Computational process for "Word Embedding":

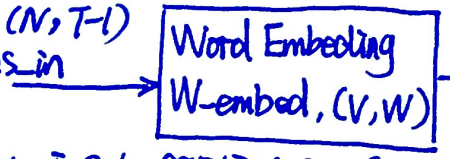
- given images input "features": (N, D) N: number of images, D: feature dimension for each image
- "captions" with size (N, T). Each row here is a vector, each element in a vector is a number, correspond to a word using (idx-to-word). T means there T number total of words in each word vector.

- captions_in = captions[:, :-1], size (N, T-1)
- captions_out = captions[:, 1:], size (N, T-1)
- mask = (captions_out != self._null) size (N, T-1)

(1) • Transfer input features into hidden inputs.

$h0 = features \cdot W_{proj} + b_{proj}$
(N, D), (D, H), (H,)

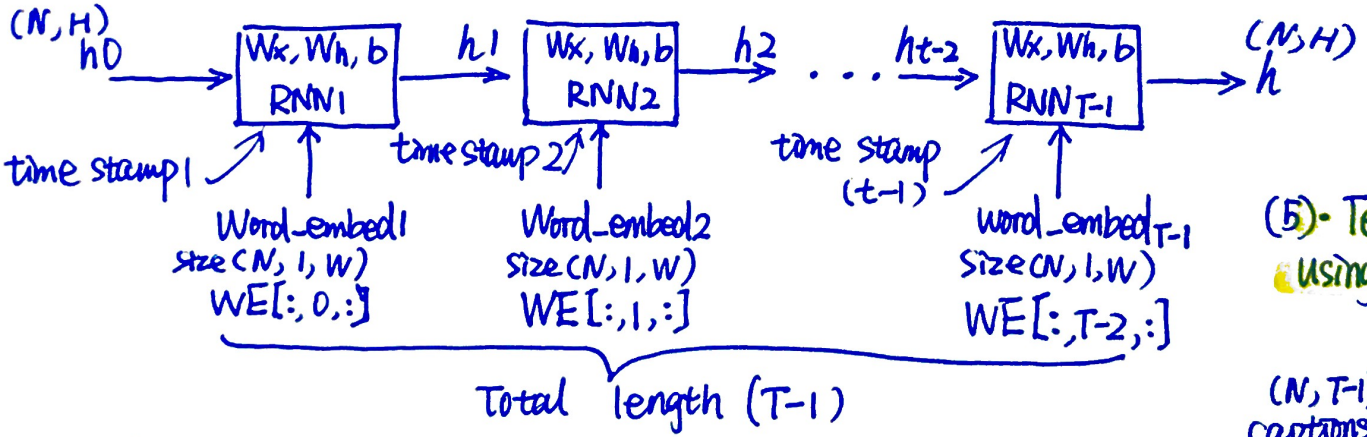
(2) • Use Word embeding to transfer captions_in



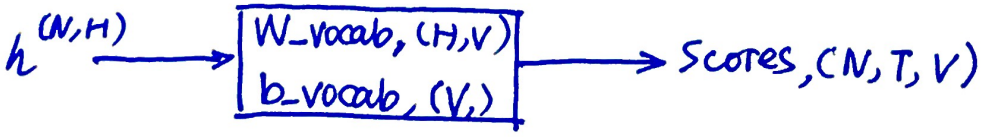
V: Vocabulary size, how many words.
W: dimension for each word.

convert input captions_in (a series of words) into a 3D matrix, where each word at a time point has its own vector.

(3) • Recurrent Neural Network (RNN) process:
- use either Vanilla RNN or lstm RNN, Wx, Wh, b might have different size, depends which RNN.



(4) • Temporal affine transform to compute scores of all words at each time stamp.



(5) • Temporal softmax to compute loss using captions_out.