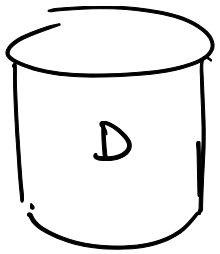
BERT Contextual Model

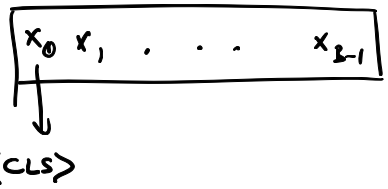


S_1
S_2
$\vdots$
S_N

$N \approx 500k$

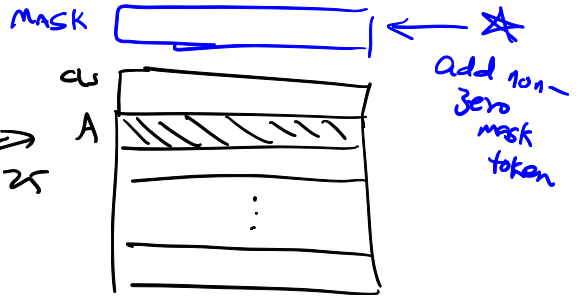
$|S| \leq 1000$

k-mer = 1

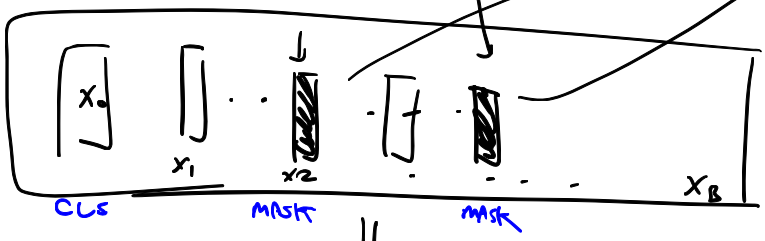


B = block size

B = 1000

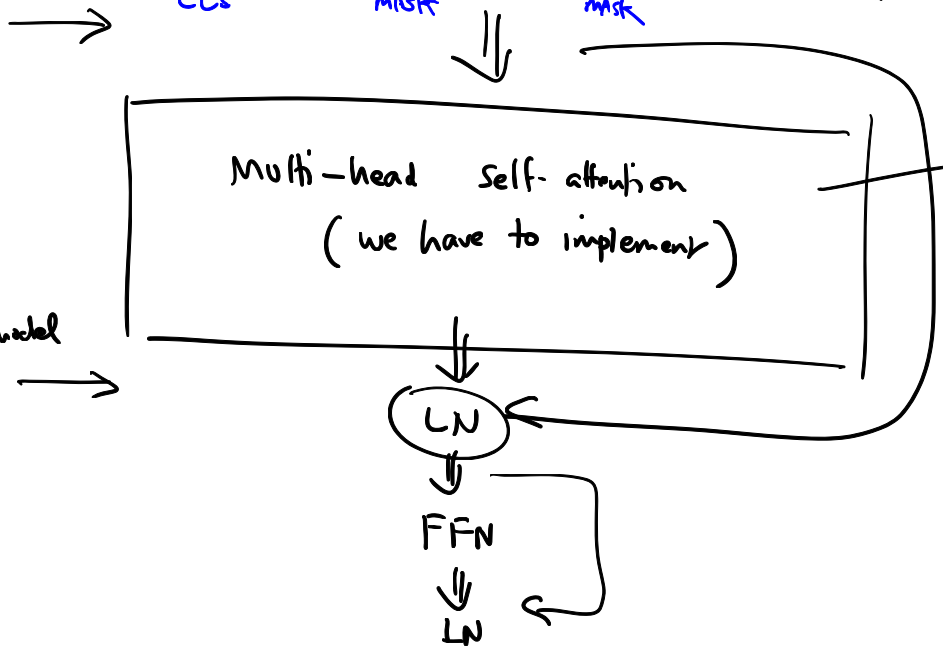


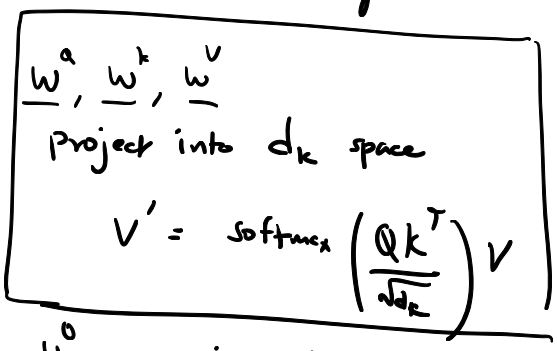
MAIGLA . .



$d_{model} = ?\ 128$

initial token embeddings





W^0 ← project back into d_{model}

h-heads
Concat
Use W^0

$d_k = d_q = d_v$

Transformer Block :

```
--init--()
    mhsa = MHSA_ner( )
    ln1 = LN( )
    ffn = FFN(. . .)
    ln2 = LN( )
--forward--(x)
    x' = mhsa(x)
    x'' = ln1(x + x')
    x''' = ffn(x'')
    x'''' = ln1(x''' + x'')
    return x''''
```

```
MHSA_ner( )
--init--
    for h = 0 ... 7
        A[h] = Attn_ner( )
    Wo = linear(h x d_k, d_model)
--forward--(x)
    for h = 0 ... 7
        V[h] = Attn_ner(x)   (Attn_ner = A[h])
    V = Concat(V[0] ... V[7])
    x = Wo(V)
```

```
Transformer(# layers)
    for l = 0 ... #layers
        TB[l] = Transformer Block(. .)
```

#layers = 1

Training code

$x_i \rightarrow$ [25] output

Define ProtBERT:
```
--init--
    Transformer( )
    Linear(128, 25)
    :
```

One sequence

$x = \langle x_1, x_2 \ldots x_L \rangle$

M = random binary mask

Y = Transformer(x) → [CLS] Y_1 Y_2 Y_3 . . . Y_L

Contextualized embeddings

p = prob of masking = 0.15

	CLS	x_1	x_2	x_3	. . .	x_L
M	0	0	1	0	. . .	1
$\bar{M}$	1	1	0	1	. . .	0

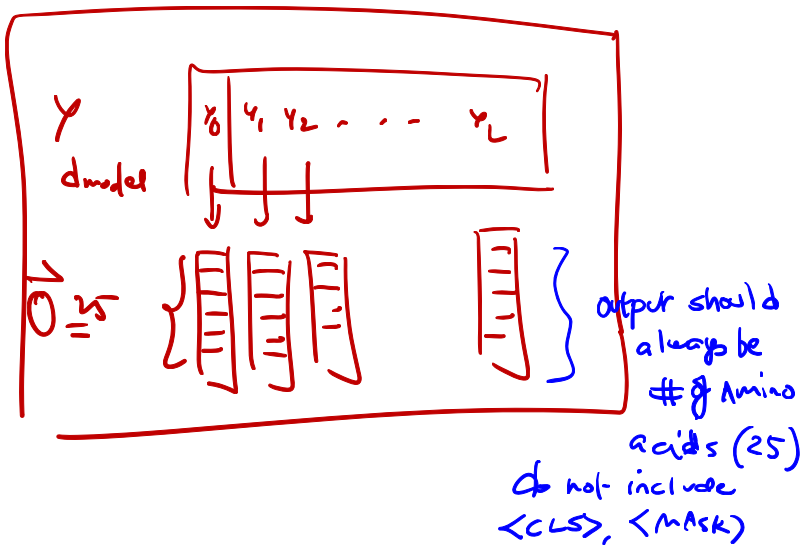
$x' = x \odot \bar{M}$

Instead of zeroing out masked positions, replace with <mask> token

input → Mask out x_2, x_L

$\vec{O} = \text{Transformer}(X')$

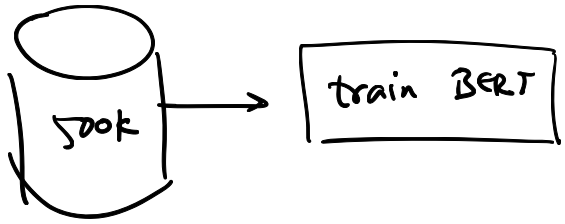
$(25 \times L)$ matrix



$O' =$ remove the $\emptyset$ positions from $\vec{O}$

25×2

$loss = CE\_with\_logits(O', \text{True label}, [x_2, x_L])$



pre-trained

---

down stream task

protein family classification

1 0

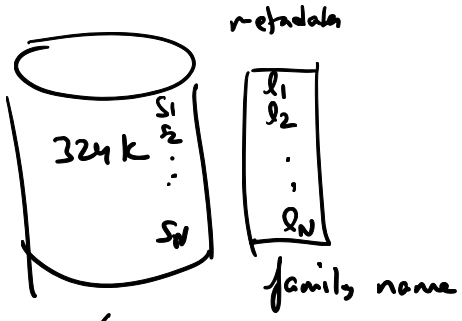
binary logistic regression

MLP

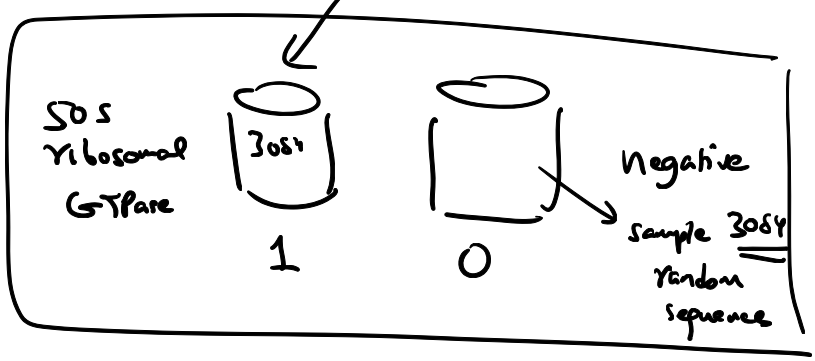
50S ribosomal GTPase

3054 instances

(+1)



50S DB



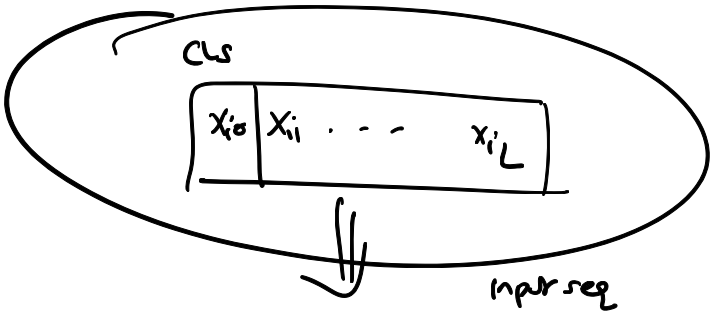
X = SOS DB → 305ex2 sequences of 1 labels

for $i = 1 \ldots 617$ B

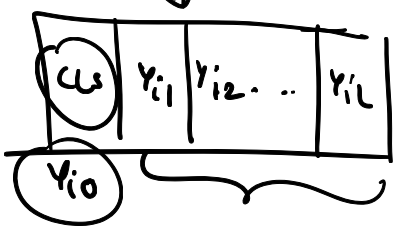
Y_i = Transformer(X_i) ← batch

Z_i = MLP($Y_{i\phi}$)

BCE loss (Z_i, true)



Transformer



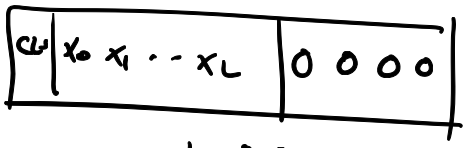
d_{model} = 128 dim

SOS DB

If a seq > 998

look at 1st 998 elements

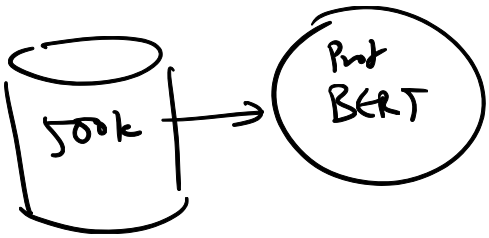
Zero-pad all blocks for transformer

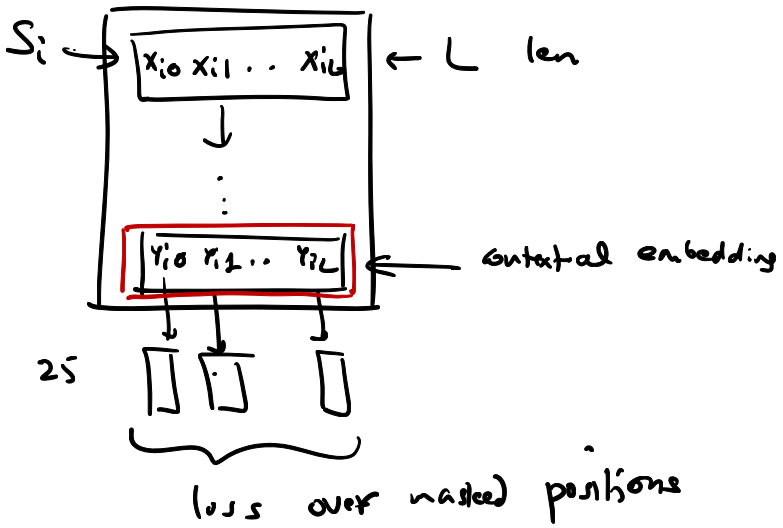


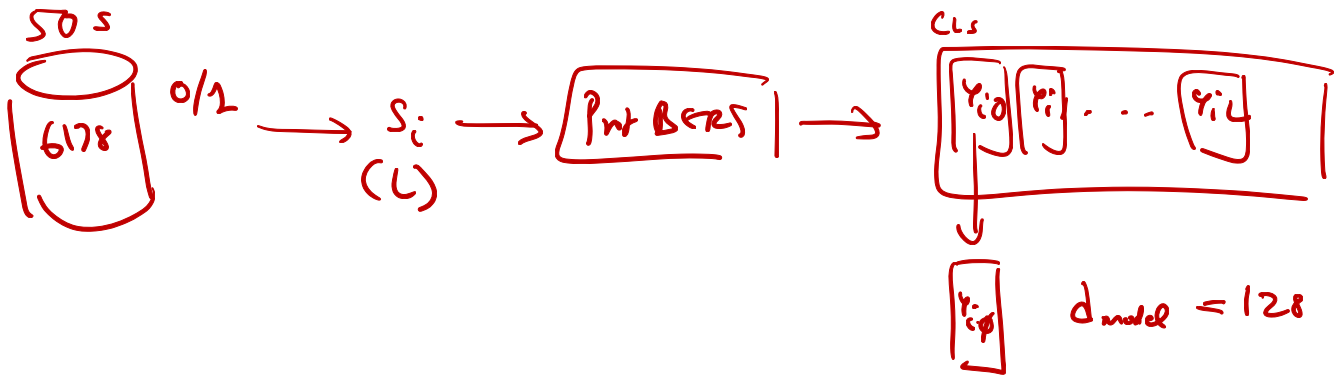
L = 300

You can use <PAD> token and init <PAD> to zero vector in embedding layer

---





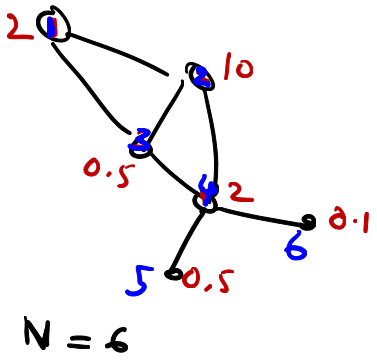


Graph Neural Networks

$L = D - A$ matrix

f_1 : feature vector $\in \mathbb{R}^N$

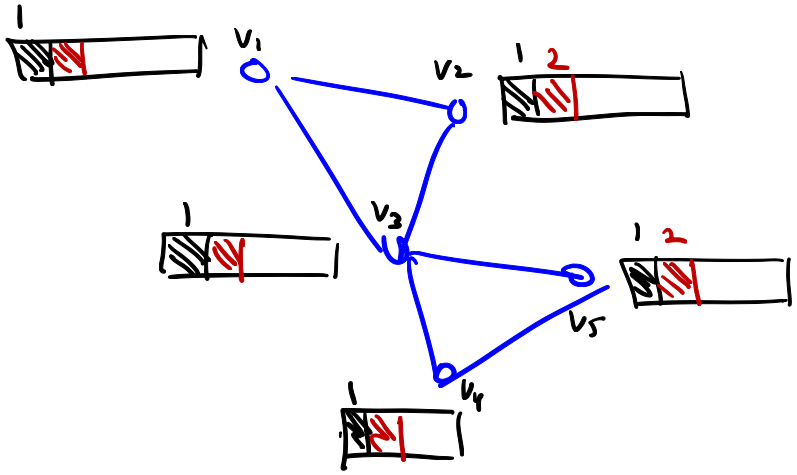
$f_1 = [2, 10, 0.5, 2, 0.5, 0.1]^T$

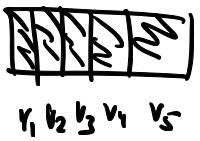


Smoothness = $f_1^T L f_1 = \sum_i \sum_{j \in N(i)} c_{ij} \left(f_1[i] - f_1[j] \right)^2$

↑

filter we need to learn



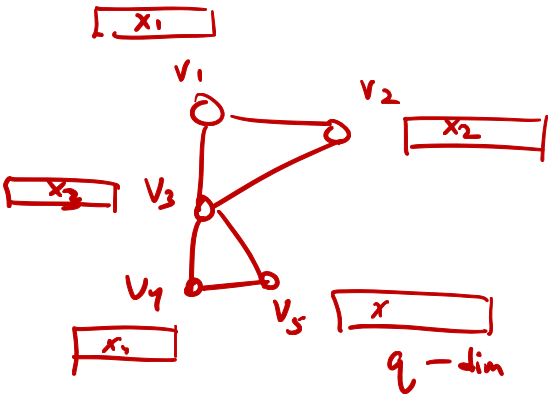
$f_1 =$ 

$f_2 =$ 

$$F^T = \begin{matrix} N \\ q \end{matrix} \begin{matrix} \overbrace{\quad\quad} \\ \end{matrix} \begin{matrix} f_1 \\ f_2 \\ \vdots \\ f_q \end{matrix}$$

q channels

GNN



$$F = \begin{matrix} & q \\ N & \end{matrix} \begin{matrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{matrix} \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{matrix}$$

x_i : feature vector for node v_i

q-dim

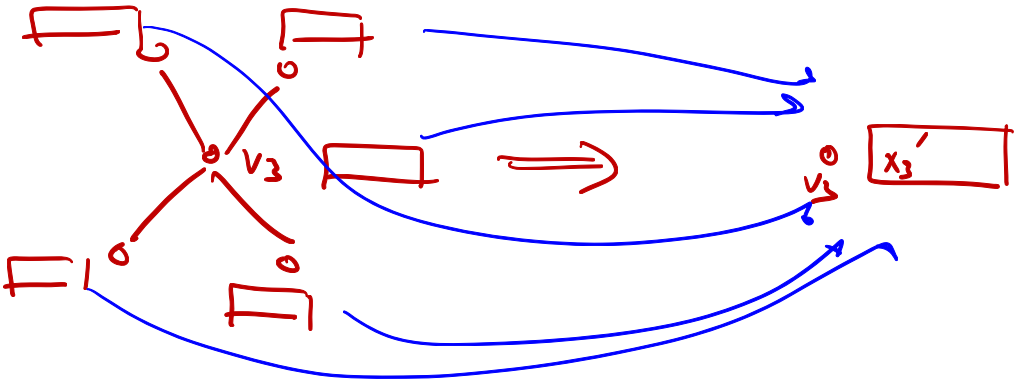
each layer update the node feature

a) spatial update / neighborhood-based (1-hop)

b) spectral update

GCN: graph convolutional network

$$x_i' = \sum_{j \in N(x_i)} \frac{1}{\sqrt{d_i \cdot d_j}} \vec{x}_j \; w_{ij} \qquad N'(x_i) = N(x_i) \cup \{x_i\}$$


v_3
v_3'
x_3'