peaceful - calm$$
$$x = happy + peaceful - calm$$
$$x = [6, 4] + [4, 2] - [4, 1] = [6, 5]$$

Solution 2 (2)

2) Use vector arithmetic to find a word that is to “happy” as “peaceful” is to “calm”.

$$x - happy = peaceful - calm$$
$$x = happy + peaceful - calm$$
$$x = [6, 4] + [4, 2] - [4, 1] = [6, 5]$$

- “happy” = [6, 4] —> 0.994
- “joyful” = [5, 5] —> 0.996
- “sad” = [1, 2] —> 0.916
- “angry” = [2, 6] —> 0.851

- “calm” = [4, 1] —> 0.901
- “peaceful” = [4, 2] —> 0.974
- “tense” = [1, 5] —> 0.778
- “**frustrated**” = [2, 5] —> 0.880

Question 3

Imagine you're working with a simple neural language model designed to predict the next word in a sentence. The model has been trained on a small dataset of children's stories. Here's a sample from the corpus: *"Jack loves climbing trees"*.

1) Give the mathematical formulation on the probability of the above sentence.

Solution 3 (1)

$$P(s) = P(\text{“<s> Jack loves climbing trees </s>”})$$

$$= P(\text{Jack}|\text{<s>}) \cdot P(\text{loves}|\text{<s> Jack}) \cdot P(\text{climbing}|\text{<s> Jack loves})$$

$$\cdot P(\text{trees}|\text{<s> Jack loves climbing}) \cdot P(\text{</s>}|\text{<s> Jack loves climbing trees})$$

Question 3

Imagine you're working with a simple neural language model designed to predict the next word in a sentence. The model has been trained on a small dataset of children's stories. Here's a sample from the corpus: *"Jack loves climbing trees"*.

2) If Markov Assumption with window size 1 is used, what is the revised probability of the above sentence?

Solution 3 (2)

If we're using a Markov assumption of size 1 (bigram model), then each word only depends on the previous word. In such a scenario, our equation simplifies to:

$$
\begin{aligned}
P(s) &= P(\text{“<s> Jack loves climbing trees </s>”}) \\
&= P(\text{Jack}|\text{<s>}) \cdot P(\text{loves}|\text{Jack}) \cdot P(\text{climbing}|\text{loves}) \\
&\cdot P(\text{trees}|\text{climbing}) \cdot P(\text{</s>}|\text{trees})
\end{aligned}
$$

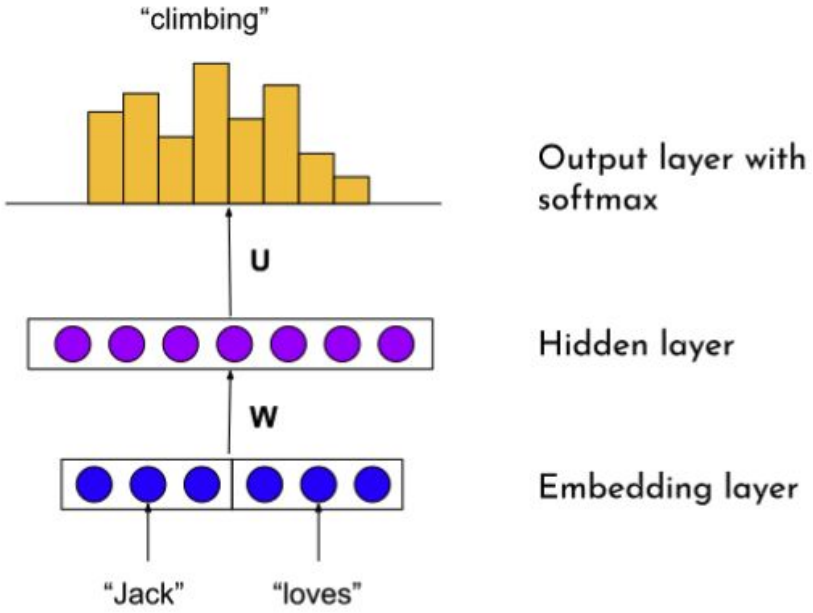
Question 3

Imagine you're working with a simple neural language model designed to predict the next word in a sentence. The model has been trained on a small dataset of children's stories. Here's a sample from the corpus: *"Jack loves climbing trees"*.

3) Describe the transformation process of the input words "Jack loves" into a probability distribution over the next possible word in the vocabulary. Outline the key stages in this transformation.

Solution 3 (3)

The overall framework can be illustrated in the following figure:



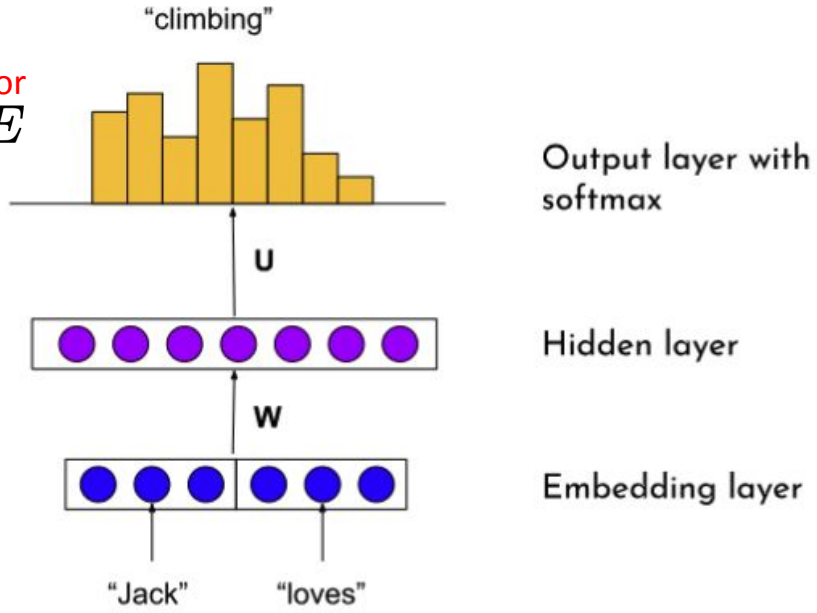
Solution 3 (3)

The overall framework can be illustrated in the following figure:

- Embedding layer: $e \in \mathbb{R}^d$

1-hot vector

$$e_{Jack} = x_{Jack} \cdot E \quad e_{loves} = x_{loves} \cdot E$$



Solution 3 (3)

The overall framework can be illustrated in the following figure:

- Embedding layer: $e \in \mathbb{R}^d$

1-hot vector

$$e_{Jack} = x_{Jack} \cdot E \quad e_{loves} = x_{loves} \cdot E$$

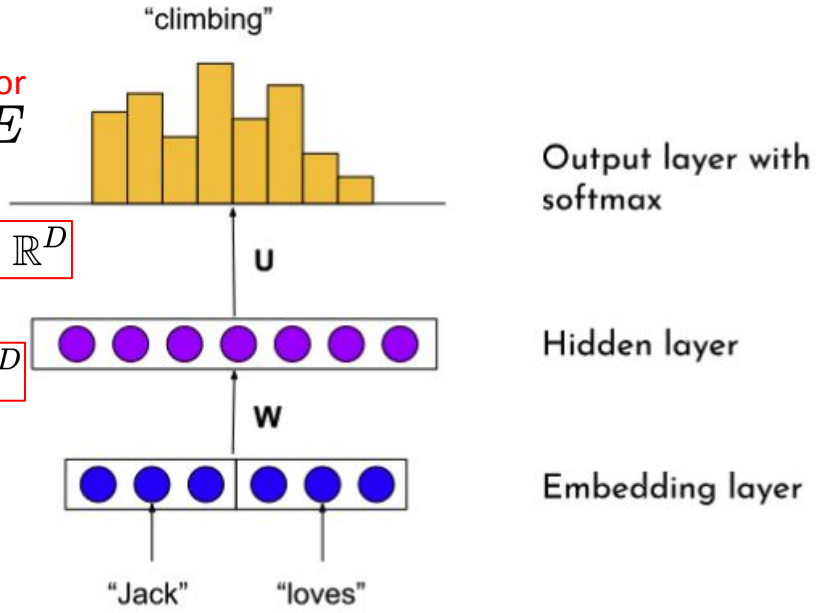
- Hidden layer: $h \in \mathbb{R}^D$

$W \in \mathbb{R}^{D \times d}, b \in \mathbb{R}^D$

$$h = \tanh(W(e_{Jack} + e_{loves})/2 + b)$$

$W \in \mathbb{R}^{D \times 2d}, b \in \mathbb{R}^D$

$$h = \tanh(W[e_{Jack}; e_{loves})] + b)$$



Solution 3 (3)

The overall framework can be illustrated in the following figure:

- Embedding layer: $e \in \mathbb{R}^d$

1-hot vector

$$e_{Jack} = x_{Jack} \cdot E \quad e_{loves} = x_{loves} \cdot E$$

- Hidden layer: $h \in \mathbb{R}^D$

$W \in \mathbb{R}^{D \times d}, b \in \mathbb{R}^D$

$$h = \tanh(W(e_{Jack} + e_{loves})/2 + b)$$

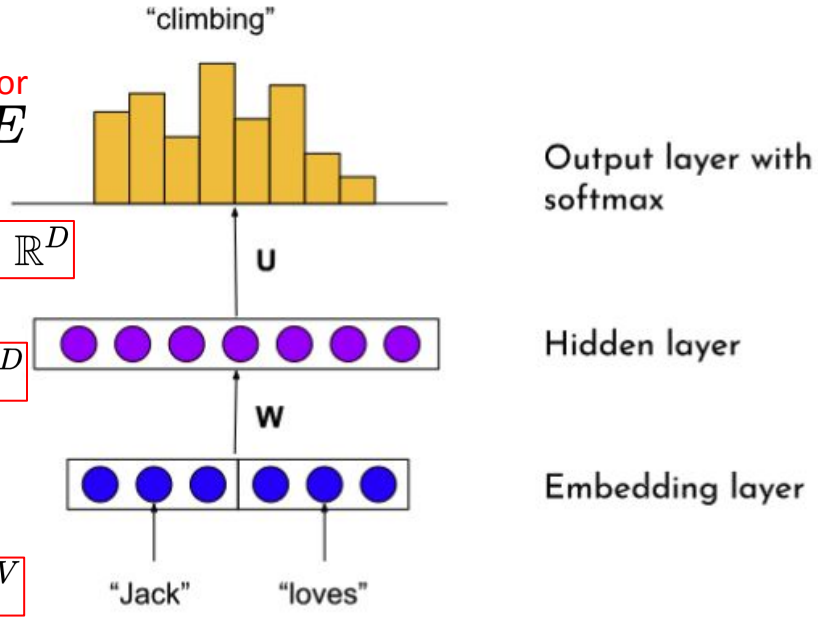
$W \in \mathbb{R}^{D \times 2d}, b \in \mathbb{R}^D$

$$h = \tanh(W[e_{Jack}; e_{loves})] + b)$$

- Output layer: $y \in \mathbb{R}^V$

$$y = softmax(Uh + c)$$

$U \in \mathbb{R}^{V \times D}, c \in \mathbb{R}^V$



Coding Practice

- Word2vec: https://colab.research.google.com/drive/164dB-Vemzwavf1ffqDDVNtx7Y5VtcmQh?usp=sharing