NLP Sheet 2

- Missing from Sheet 1

5. Mention two words that have

		lemma	Stem (ed, s, ing, tion, or, ...)
Same lemmas and Same Stems	Played	Play	Play
	Playing	Play	Play
Same lemmas and different Stems	lie	lie	lie
	lied	lie	li
different lemmas and Same Stems	action	action	ac
	aced	ace	ac
diffent lemmas & different Stems	walk	walk	walk
	run	run	run

1. Write the equation for trigram probability estimation

$$P(w_i|w_{1:i-1}) \simeq P(w_i|w_{i-2}w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})}$$

N-gram → N-1 words

then write all nonzero probabilities for
3-gram

<S><S> I am Sam </S> <S><S> Sam I am </S>
<S><S> I do not like green eggs and ham </S>

- $P(I|<S><S>) = \frac{2}{3}$ ← C(<S><S>I) / ← C(<S><S>)

• Count denominator 1st.

$P(am|<S>I) = \frac{1}{2}$ ← C(<S>I am) / ← C(<S>I)

$P(Sam|I\,am) = \frac{1}{2}$

$P(</S>|am\,Sam) = \frac{1}{1}$

$P(<S>|Sam</S>) = \frac{1}{1}$

$P(<S>|</S><S>) = \frac{2}{2}$

$P(Sam|<S><S>) = \frac{1}{3}$

$P(I|<S>Sam) = \frac{1}{1}$

$P(am|Sam\,I) = \frac{1}{1}$

$P(</S>|I\,am) = \frac{1}{2}$

$P(<S>|am</S>) = \frac{1}{1}$

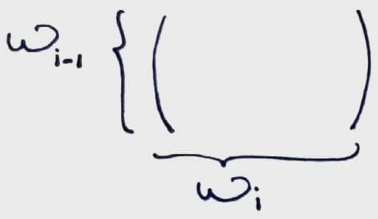
$P(<S>|</S><S>) = \frac{2}{2}$

* & more probabilities, all equal to 1

...

- Note that by writing the Probabilities in that way, all of them are guaranteed to be nonzero (the other approach is to build a count matrix with rows being all consecutive word pairs (unique) and columns being all words (unique))

2. Given is a bigram Probability matrix



- normal bigram and add-1 smoothed version.

- Compute Probability of Sentence "I want Chinese Food"

→ <S> I want Chinese Food </S>

Recall, $P(w_{1:n}) = \prod_{i=1}^{N} P(w_i|w_{i-1})$

- $i=1 \leftrightarrow$ (I) after <S>

a) using normal Bigram Probabilities:

$$P(w_{1:n}) = P(I|<S>)\,P(want|I)\,P(Chinese|want)\,P(Food|Chinese)\,P(</S>|Food)$$

$$= 0.25 \times 0.33 \times 0.0065 \times 0.52 \times 0.68$$
$$= 0.00189$$

b) using add-1 Smoothed Probabilities:

$$= 0.19 \times 0.21 \times 0.0029 \times 0.052 \times 0.4$$
$$= 0.0000024$$

c) unsmoothed is higher because Smoothing moves Probability mass from events with nonzero Probability

to events of zero probability. This is evident by how it adds 1 in the numerator and $|V|$ in the denominator: this decreases any fraction with a nonzero denominator as $|V| > 1$.

↳ vocab. length

3. Train a bigram model on the following corpus without adding an end of sentence token

<s> a b
<s> b b
<s> b a
<s> a a

* Recall that training means to compute all the needed bigram probabilities.

$w_{i-1} = (\;\;)$ • For all possible w_{i-1}, w_i

w_i

w_{i-1}	a	b	Sum w_{i-1} →
<s>	2/4	2/4	4
a	1/2	1/2	2
b	1/2	1/2	2

• First write the count in black ($w_{i-1} w_i$)
• Sum for the count w_{i-1} (red)
• divide to get Probs

⇒ Find the sum of the probabilities of all possible two-word sequences

→ They are aa, ab, ba, bb

$$P(\langle s\rangle aa) = P(a|\langle s\rangle)\,P(a|a) = \frac{2}{4} \times \frac{1}{2} = \frac{1}{4}$$

$$P(\langle s\rangle ab) = P(a|\langle s\rangle)\,P(b|a) = \frac{2}{4} \times \frac{1}{2} = \frac{1}{4}$$

$$P(\langle s\rangle ba) = P(b|\langle s\rangle)\,P(a|b) = \frac{2}{4} \times \frac{1}{2} = \frac{1}{4}$$

$$P(\langle s\rangle bb) = P(b|\langle s\rangle)\,P(b|b) = \frac{2}{4} \times \frac{1}{2} = \frac{1}{4}$$

→ Their Sum is clearly 1

⇒ The Sum of Probabilities of all Possible 3-word Sequences

$$P\begin{pmatrix} \langle s\rangle\, a\, b\, a \\ \langle s\rangle\, b\, a\, a \\ \langle s\rangle\, a\, a\, a \\ \langle s\rangle\, b\, b\, a \\ \langle s\rangle\, a\, b\, b \\ \langle s\rangle\, b\, a\, b \\ \langle s\rangle\, a\, a\, b \\ \langle s\rangle\, b\, b\, b \end{pmatrix} = \begin{pmatrix} 1/8 \\ 1/8 \\ 1/8 \\ 1/8 \\ 1/8 \\ 1/8 \\ 1/8 \\ 1/8 \end{pmatrix} \xrightarrow{\text{Sums to}} 1$$

↳ each is a Product of 3 Probabilites, each being = to half.

* Conclusion: the Bigram model makes all Sequences of a given length have Probabilities that Sum to 1
  - → That is, it doesn't assign a Single Probability distribution across all Sentences.
    - i.e., when we Compute the Probability of a Sentence, we don't get its true Probability across all Possible Sentences.
  - ⇒ we want the distribution across all Sentences to be 1 for this
  - * This is the role of the </s> token
    - → once we add it the matrix becomes

	a	b	</s>
<s>	2/4	2/4	0
a	1/4	1/4	2/4
b	1/4	1/4	2/4

and you can show now that sum of Probabilities of all Possible Sentences is 1

	one word	two words	three words	...
P	0.5	0.25	0.125	

$$\sum_{n=1}^{\infty} \frac{1}{2^n} = 1$$

4. We are given the following corpus

<S> I am Sam </S>
<S> Sam I am </S>
<S> I am Sam </S>
<S> I do not like green eggs and Sam </S>

- the formula we already know $C(w_{i-(N-1):i-1} w_i) / C(w_{i-(n-1):i-1})$ is a maximum likelihood estimate.

* Use linear interpolation between a bigram and unigram model to compute P(Sam|am)

$$\hat{P}(Sam|am) = \lambda_1 P(Sam|am) + \lambda_2 P(Sam)$$

- given $\lambda_1 = \lambda_2 = \frac{1}{2}$

$P(Sam|am)$: $\frac{2}{3}$ ← C(am)

$P(Sam)$: $\frac{4}{25}$ ← all words

$$= 0.413$$

5. Given a training Set of 100 numbers
→ 91 of which are Zeros
→ 9 other digits are 1, 2, 3, ..., 9

- What is unigram Perplexity for the test Set 0 0 0 0 0 3 0 0 0 0

⇒ From the training Set, it's true that

$$P(0) = \frac{91}{100}, \quad P(3) = \frac{1}{100}$$

- $P(w_{1:N}) = \prod_{i=1}^{N} P(w_i) = \left(\frac{91}{100}\right)^9 \cdot \frac{1}{100}$
  ↳ test Set

$$= \frac{91^9}{100^{10}}$$

$$PP(W) = \sqrt[N]{\frac{1}{P(W_{1:N})}} = \sqrt[10]{\left(\frac{91^9}{100^{10}}\right)^{-1}}$$

$x^{-1} = \frac{1}{x}$

$$= 1.725$$

6. Write the Feed Forward equations of an RNN

$$h_t = P(Wx_t + Uh_{t-1})$$

$$y_t = g(Vh_t)$$

- In this example, $W = V = U = 1$ (hence x_t and h_t are scalars), $P(x) = x$ and $g(x) = x$ (linear units)

The equations, hence, reduce to

$$h_t = x_t + h_{t-1}$$
$$y_t = h_t$$

→ given $X = [2, -0.5, 1]$ (x_0, x_1, x_2) find all network values. (h_t, y_t for $t \in \{0,1,2\}$)

- assume $h_{-1} = 0$

$x_0 = 2 \rightarrow h_0 = 2 + 0 \rightarrow y_0 = 2$

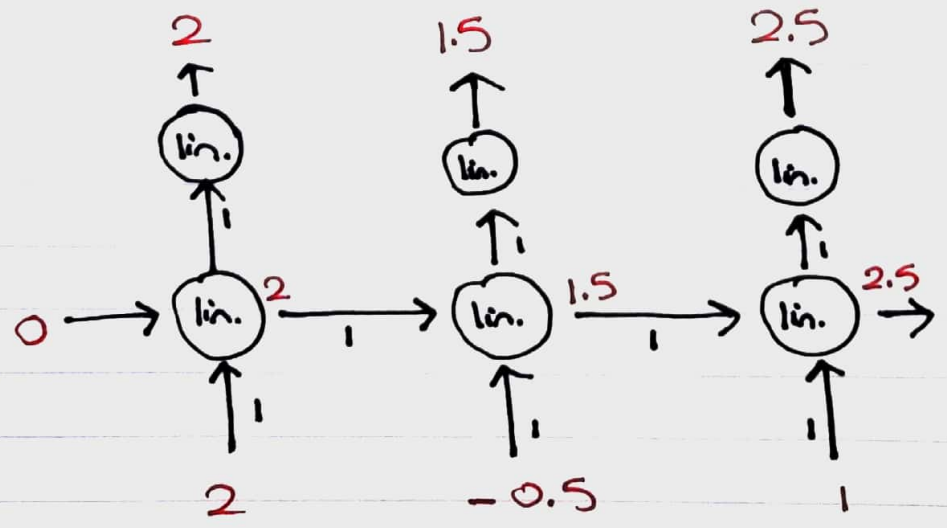
$x_1 = -0.5 \rightarrow h_1 = -0.5 + 2 = 1.5 \rightarrow y_0 = 1.5$

$x_2 = 1 \rightarrow h_2 = 1 + 1.5 = 2.5 \rightarrow y_0 = 2.5$

$2 + -0.5 \rightarrow 1.5 + 1 \rightarrow 2.5$

It's learning addition (in fact, the equations by definition describe a running sum)

Unrolled Network



7.

→ given is an RNN char. level language model. (token is a char.)

Vocab = {h, e, l, o}

→ each token is represented by a one-hot vector

- Use Softmax to compute final output

recall, $P(x) = \frac{e^{x}}{\sum_j e^{x_j}}$ ← element-wise is the Softmax fun.

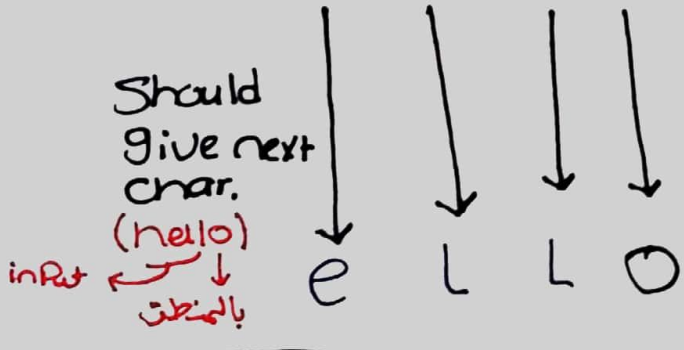
↑ vector

- Check lec. for more

→ Use it to compute the probability dist. vector by applying it on each of the 4 output vectors

e.g. $P\begin{pmatrix}1.0\\2.2\\-3.0\\4.1\end{pmatrix} = \begin{pmatrix}e^{1}\\e^{2.2}\\e^{-3.0}\\e^{4.1}\end{pmatrix} \cdot \frac{1}{e^{1}+e^{2.2}+e^{-3}+e^{4.1}}$

- the given input is h e l l



recall that the output is a dist. over the vocab.

what the model gave	O	O	L	O
	x	x	✓	✓

(Prediction is largest index in output vector) ( ) h e l o

→ before or after Softmax, it's the same