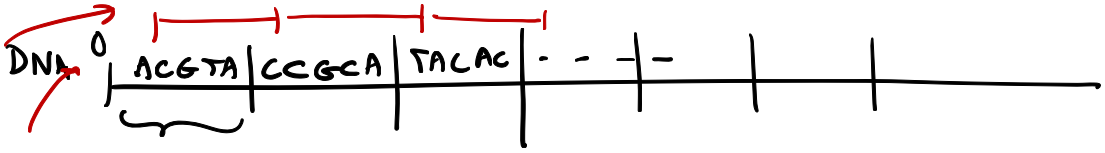
DNA/genome : 3 billion base pairs

Protein sequence : Uniprot ~ 100 million known sequence

k-mers
n-grams



k=5

offset

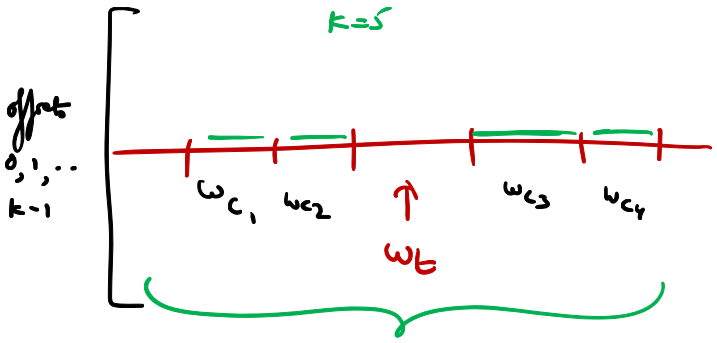
I) Need 3 ORF → 0, 1, 2

Open reading Frame ← transcription / translation

II) for k-mer

use all k offsets

k = 0, 1, 2, . . , k-1



$\forall$ offsets

$\forall$ possible windows of size q

a) Create positive example

$P(w_{c_i} | w_t) = \sigma(\vec{c_i}^T \vec{t})$

b) Create γ negative samples

$1 - P((w_{n_i}) | w_t)$

DNA Vocab : $k=5$

$4^5 = 2^{10} = 1024$

Practical : {A, C, G, T} ← Base Alphabet

{N: Unknown, R, S..} ← expanded Alphabet

Proteins

20 ← base alphabet

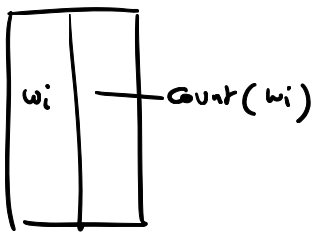
23-25 ← expanded protein alphabet

$k=5$ 20^5

Negative Sampling

Any random word from Vocab.

a) Create the vocab
↳ count while tokenizing



$$P(w_i) = \frac{count(w_i)^{\alpha}}{\sum_{j=1}^{|V|} count(w_j)^{\alpha}}$$

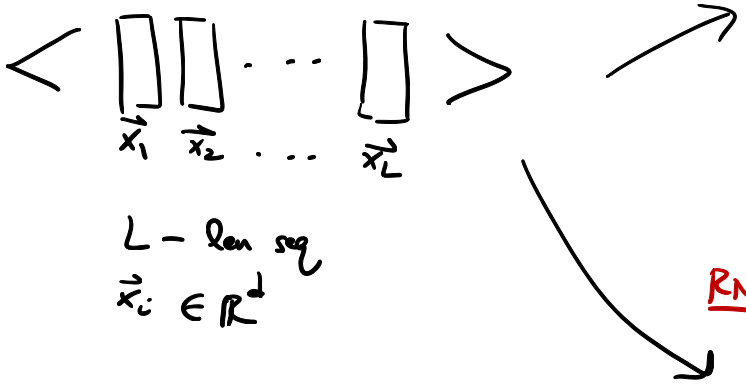
$\alpha = 0.75$ ← tamp down high freq words, bump up low freq words

$$CE := -\Big(\underbrace{\log P(w_{ci} \mid w_t)}_{1:+ve} + \underbrace{\sum_{j=1}^{r} \log\big(1 - P(w_{nj} \mid w_t)\big)}_{r:-ve} \Big)$$

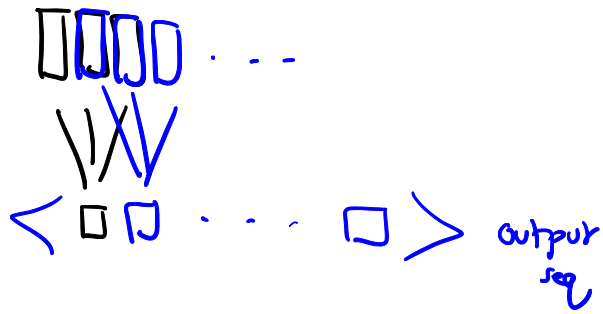
sum over all context words c_i in the window

Sum over **all** windows

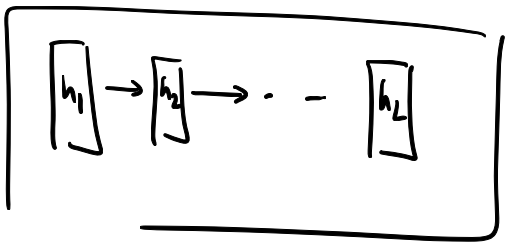
Self-Attention : Transformer



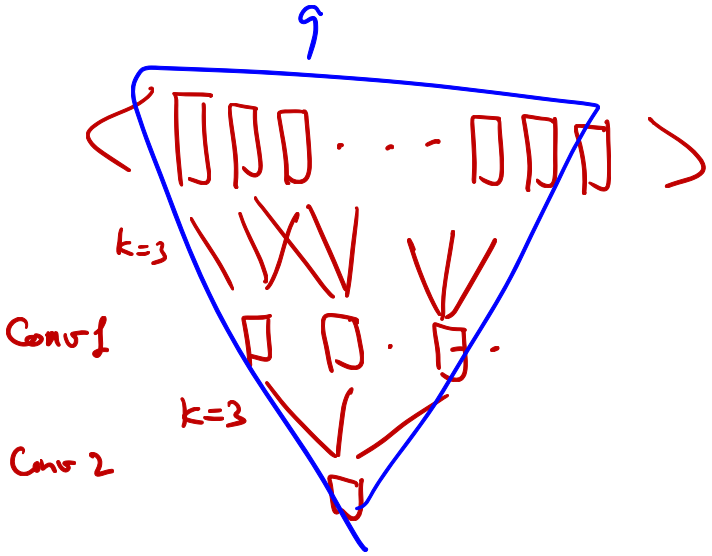
CNN: Kernel-size = 3

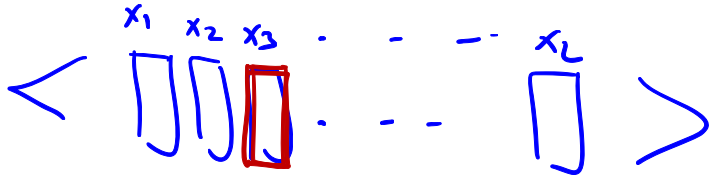


RNN:



CNN: hierarchical dependence





key / context

attention ($\vec{x}_j$ | $\vec{x}_3$) $j = 1, \ldots, L$ (including $j = 3$)

pair-wise

target / query

Compute for all elements as the target/query

$O(L^2)$ cost / Attention Matrix

---

word2vec Implementation

1) Create the vocab : V_K

k-mer word size

model ( ) :

init
- T: $V_K \times d$
- C: $V_K \times d$
- d: embed dim 300, 512, 768, 1024 ?

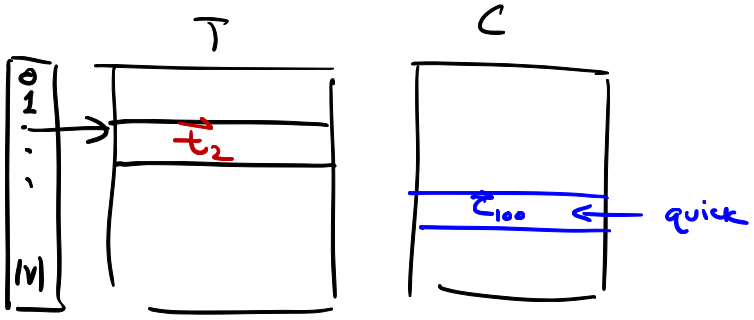
Parameter

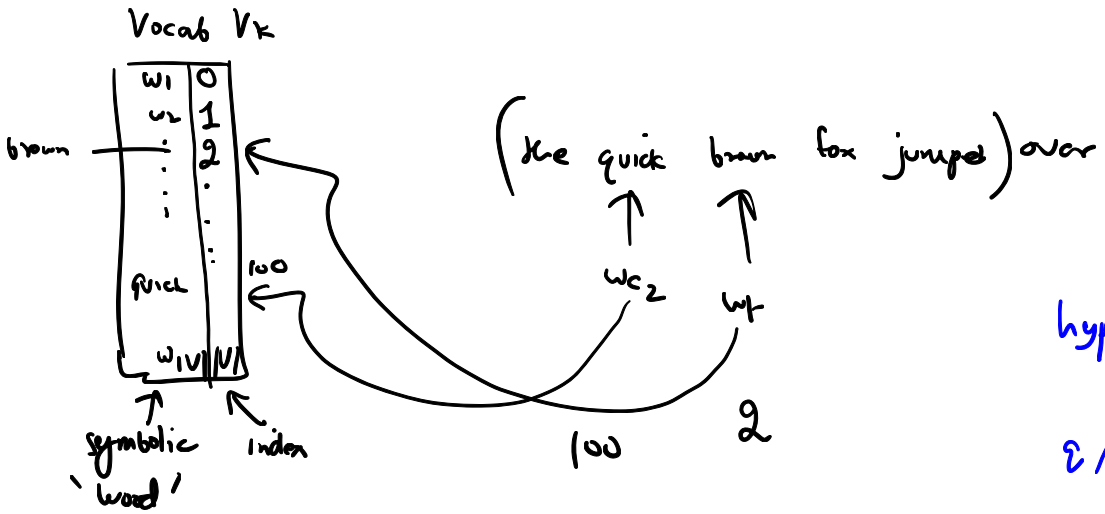
forward (y, x : BX)

index into the vocab

$\left(\frac{2}{w_t}, \frac{100}{w_{c'}} \right)$ $y = 1$

brown





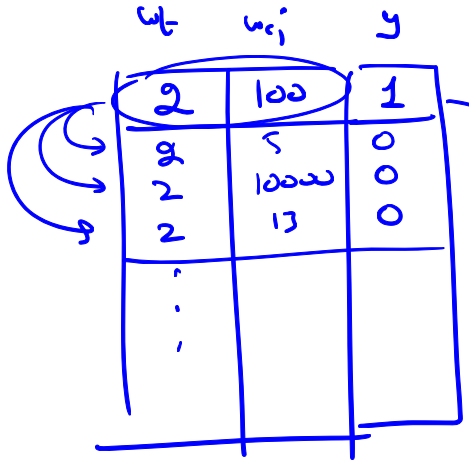
hyperparameters

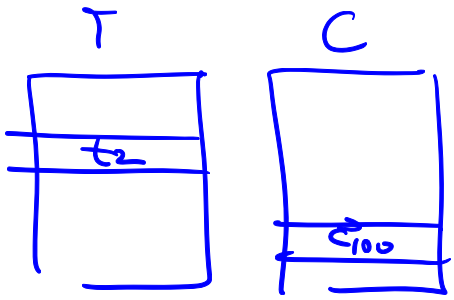
k: k-mer

ℓ/w: window size

r: negative sample

d: embed dim





forward(X, y):

X: B x 2

y: B x 1

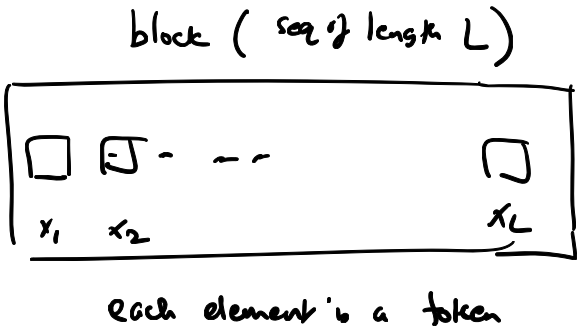
$\log\left(\sigma\left(\vec{t}_2^{\,T}\,\vec{c}_{100}\right)\right)$

a) with logits: just do the dot product!

b) σ / log

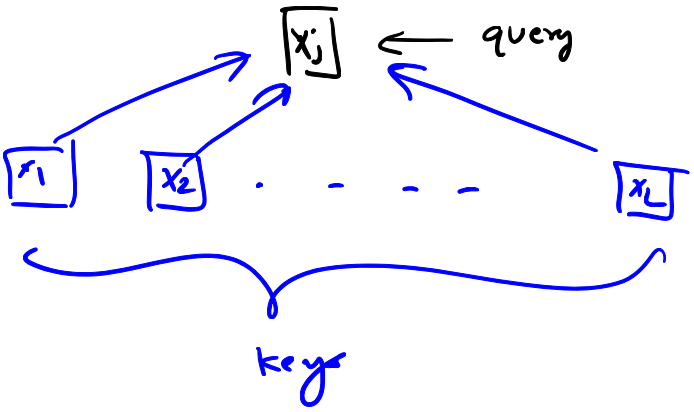
t-SNE to visualize in 2D.

Self-Attention



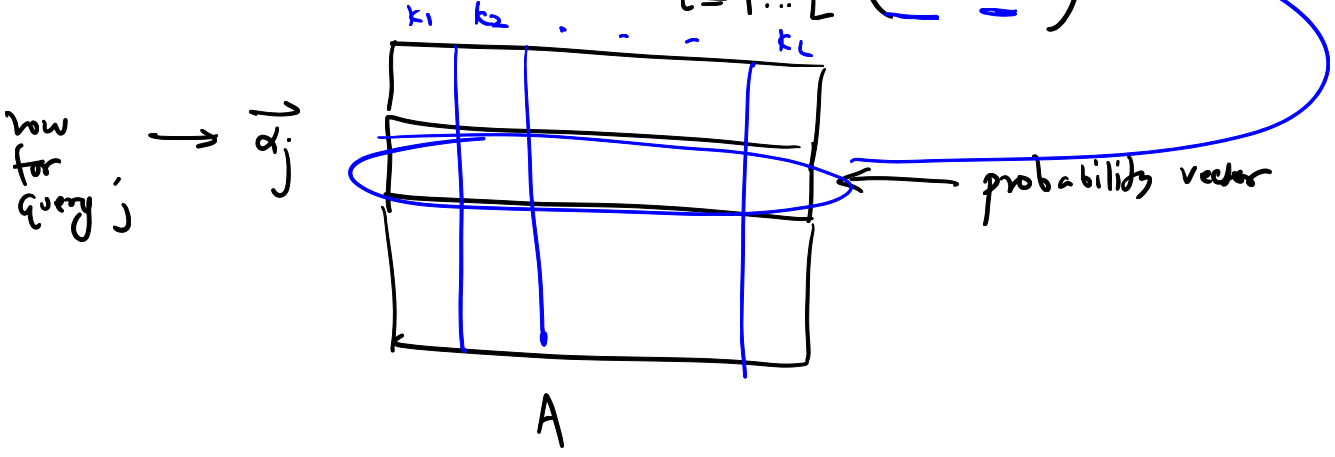
each element is a token

$$\alpha_{ji} \equiv \text{attention}\left(\text{key}_i \mid \text{query}_j\right)$$



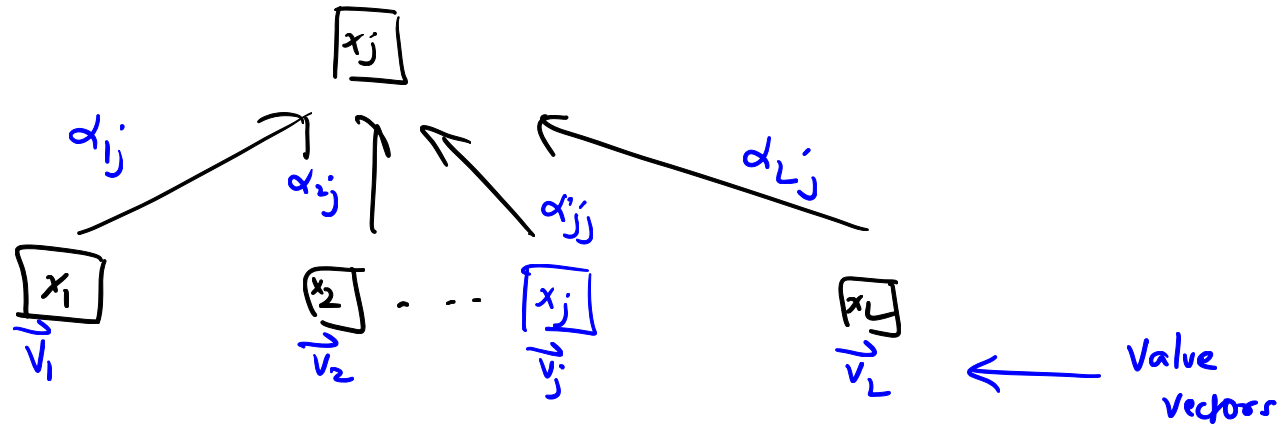
Key "attend to" query x_j

$$\vec{\alpha}_j = \text{Softmax}\left(\vec{q}_j^{\,T}\,\vec{k}_i\right), \quad i = 1 \dots L$$



A

Attention matrix



Output

$$\vec{v}_j = \sum_{i=1}^{L} \alpha_{ji} \vec{v}_i$$

weighted sum of value vectors

proportional to attention