

CS 360: Machine Learning

Sara Mathieson, Sorelle Friedler

Spring 2024



Haverford
COLLEGE

Admin

- Thank you for the project proposals!
 - Will review in the order received
- **Lab 8** due April 18 (next Thurs)
- **Midterm April 25** in class
- **Project presentations**: last week of classes
- **Writeup** due by the end of finals period
 - May 11 for seniors
 - May 17 for non-seniors

Outline for April 9

- Recap RNNs and text generation
- Attention mechanisms
- Positional encoding (if time)

Outline for April 9

- Recap RNNs and text generation
- Attention mechanisms
- Positional encoding (if time)

Preprocessing for a language model

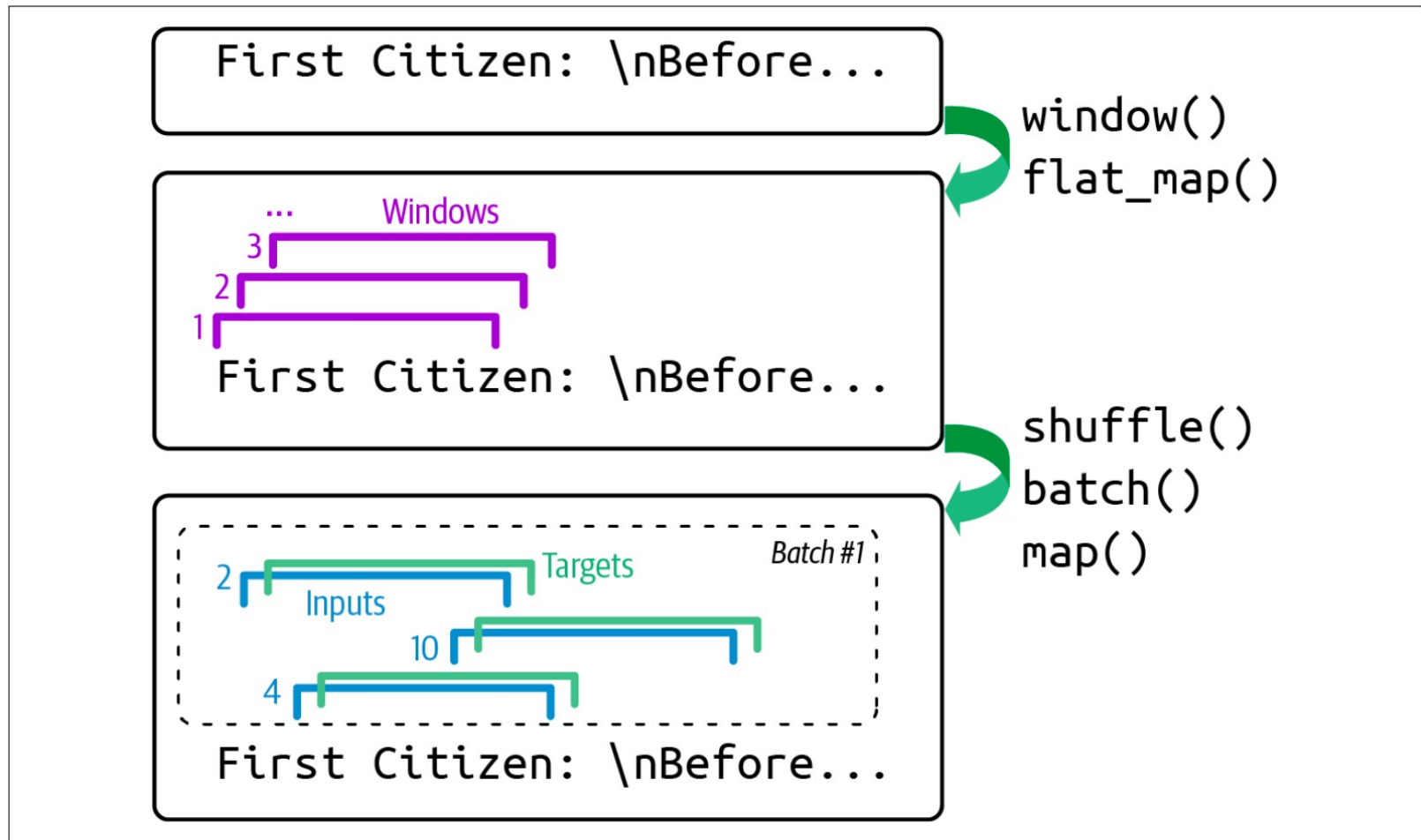


Figure 16-1. Preparing a dataset of shuffled windows

window length = 5

"to be"

5 chars long

input (x) = $\begin{matrix} t & o & b & e \\ [20, 15, 27, 2, 5] \end{matrix}$

target (y) = $[15, 27, 2, 5, 27]$
Space

logits

Softmax

v = vocab size

$$\text{Scores} = [s_1, s_2, \dots, s_v]$$

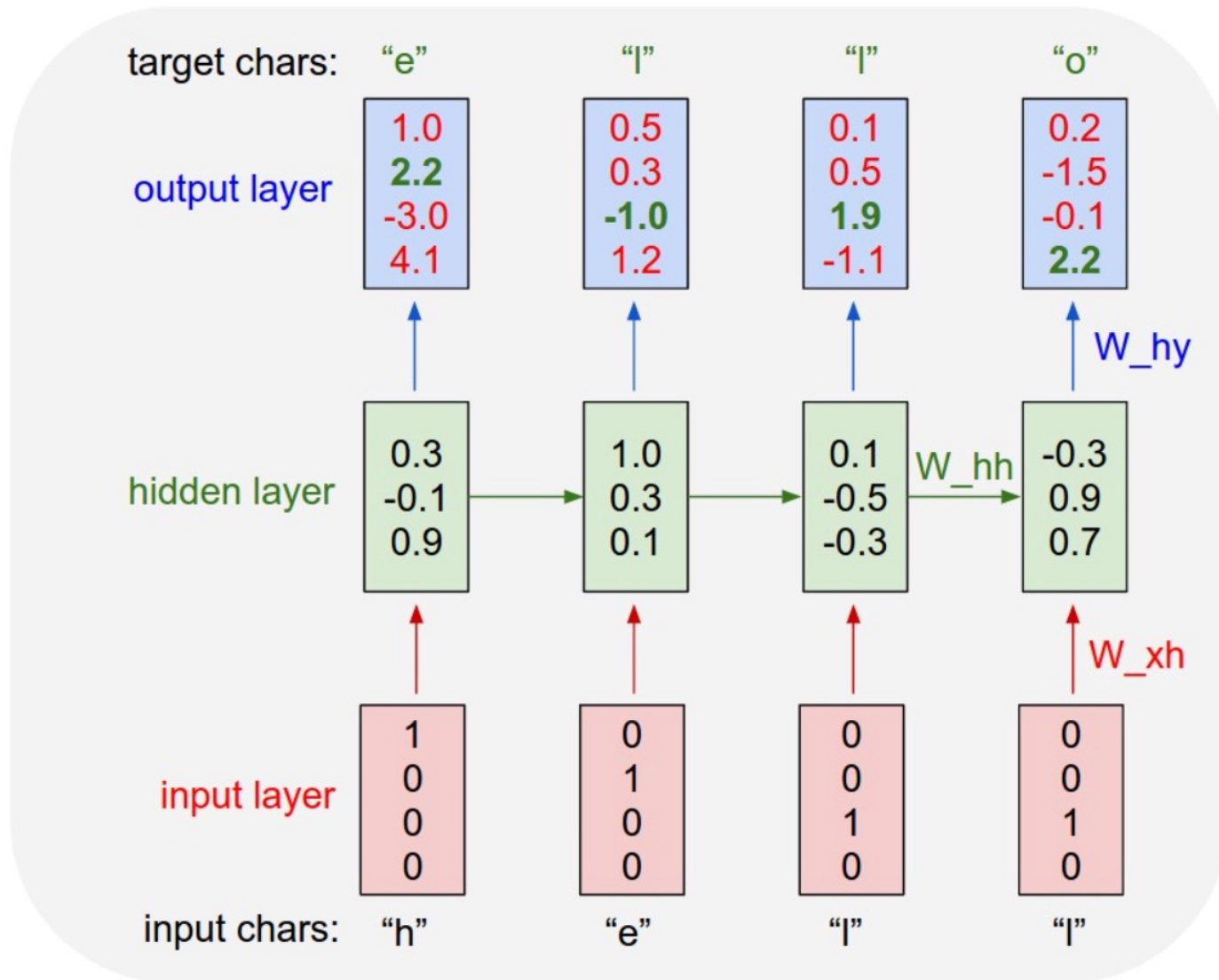
$0.2, -1.5, -0.1, 2.2$

$$\text{softmax}(s_i) = \frac{e^{s_i}}{\sum_{j=1}^v e^{s_j}}$$

h, e, l, o

Probability distribution

Example of input/output



Word Embeddings

- If we have 50,000 words and one-hot encoding, doesn't scale! (Very sparse matrix)
- Instead: embed in a lower dimension space

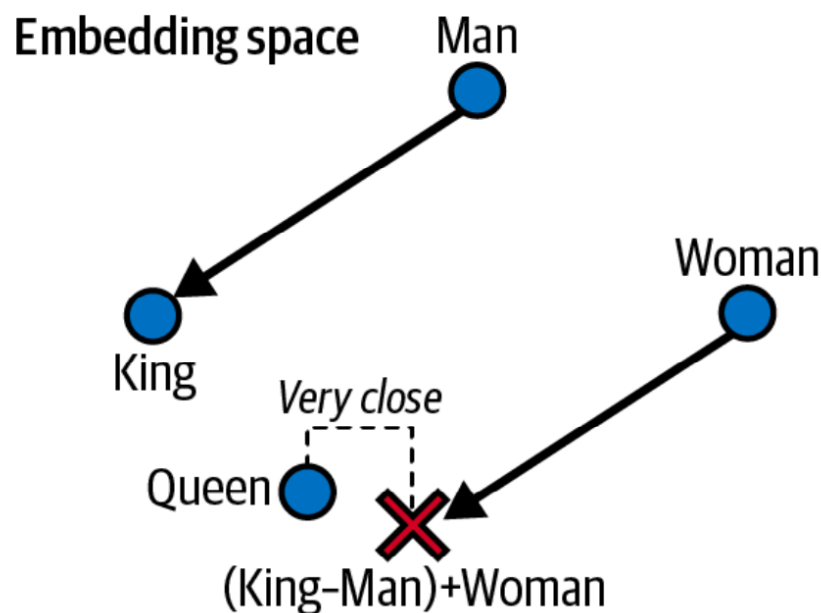


Figure 13-7. Word embeddings of similar words tend to be close, and some axes seem to encode meaningful concepts

Temperature: don't always pick the letter with maximum probability

```
>>> print(extend_text("To be or not to be", temperature=0.01))  
To be or not to be the duke  
as it is a proper strange death,  
and the
```

```
>>> print(extend_text("To be or not to be", temperature=1))  
To be or not to behold?  
  
second push:  
gremio, lord all, a sistermen,
```

```
>>> print(extend_text("To be or not to be", temperature=100))  
To be or not to bef ,mt'&o3fpadm!$  
wh!nse?bws3est--vgerdjw?c-y-ewznq
```

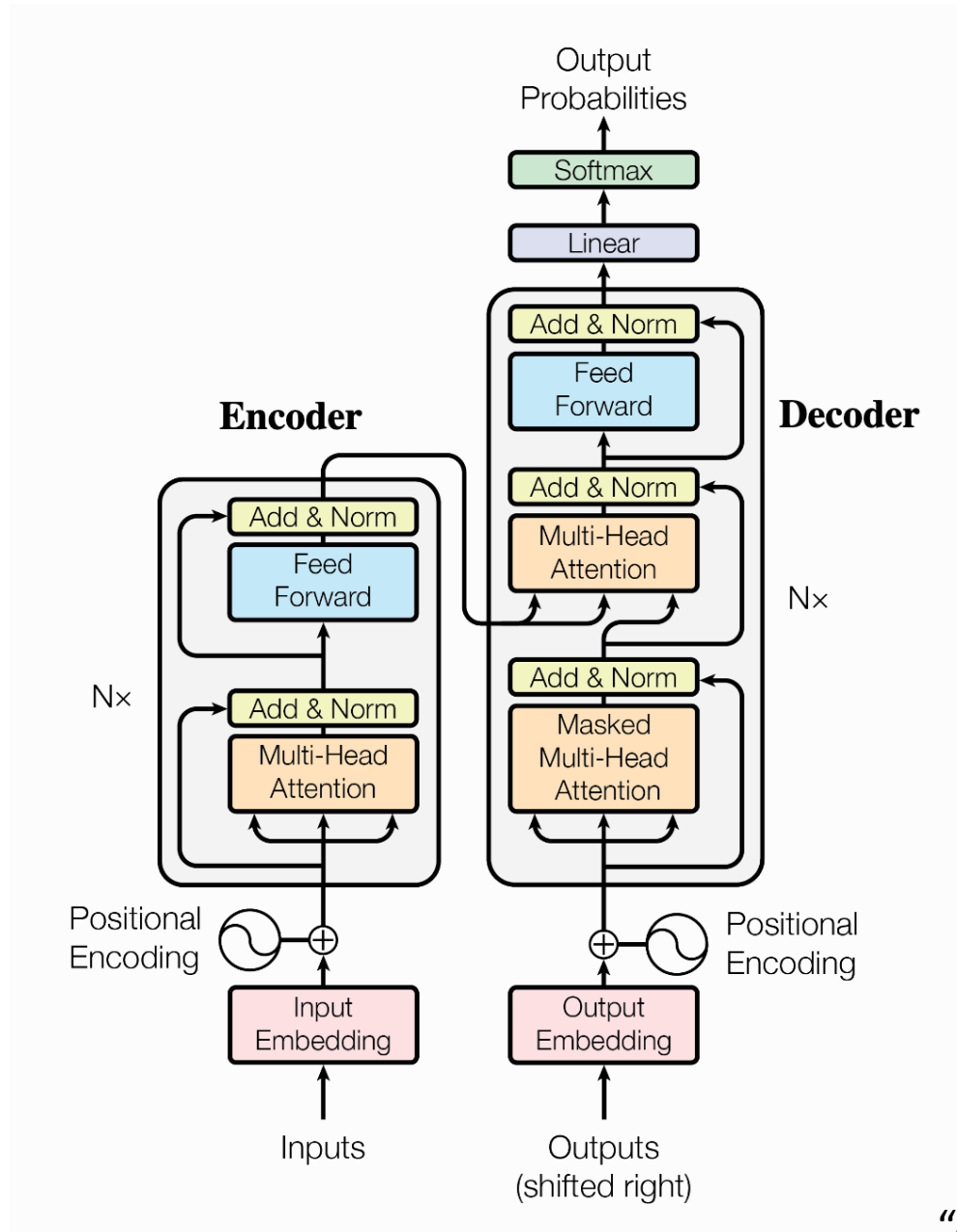
Handout 21, Q1

- Temperature = 1
 - Scores (logits) $[-0.6931472, -0.9162907, -2.3025851]$
 - Probabilities $[0.5, 0.4, 0.1]$
- Temperature = ~~0.01~~ **100 (corrected!)**
 - Scores (logits) $[-0.00693147, -0.00916291, -0.02302585]$
 - Probabilities $[0.33536728108, 0.3346197634, 0.3300129554]$
- Temperature = ~~100~~ **0.01 (corrected!)**
 - Scores (logits) $[-69.31472, -91.629074, -230.25851]$
 - Probabilities $[0.9999999, 2.0370382e-10, 1.26765211e-70]$

Outline for April 9

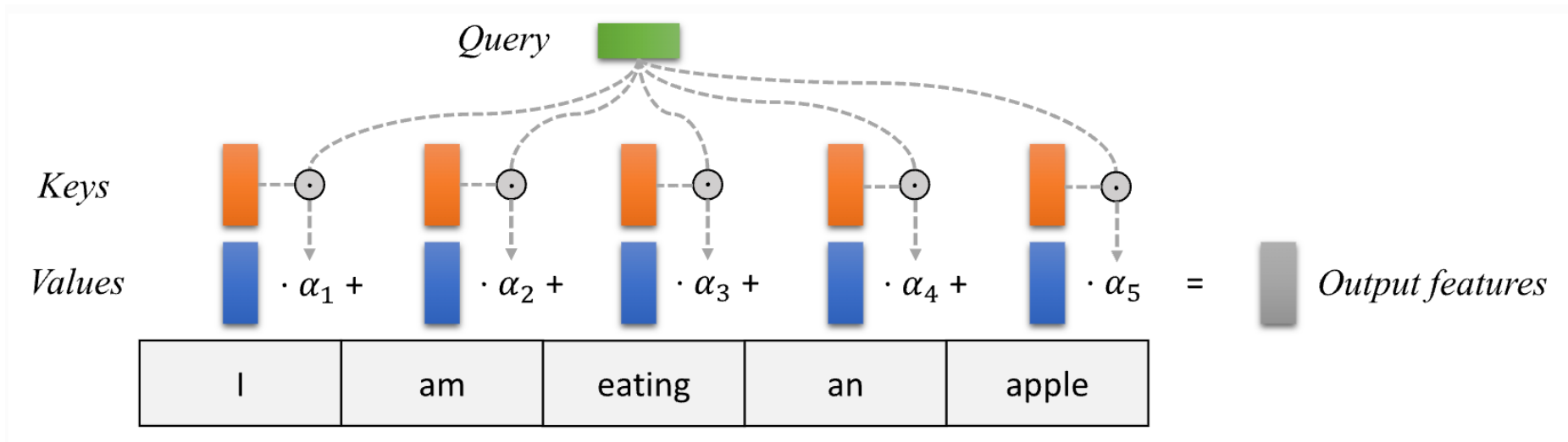
- Recap RNNs and text generation
- **Attention mechanisms**
- Positional encoding (if time)

Transformer Architecture



“Attention is all you need”

Attention mechanisms



$$\alpha_i = \frac{\exp(f_{\text{attn}}(\text{key}_i, \text{query}))}{\sum_j \exp(f_{\text{attn}}(\text{key}_j, \text{query}))}, \quad \text{out} = \sum_i \alpha_i \cdot \text{value}_i$$

Attention

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_{\text{keys}}}} \right) V$$

Q: queries (specific word)

K: keys (all the other words)

V: value (self attention: transformation of input)

$$\frac{QK^T}{\sqrt{d_{\text{keys}}}}$$

dimension of keys

Back to kernels

Say we have inputs + outputs

$(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_t, y_t)$
goal: predict \hat{y} for new pt \vec{x}_0
test example

$K(\vec{u}, \vec{v})$ kernel function to measure similarity between \vec{u} & \vec{v}
 \Rightarrow "nearest neighbors"

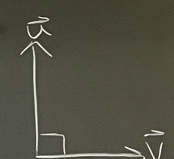
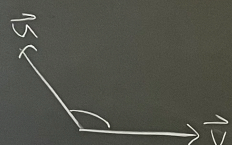
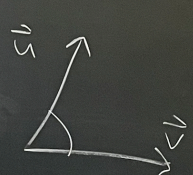
$$\hat{Y} = \sum_{i=1}^t y_i \frac{K(\vec{x}_i, \vec{x}_0)}{\sum_{j=1}^t K(\vec{x}_j, \vec{x}_0)}$$

kernel smoothing

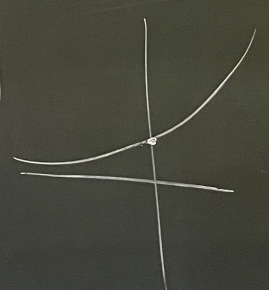
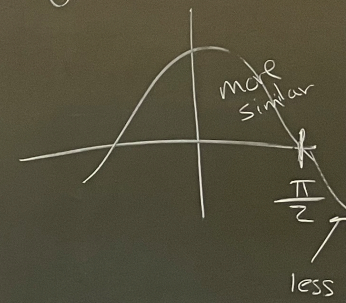
} similarity between test & each train
 } normalize

example

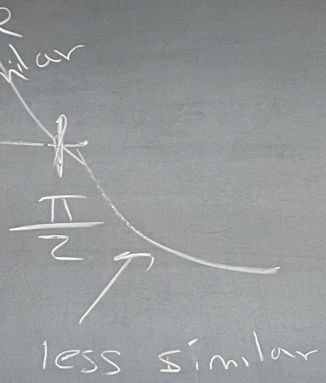
$K(\vec{u}, \vec{v}) = \exp(\vec{u} \cdot \vec{v})$

- if  $\vec{u} \cdot \vec{v} = 0$
 $\exp(0) = 1$
- if  $\vec{u} \cdot \vec{v} < 0$
 $\exp(-) < 1$
- if  $\vec{u} \cdot \vec{v} > 0$
 $\exp(+)> 1$

$\vec{u} \cdot \vec{v} = \|\vec{u}\| \|\vec{v}\| \cos(\theta)$



$$\|\vec{v}\| \cos \theta$$



$$K(\vec{u}, \vec{v}) = \exp((w_1 \vec{u}) \cdot (w_2 \vec{v}))$$

weight matrices

\vec{x}_0 = word that has been embedded (query)

\vec{x}_i = another word in the sentence (keys)

\vec{y}_i = next word (target) (after \vec{x}_0) (values)

attention maps

Window = 3

$$\text{Softmax} \rightarrow \begin{bmatrix} 0 & -\infty & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 \end{bmatrix}$$

Second char can only "pay attention" to the first char

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.5 & 0 \\ 0.33 & 0.33 & 0.33 \end{bmatrix}$$

$$\frac{e^0}{e^0 + e^{-\infty} + e^{-\infty}} = \frac{1}{1+0+0}$$

Handout, Q2

attention
q

