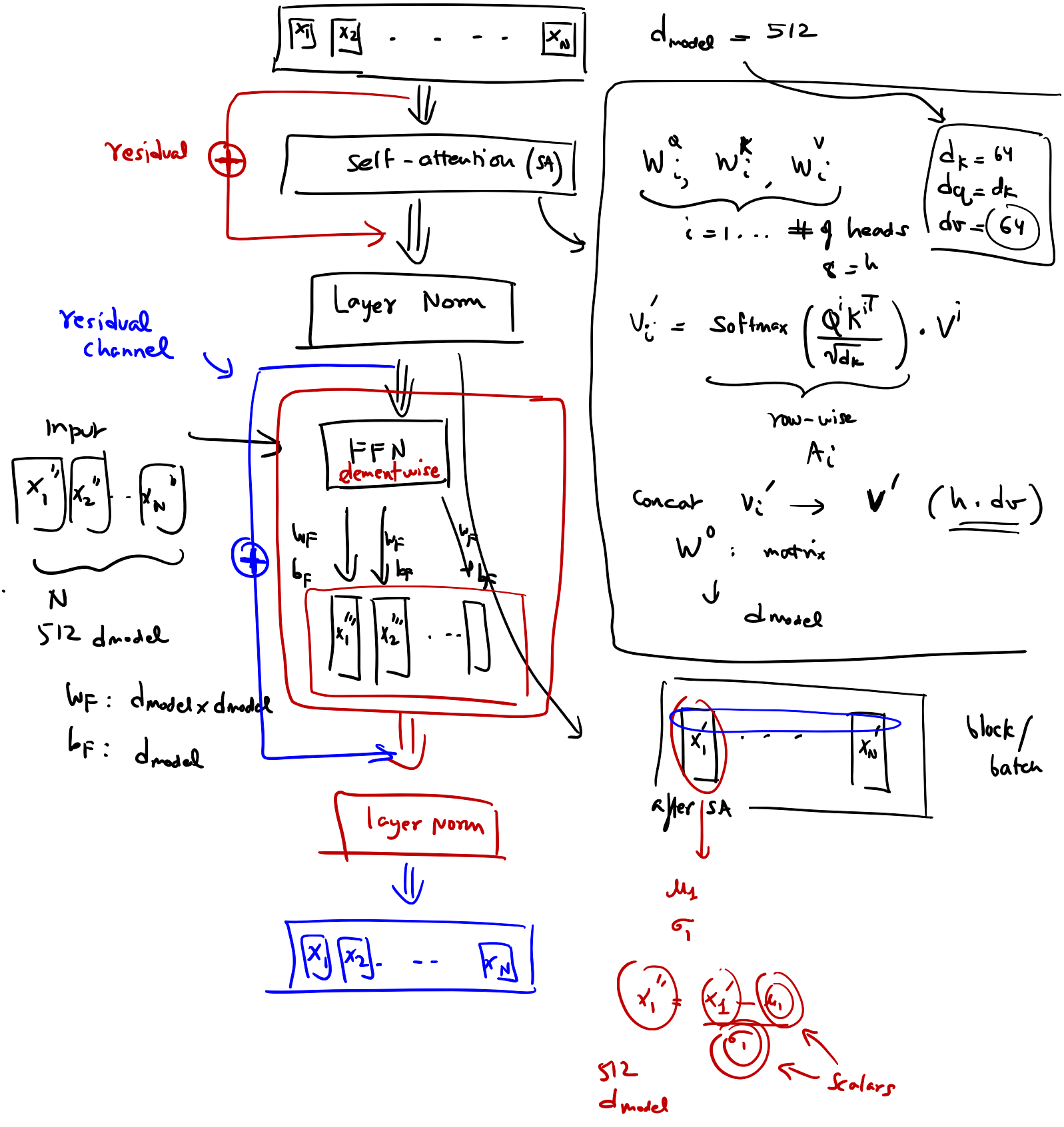
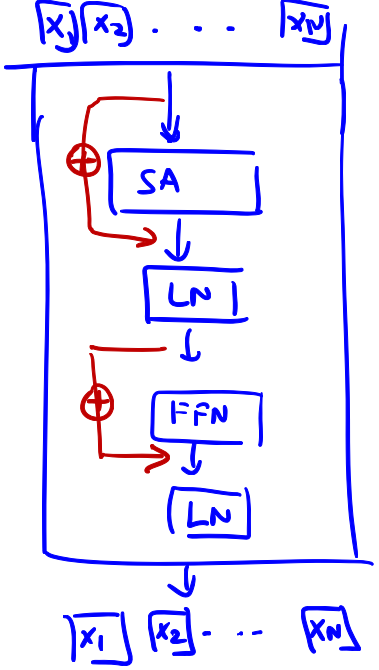
Transformer

self-attention : Multi-headed

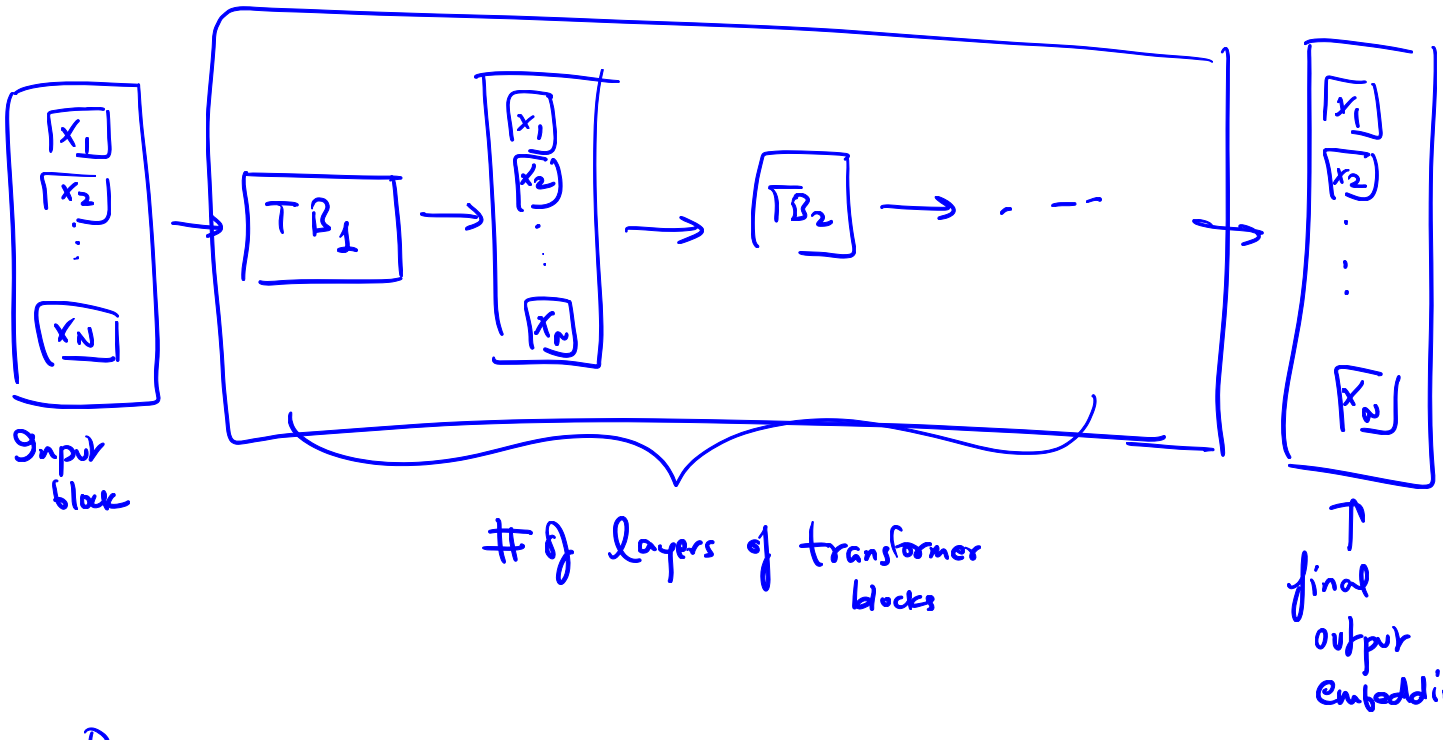


Transformer block

$d_{model} = 512$



TB



Pretraining

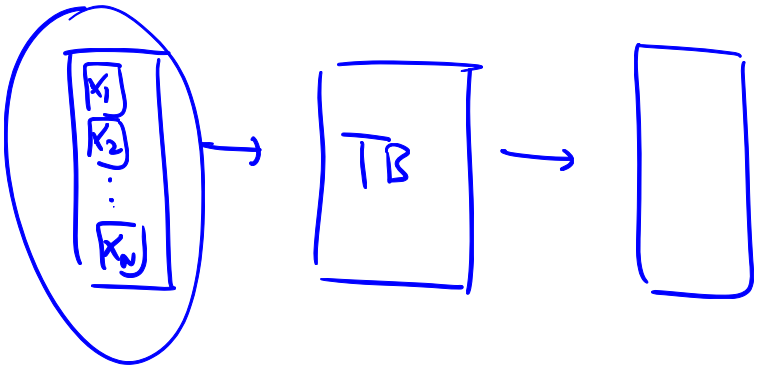
↳ Word2vec : 1 token/word → 1 embedding

I was sitting by the river bank →

I went to the bank and sat on the bench →

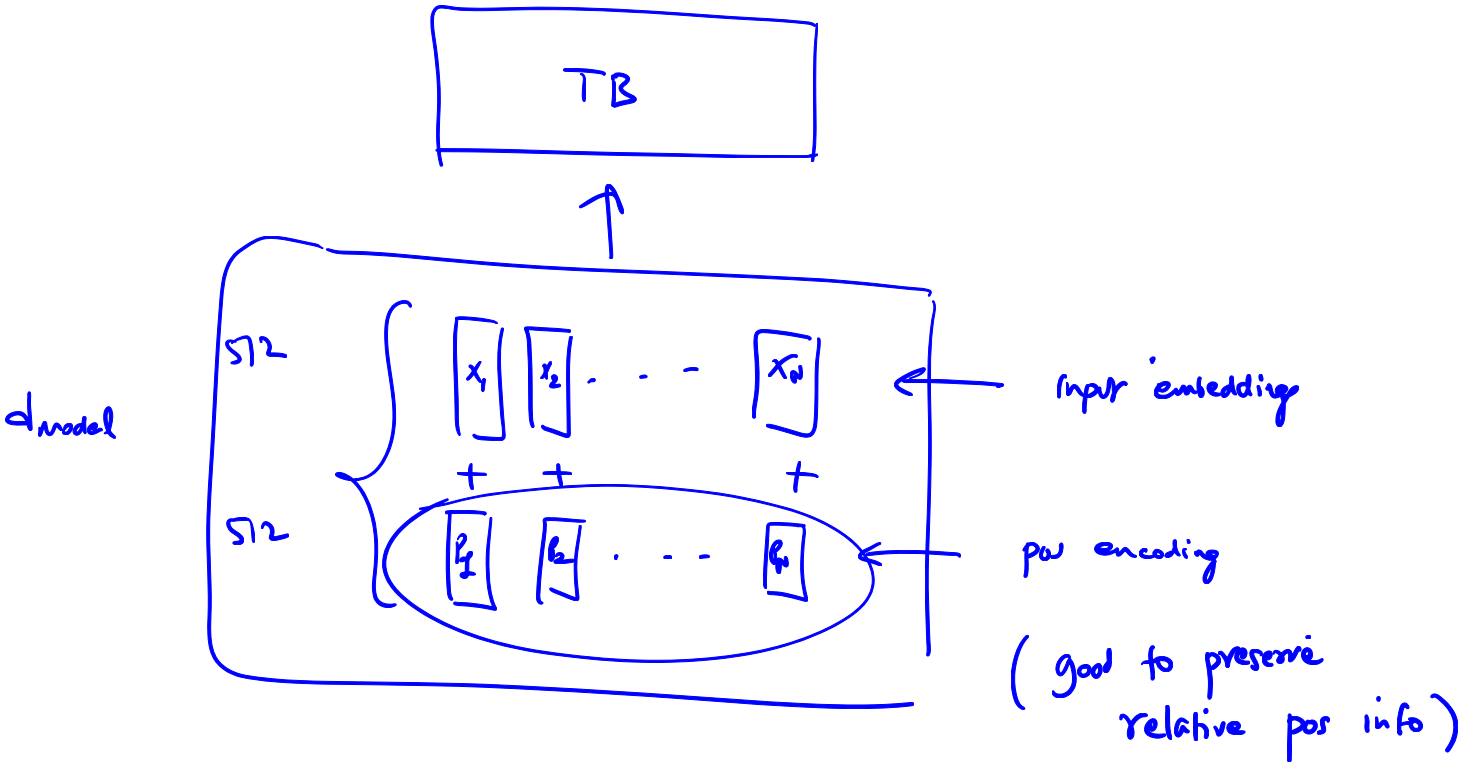
↳ transformer :

1 token/word → multiple embeddings (as many blocks/occurrences)

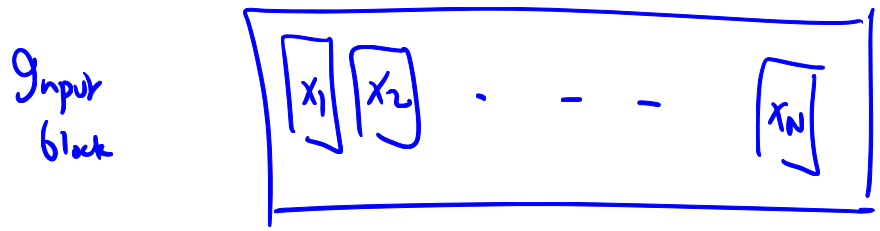


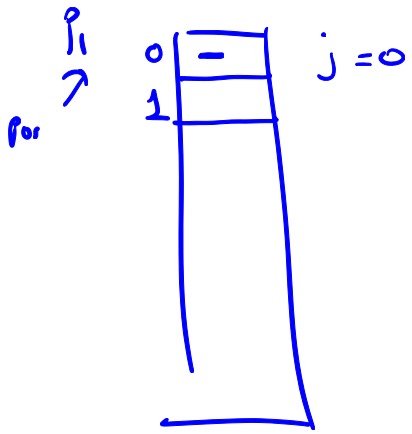
no order info

↳ Position Encoding



Pos embedding: 1) Cos / sin based
2) preserve relative pos info





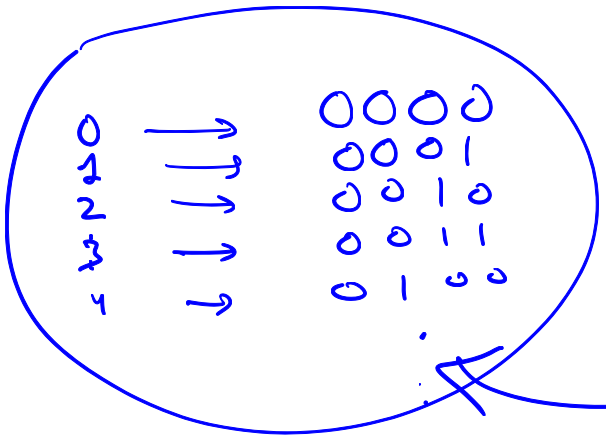
$$P_{pos,2j} = \sin\left(\frac{pos}{(10000)^{2j/d_{model}}}\right)$$

pos = 1

dim

$$P_{pos,2j+1} = \cos\left(\frac{pos}{(10000)^{2j/d_{model}}}\right)$$

dim



binary encoding

Continuous version of binary encoding

BERT: pre-trained model to generate contextual embeddings

↳ base model

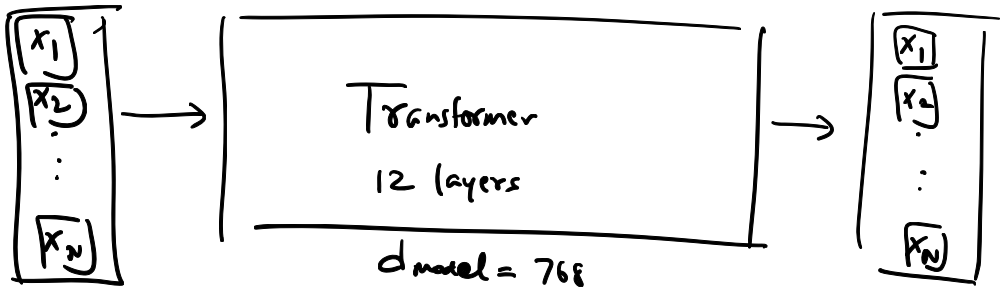
$d_{model} = 768$

block size = 512 = N $x_1 \cdots x_N$

# of layers = 12 (transformer layers)

# of heads = 8/12 ?

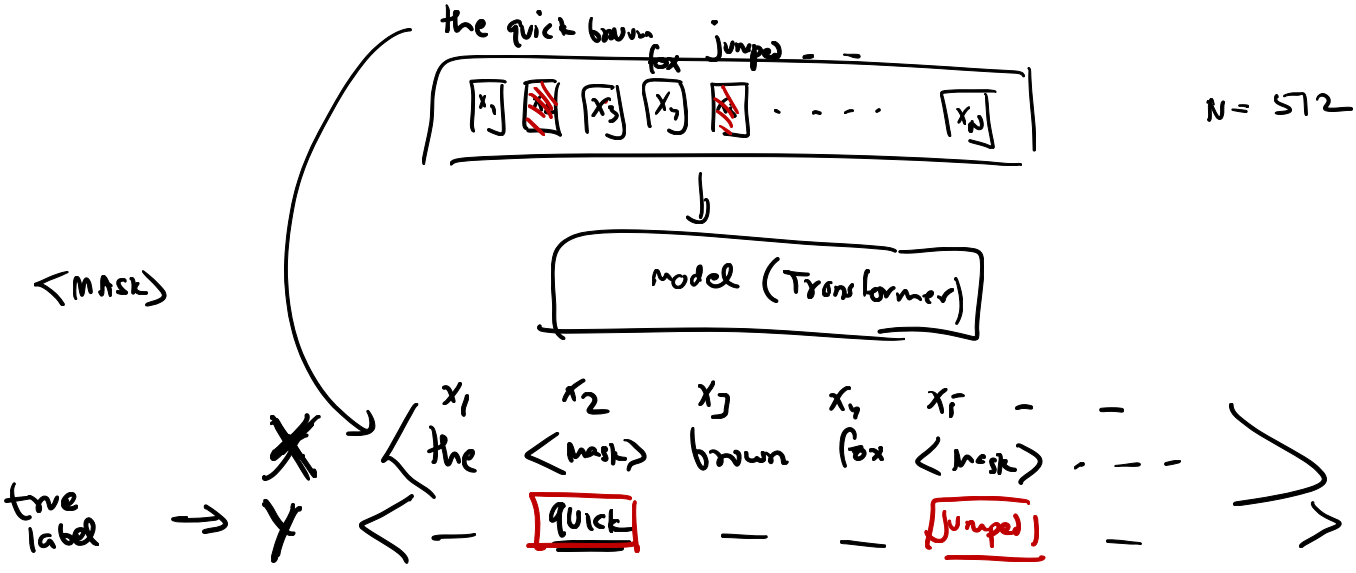
$d_k = d_q$, d_v = 64

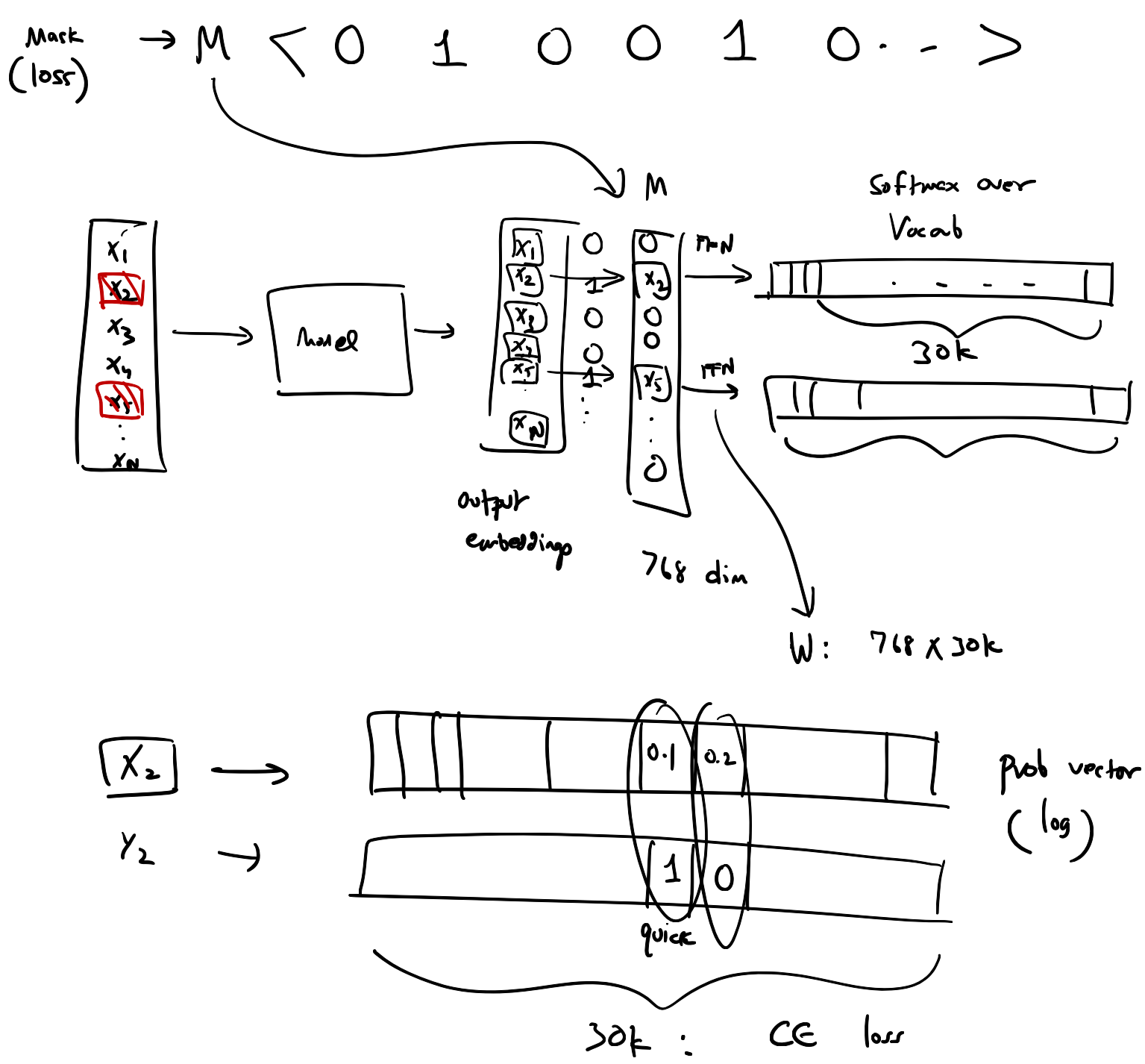


30k vocab of tokens (word-piece)

MLM: Masked Language model

(self-supervised)





BERT :

15% of the positions are masked randomly

a) 80% replace the actual token with <mask>

b) 10% replace with a random token

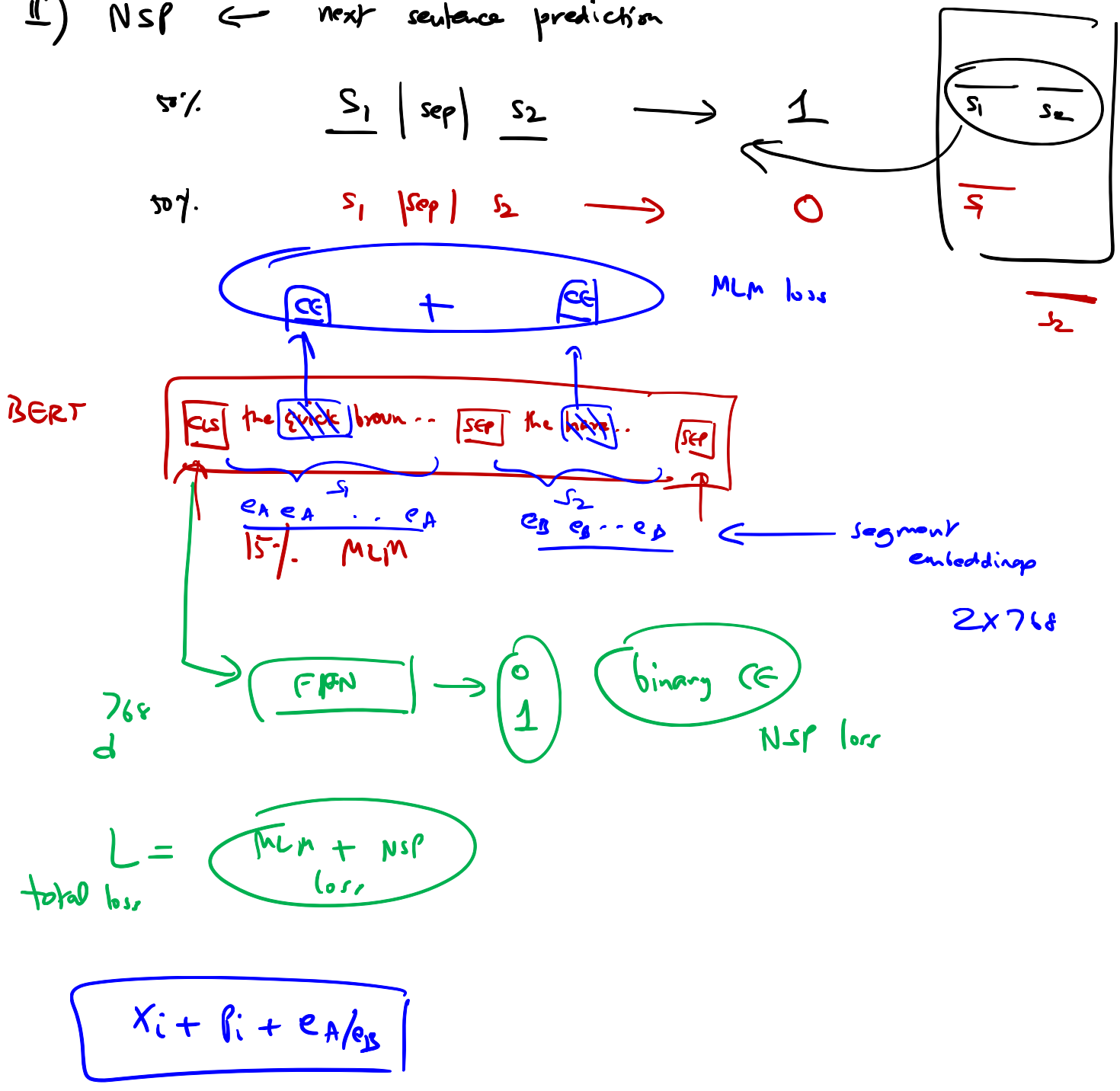
the quick brown fox

the bank brown fox

c) 10%. leave as is
the quick brown fox

I) MLM ← loss CE over 30k vocab
per masked position

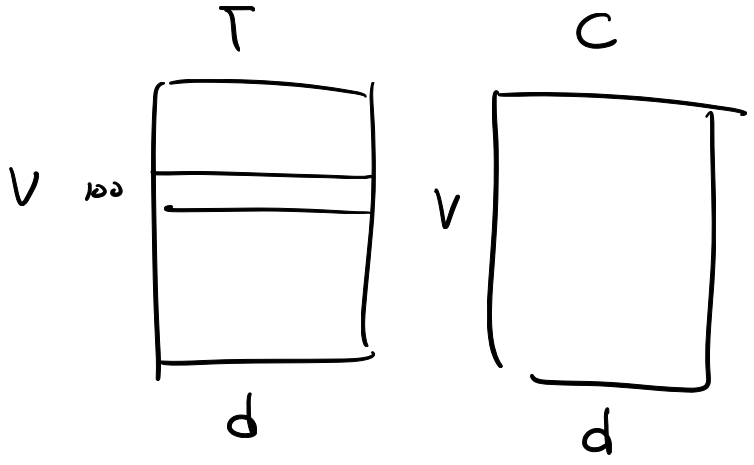
II) NSP ← next sentence prediction



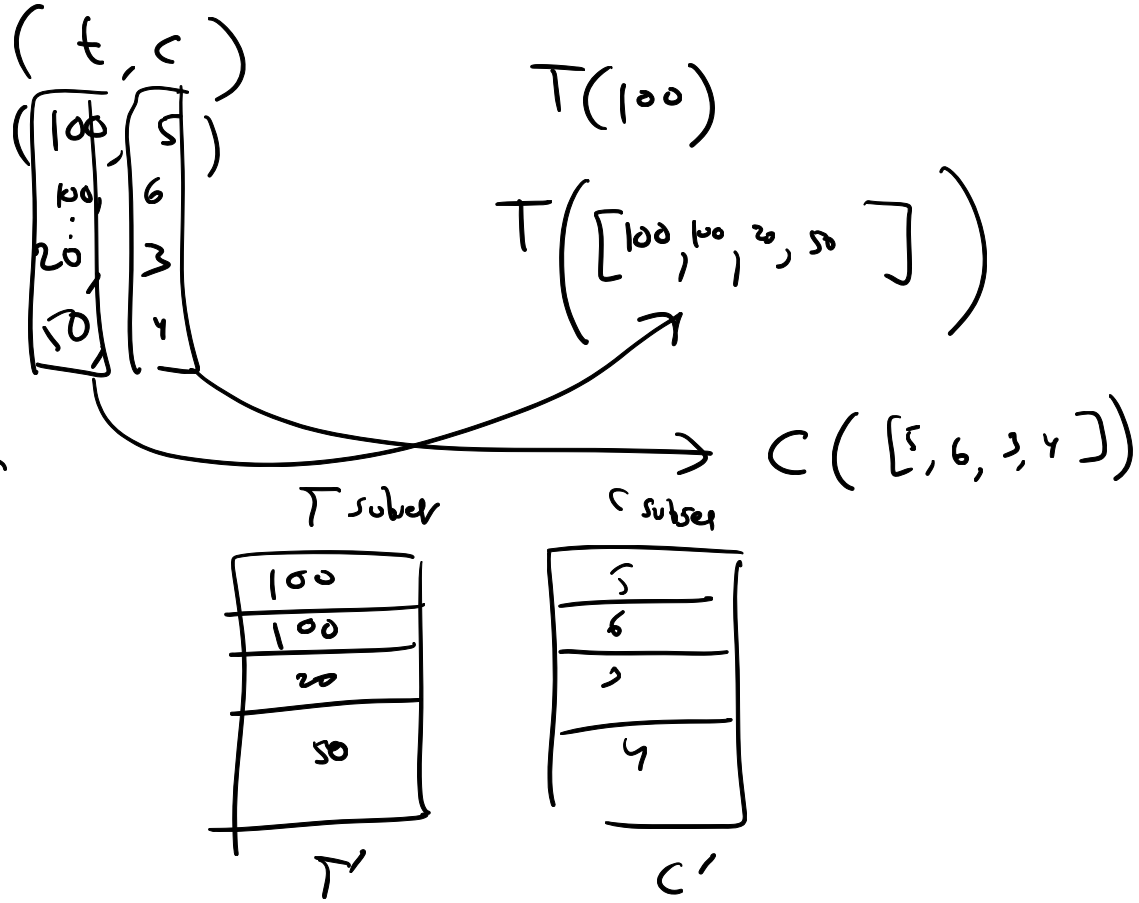
Word2vec

Model

nn.Embedding (linear / Dense)

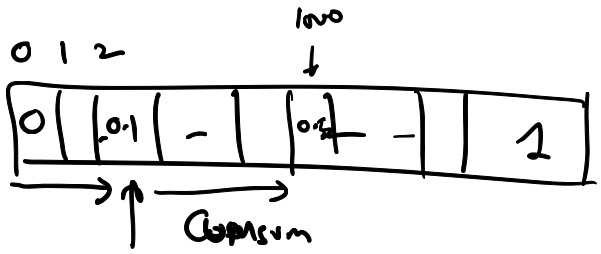


T = nn.Embedding(V, d)



logits = torch.sum(T' * C', dim = 1)

BCE with logits



random = 0.1 idx 2
0.5 idx 1000

np.random.choices(, 1000)
prob vector

np.searchsorted( )
(0.1, 0.5, ..)
1024
Cumsum
V