NLP esame giugno

July 9, 2024

1 Exercise 3.a

1.1

$$\#parameter = |V| * |D| * 2 = 5 * 10^6 * 100 * 2 = 10 * 10^9$$

1.2

To train W2V we make use of the cross entropy loss function

$$\mathcal{L} = -y^T * \ln(softmax(\theta_C * \theta_W[i]^T))$$

where θ_C is the context embedding matrix, $\theta_W[i]^T$ is the center embedding matrix where we fix the ground truth center word embedding at position i, the vector y is the one hot encoding vector that has 1 only in position of the ground truth token i.
This means y will select only the probability associated with the ground truth.

Another solution would be to describe the loss with negative sampling

To train w2v we use the cross entropy function with negative sampling.

$$\mathcal{L} = -\ln \sigma(\theta_W[gt]^T * \theta_C[i]) - \sum_{k=1}^{K} \ln \sigma(\theta_W[k] * \theta_C[i]^T)$$

where θ_C is the context embedding matrix, θ_W is the center embedding matrix. We are getting close to similar center and context embedding, and moving away from sampled negative center and context embedding.

1.3

Comparing center words with all context words ensures that the model learns to represent meaningful relationships between words.

2 Exercise 3.b

2.1

To do LSA we must use SVD (singular value decomposition) to decompose our *tf-idf* matrix into 3 different matrices $X \approx USV^T$, where $U \in \mathbb{R}^{|D|\times|D|}$, $S \in \mathbb{R}^{|D|\times|V|}$ is a diagonal matrix with each value corresponding to topic relevance (naturally sorted in descending order), and $V^T \in \mathbb{R}^{|V|\times|V|}$. But since we are searching for k latent topics we must truncate the matrices considering just the k^{th} most important singular values; the new matrices will be $X \approx X_k = U_k S_k V_k^T$, where $U_k \in \mathbb{R}^{|D|\times k}$, $S \in \mathbb{R}^{k\times k}$, $V_k \in \mathbb{R}^{|V|\times k}$. The algorithm for SVD is the following:

$$argmin_{U,S,V} \|X - USV^T\|_2^F \text{ such that } UU^T = Id, VV^T = Id, S \text{ is diagonal}$$

We seek USV^T to be low rank. Then we truncate all matrices to the k^{th} column to keep only the most important latent topics.

2.2

The most dominant topic is the highest singular value in S, which is the first row in the matrix. To find terms that contribute most and terms that contribute negatively we can simply multiply $S[0]V^T$ ($S[0] \in \mathbb{R}^{1\times k}$) which will generate a vector $sv \in \mathbb{R}^{1\times|V|}$ that will contain how much a term i contribute to the topic.

2.3

To encode a new document in the new space we must calculate the *tf* vector d of the document and multiply it by our matrix V_k^T, so

$$V_k^T d$$

2.4

We make sure to learn different topics forcing the orthonormality constraints on the D matrix so that each U and V dimensions conveys unique information. Also, we seek USV^T to be low rank because if we do not impose a bottleneck we can reconstruct perfectly the data, so we aren't learning anything.

3 Exersice 4.a

3.1

Nope

3.2 4.b

3.3

In class, we reviewed the LSTM model, which has the forgetting factors. The LSTM has 3 gates: forget, input, and output.
The model has the following equation:

$$F = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \tag{1}$$
$$I = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \tag{2}$$
$$O = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \tag{3}$$
$$\tilde{C}_t = \tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \tag{4}$$

Putting it all together:

$$c_t = F \odot c_{t-1} + I \odot \tilde{C}_t$$
$$h_t = O \odot \tanh(c_t)$$

3.4

We must do the following, we declare the new

$$F' = \sigma(X_t W_{xf} + b_f)$$
$$c_t \doteq c_{t-1} \odot F'$$

4 Exercise 5.a

We have our learnable matrices $K, Q, V \in \mathbb{R}^{D \times D}$, K and V come from the encoder while Q comes from the decoder. We have Y from the encoder and X from decoder $\in \mathbb{R}^{N \times D}$.
The learnable cross-attention formula with the single head is:

$$softmax\Big[XQ(YK)^T\Big]YV$$

4.1

They can have different sizes, things about, and autoregressor that spawn new tokens each iteration, the Y matrix will be different from X.
We can also show it using different sizes in the formula: $X \in \mathbb{R}^{K \times D}$ and $Y \in \mathbb{R}^{N \times D}$, we have that $XQ \in \mathbb{R}^{K \times D}$, $(YK)^T \in \mathbb{R}^{D \times N}$, $YV \in \mathbb{R}^{N \times D}$. So $XQ(YK)^T \in \mathbb{R}^{K \times N}$ and

$$softmax\Big[XQ(YK)^T\Big]YV \in \mathbb{R}^{K \times D}$$

So our cross attention values will be on the tokens we have previously generated.

4.2

Since our task is language modeling, we must not look into the future tokens. Because otherwise, it would be like cheating. So if after one iteration our loss goes to zero is possible that we are not masking future tokens, so the model cheats and over-fit immediately.

4.3

We can use a projection matrix to scale the vector x to the same dimension of y.

5 Exercise 6.a

5.1

Yes, CLIP can solve the classification problem.

5.2

We can use a beam search algorithm as follows:

1. Start with a prompt containing a masked word, such as 'In the image we have the generic object < object >.'
2. Pass the prompt and the image to CLIP to get the prediction score and predicted class.
3. Embed the predicted object into the prompt to form a new sentence, like "In the image we have the generic object ¡previous object¿ which is the generic form of < object >."
4. This generates the next node in the search tree. Continue embedding the updated prompts in this format, collecting prediction scores.
5. At each level, beam search generates k predictions. Calculate the log-likelihood of each leaf, normalizing by the branch depth.

The final class is the one with the highest probability.