Context Sensitive Memory

- key/value pairs
    - keys and values are *vectors* of length d
- like a Python dict

Key	Value
k_1	v_1
k_2	v_2

$\vdots$ $k_{\bar{T}}$ | $v_{\bar{T}}$

Match a query q_1 against each key

- query is a vector of length d (just like keys)
- $\text{score}(q_i, k_j)$is a measure of the similarity of query q_i and key k_j

Key	Value	Score(q,k)
k_1	v_1	$q_1 \cdot k_1$
k_2	v_2	$q_1 \cdot k_2$

$\vdots$ $k_{\bar{T}}$ | $v_{\bar{T}}$ | $q_1 \cdot k_{\bar{T}}$

Here we use dot product (cosine similarity) as our measure of similarity of query and key.

Normalize the scores: turn them into weights $\alpha(q, k)$ so that they sum to 100%

- using a Softmax to exaggerate the differences

Key	Value	Score(q,k)	$\alpha(q, k)$
k_1	v_1	$q_1 \cdot k_1$	$\mathrm{Norm}(q_1, k_1)$
k_2	v_2	$q_1 \cdot k_2$	$\mathrm{Norm}(q_1, k_2)$
$\vdots$			
$k_{\bar{T}}$	$v_{\bar{T}}$	$q_1 \cdot k_{\bar{T}}$	$\mathrm{Norm}(q_1, k_{\bar{T}})$

Lookup returns the weighted sum of values

$$\sum_i \alpha(q, k_i) * v_i$$

This is similar to a Python dict lookup except

- a value is *always* returned, even if there is no **exact** match of the query with any key
- if there is an exact match with one key: a single Score should be 100% and all others equal to 0%

We call this a *Soft lookup*

Multiple queries at once

We can execute many queries in parallel.

Let Q be a matrix of shape $(T \times d)$

- a collection of T queries
    - each a vector of length d, like the keys
- Let K be a matrix of shape $(\bar{T} \times d)$
    - the $\bar{T}$ keys in the CSM
- Let V be a matrix of shape $(\bar{T} \times d)$
    - the $\bar{T}$ values in the CSM

We can compute the score of each query againsts each key via matrix multiplication

$$Q * K^T$$

which has shape $(T \times \bar{T})$

- each query against the $\bar{T}$ keys

And the weighted (un-normalized) sum of values of all queries can be obtained by

$$Q * K^T * V$$

which has shape $(T \times d)$

- T query results, each of length d

In [2]:
```
print("Done")
```

```
Done
```