Conditional Language Modeling

Chris Dyer
Unconditional LMs

A language model assigns probabilities to sequences of words, \( w = (w_1, w_2, \ldots, w_\ell) \).

It is convenient to decompose this probability using the chain rule, as follows:

\[
p(w) = p(w_1) \times p(w_2 \mid w_1) \times p(w_3 \mid w_1, w_2) \times \cdots \times p(w_\ell \mid w_1, \ldots, w_\ell-1)
\]

\[
= \prod_{t=1}^{\mid w \mid} p(w_t \mid w_1, \ldots, w_{t-1})
\]

This reduces the language modeling problem to \text{modeling the probability of the next word}, given the \text{history} of preceding words.
Evaluating unconditional LMs

How good is our unconditional language model?

1. **Held-out** per-word **cross entropy** or **perplexity**

   \[ H = -\frac{1}{|w|} \sum_{i=1}^{|w|} \log_2 p(w_i \mid w_{<i}) \]  
   (units: bits per word)

   \[ \text{ppl} = b^{-\frac{1}{|w|}} \sum_{i=1}^{|w|} \log_b p(w_i \mid w_{<i}) \]  
   (units: uncertainty per word)

   Same as training criterion. *How uncertain is the model at each time position, an average?*

2. **Task-based evaluation**

   Use in a task-model that uses a language model in place of some other language model. Does it improve?
History-based LMs

A common strategy is to make a Markov assumption, which is a conditional independence assumption.

\[ p(w) = p(w_1) \times p(w_2 \mid w_1) \times p(w_3 \mid w_1, w_2) \times p(w_4 \mid w_1, w_2, w_3) \times \ldots \]
History-based LMs

A common strategy is to make a Markov assumption, which is a conditional independence assumption.

\[ p(w) = p(w_1) \times p(w_2 \mid w_1) \times p(w_3 \mid w_1, w_2) \times p(w_4 \mid w_1, w_2, w_3) \times \ldots \]

Markov: forget the distant past.

Is this valid for language? No…

Is it practical? Often!
History-based LMs

A common strategy is to make a **Markov assumption**, which is a conditional independence assumption.

\[
p(w) = p(w_1) \times p(w_2 | w_1) \times p(w_3 | w_1, w_2) \times p(w_4 | w_1, w_2, w_3) \times \ldots
\]

Markov: forget the distant past.

Is this valid for language? No…

Is it practical? Often!

Why RNNs are great for language: no more Markov assumptions!
History-based LMs with RNNs

\[
p(W_5 | w_1, w_2, w_3, w_4)
\]

random variable

RNN hidden state

vector, length=|vocab|

softmax

h_0 \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow h_4

w_1 \rightarrow w_2 \rightarrow w_3 \rightarrow w_4

observed context word

vector (word embedding)
History-based LMs with RNNs

\[
p(W_5 | w_1, w_2, w_3, w_4)
\]

RNN hidden state

observed context word

vector (word embedding)

vector, length = |vocab|

softmax

random variable
Distributions over words

Each dimension corresponds to a word in a closed vocabulary, $V$.

$$u = Wh + b$$

$$p_i = \frac{\exp u_i}{\sum_j \exp u_j}$$

The $p_i$'s form a distribution, i.e.

$$p_i > 0 \ \forall i, \ \sum_i p_i = 1$$

To enforce this stochastic constraint, we suggest a normalised exponential output non-linearity,

$$o_j = e^{I_j} / \sum_k e^{I_k}.$$  

This “softmax” function is a generalisation of the logistic to multiple inputs. It also generalises maximum picking, or “Winner-Take-All”, in the sense that that the outputs change smoothly, and equal inputs produce equal outputs. Although it looks rather cumbersome, and perhaps not really in the spirit of neural networks, those familiar with Markov random fields or statistical mechanics will know that it has convenient mathematical properties. Circuit designers will enjoy the simple transistor circuit which implements it.

Bridle. (1990) Probabilistic interpretation of feedforward classification...
Distributions over words

\[ u = Wh + b \]

\[ p_i = \frac{\exp u_i}{\sum_j \exp u_j} \]

\[ p(w) = p(w_1) \times \\
\quad p(w_2 | w_1) \times \\
\quad p(w_3 | w_1, w_2) \times \\
\quad p(w_4 | w_1, w_2, w_3) \times \\
\quad \ldots \]
Distributions over words

\[ u = Wh + b \]

\[ p_i = \frac{\exp u_i}{\sum_j \exp u_j} \]

\[ p(w) = p(w_1) \times p(w_2 | w_1) \times p(w_3 | w_1, w_2) \times p(w_4 | w_1, w_2, w_3) \times \ldots \]

histories are sequences of words...
Distributions over words

\[ u = Wh + b \]

\[ p_i = \frac{\exp u_i}{\sum_j \exp u_j} \]

**What are the dimensions of \( b \)?**

\[ h \in \mathbb{R}^d \]
\[ |V| = 100,000 \]

histories are sequences of words…

\[ p(w) = p(w_1) \times \\
  p(w_2 | w_1) \times \\
  p(w_3 | w_1, w_2) \times \\
  p(w_4 | w_1, w_2, w_3) \times \]

...
Distributions over words

$$h \in \mathbb{R}^d$$

$|V| = 100,000$

What are the dimensions of $W$?

$$p(w) = p(w_1) \times$$

$$p(w_2 \mid w_1) \times$$

$$p(w_3 \mid w_1, w_2) \times$$

$$p(w_4 \mid w_1, w_2, w_3) \times$$

...
RNN language models
RNN language models
RNN language models
RNN language models

\[ p(tom \mid \langle s \rangle) \]
RNN language models

\[ p(tom \mid \langle s \rangle) \]
RNN language models

\[ p(tom \mid \langle s \rangle) \]
RNN language models

\[ p(\text{tom} \mid \langle s \rangle) \times p(\text{likes} \mid \langle s \rangle, \text{tom}) \]
RNN language models

\[ p(tom \mid \langle s \rangle) \times p(likes \mid \langle s \rangle, tom) \times p(beer \mid \langle s \rangle, tom, likes) \]
RNN language models

\[ p(\text{tom} \mid \langle s \rangle) \times p(\text{likes} \mid \langle s \rangle, \text{tom}) \times p(\text{beer} \mid \langle s \rangle, \text{tom}, \text{likes}) \times p(\langle /s \rangle \mid \langle s \rangle, \text{tom}, \text{likes}, \text{beer}) \]
Training RNN language models
Training RNN language models

\[
\begin{align*}
\text{tom} & \rightarrow \text{cost}_1 \\
\hat{p}_1 & \rightarrow \text{softmax} \\
\hat{h}_1 & \rightarrow \text{x}_1 \\
<x> & \rightarrow \text{h}_0 \\
\text{likes} & \rightarrow \text{cost}_2 \\
\text{softmax} & \rightarrow \text{softmax} \\
\text{softmax} & \rightarrow \text{softmax} \\
\text{softmax} & \rightarrow \text{softmax} \\
\text{softmax} & \rightarrow \text{softmax} \\
\text{beer} & \rightarrow \text{cost}_3 \\
\text{cost}_1 & \rightarrow \text{cost}_2 \\
\text{cost}_2 & \rightarrow \text{cost}_3 \\
\text{cost}_3 & \rightarrow \text{cost}_4 \\
\text{cost}_4 & \rightarrow \text{cost}_4
\end{align*}
\]
Training RNN language models

$\hat{p}_1, p_1$  
\[ \text{softmax} \]
\[ \text{softmax} \]
\[ \text{softmax} \]
\[ \text{softmax} \]

$\{\text{log loss/cross entropy}\}$

$\hat{p}_1, \hat{p}_1$

$\dot{p}_1, \dot{p}_1$

$\text{softmax}$

$\text{softmax}$

$\text{softmax}$

$\text{softmax}$

$h_0, h_0$

$x_1, x_1$

$h_1, h_1$

$x_2, x_2$

$h_2, h_2$

$x_3, x_3$

$h_3, h_3$

$x_4, x_4$

$h_4, h_4$

<s>, </s>
Training RNN language models
Training RNN language models
Training RNN language models

The cross-entropy objective seeks the maximum likelihood (MLE) objective.

“Find the parameters that make the training data most likely.”
Training RNN language models

The cross-entropy objective seeks the maximum likelihood (MLE) objective.

“Find the parameters that make the training data most likely.”

You will overfit.

1. Stop training early, based on a validation set
2. Weight decay / other regularizers

In contrast to count-based models, zeroes aren’t a problem.
RNN language models

• Unlike Markov \((n\)-gram\) models, RNNs never forget
  • However, they don’t always remember so well (recall Felix’s lectures on RNNs vs. LSTMs)

• Algorithms
  • Sample a sequence from the probability distribution defined by the RNN
  • Train the RNN to minimize cross entropy (aka MLE)
  • What about: what is the most probable sequence?
How well do RNN LMs do?

<table>
<thead>
<tr>
<th>Model Description</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>order=5 Markov Kneser-Ney freq. est.</td>
<td>221</td>
</tr>
<tr>
<td>RNN 400 hidden</td>
<td>171</td>
</tr>
<tr>
<td>3xRNN interpolation</td>
<td>151</td>
</tr>
</tbody>
</table>

Mikolov et al. (2010 Interspeech) “Recurrent neural network based language model”
## How well do RNN LMs do?

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Perplexity</th>
<th>Word Error Rate (WER)</th>
</tr>
</thead>
<tbody>
<tr>
<td>order=5 Markov Kneser-Ney freq. est.</td>
<td>221</td>
<td>13.5</td>
</tr>
<tr>
<td>RNN 400 hidden</td>
<td>171</td>
<td>12.5</td>
</tr>
<tr>
<td>3xRNN interpolation</td>
<td>151</td>
<td>11.6</td>
</tr>
</tbody>
</table>

Mikolov et al. (2010 Interspeech) “Recurrent neural network based language model”
Conditional LMs

A **conditional language model** assigns probabilities to sequences of words, \( \mathbf{w} = (w_1, w_2, \ldots, w_\ell) \), given some conditioning context, \( \mathbf{x} \).

As with unconditional models, it is again helpful to use the chain rule to decompose this probability:

\[
p(\mathbf{w} \mid \mathbf{x}) = \prod_{t=1}^{\ell} p(w_t \mid \mathbf{x}, w_1, w_2, \ldots, w_{t-1})
\]

What is the probability of the next word, given the history of previously generated words and conditioning context \( \mathbf{x} \)?
## Conditional LMs

<table>
<thead>
<tr>
<th><strong>x</strong> “input”</th>
<th><strong>w</strong> “text output”</th>
</tr>
</thead>
<tbody>
<tr>
<td>An author</td>
<td>A document written by that author</td>
</tr>
<tr>
<td>A topic label</td>
<td>An article about that topic</td>
</tr>
<tr>
<td>{SPAM, NOT_SPAM}</td>
<td>An email</td>
</tr>
<tr>
<td>A sentence in French</td>
<td>Its English translation</td>
</tr>
<tr>
<td>A sentence in English</td>
<td>Its French translation</td>
</tr>
<tr>
<td>A sentence in English</td>
<td>Its Chinese translation</td>
</tr>
<tr>
<td>An image</td>
<td>A text description of the image</td>
</tr>
<tr>
<td>A document</td>
<td>Its summary</td>
</tr>
<tr>
<td>A document</td>
<td>Its translation</td>
</tr>
<tr>
<td>Meterological measurements</td>
<td>A weather report</td>
</tr>
<tr>
<td>Acoustic signal</td>
<td>Transcription of speech</td>
</tr>
<tr>
<td>Conversational history + database</td>
<td>Dialogue system response</td>
</tr>
<tr>
<td>A question + a document</td>
<td>Its answer</td>
</tr>
<tr>
<td>A question + an image</td>
<td>Its answer</td>
</tr>
</tbody>
</table>
## Conditional LMs

<table>
<thead>
<tr>
<th>( \mathbf{x} ) “input”</th>
<th>( \mathbf{w} ) “text output”</th>
</tr>
</thead>
<tbody>
<tr>
<td>An author</td>
<td>A document written by that author</td>
</tr>
<tr>
<td>A topic label</td>
<td>An article about that topic</td>
</tr>
<tr>
<td>{SPAM, NOT_SPAM}</td>
<td>An email</td>
</tr>
<tr>
<td>A sentence in French</td>
<td>Its English translation</td>
</tr>
<tr>
<td>A sentence in English</td>
<td>Its French translation</td>
</tr>
<tr>
<td>A sentence in English</td>
<td>Its Chinese translation</td>
</tr>
<tr>
<td>An image</td>
<td>A text description of the image</td>
</tr>
<tr>
<td>A document</td>
<td>Its summary</td>
</tr>
<tr>
<td>A document</td>
<td>Its translation</td>
</tr>
<tr>
<td>Meterological measurements</td>
<td>A weather report</td>
</tr>
<tr>
<td>Acoustic signal</td>
<td>Transcription of speech</td>
</tr>
<tr>
<td>Conversational history + database</td>
<td>Dialogue system response</td>
</tr>
<tr>
<td>A question + a document</td>
<td>Its answer</td>
</tr>
<tr>
<td>A question + an image</td>
<td>Its answer</td>
</tr>
</tbody>
</table>
## Conditional LMs

<table>
<thead>
<tr>
<th><strong>$\mathbf{x}$ “input”</strong></th>
<th><strong>$\mathbf{w}$ “text output”</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>An author</td>
<td>A document written by that author</td>
</tr>
<tr>
<td>A topic label</td>
<td>An article about that topic</td>
</tr>
<tr>
<td>{SPAM, NOT_SPAM}</td>
<td>An email</td>
</tr>
<tr>
<td><strong>A sentence in French</strong></td>
<td><strong>Its English translation</strong></td>
</tr>
<tr>
<td>A sentence in English</td>
<td>Its French translation</td>
</tr>
<tr>
<td>A sentence in English</td>
<td>Its Chinese translation</td>
</tr>
<tr>
<td><strong>An image</strong></td>
<td><strong>A text description of the image</strong></td>
</tr>
<tr>
<td>A document</td>
<td>Its summary</td>
</tr>
<tr>
<td>A document</td>
<td>Its translation</td>
</tr>
<tr>
<td>Meterological measurements</td>
<td>A weather report</td>
</tr>
<tr>
<td>Acoustic signal</td>
<td>Transcription of speech</td>
</tr>
<tr>
<td>Conversational history + database</td>
<td>Dialogue system response</td>
</tr>
<tr>
<td>A question + a document</td>
<td>Its answer</td>
</tr>
<tr>
<td>A question + an image</td>
<td>Its answer</td>
</tr>
</tbody>
</table>
Data for training conditional LMs

To train conditional language models, we need paired samples, \( \{(x_i, w_i)\}_{i=1}^N \).

Data availability varies. It’s easy to think of tasks that could be solved by conditional language models, but the data just doesn’t exist.

Relatively large amounts of data for:

- Translation, summarisation, caption generation,
- speech recognition
Evaluating conditional LMs

How good is our conditional language model?

These are language models, we can use cross-entropy or perplexity. okay to implement, hard to interpret

Task-specific evaluation. Compare the model’s most likely output to human-generated expected output using a task-specific evaluation metric $L$.

$$w^* = \arg \max_w p(w \mid x) \quad L(w^*, w_{ref})$$

Examples of $L$: BLEU, METEOR, WER, ROUGE. easy to implement, okay to interpret

Human evaluation. hard to implement, easy to interpret
Evaluating conditional LMs

How good is our conditional language model?

These are language models, we can use cross-entropy or perplexity.

Task-specific evaluation. Compare the model’s most likely output to human-generated expected output using a task-specific evaluation metric $L$.

$$w^* = \arg \max_w p(w | x) \quad L(w^*, w_{ref})$$

Examples of $L$: BLEU, METEOR, WER, ROUGE.

Human evaluation.
Lecture overview

The rest of this lecture will look at “encoder-decoder” models that learn a function that maps $x$ into a fixed-size vector and then uses a language model to “decode” that vector into a sequence of words, $w$.

$x \quad Kunst \ kann \ nicht \ gelehrt \ werden…$

encoder

representation

decoder

$w \quad Artistry \ can’t \ be \ taught…$
Lecture overview

The rest of this lecture will look at “encoder-decoder” models that learn a function that maps $x$ into a fixed-size vector and then uses a language model to “decode” that vector into a sequence of words, $w$.

A dog is playing on the beach.
Lecture overview

- Two questions
  - How do we encode $x$ as a fixed-size vector, $c$?
    - Problem (or at least modality) specific
    - Think about assumptions
  - How do we condition on $c$ in the decoding model?
    - Less problem specific
    - We will review one standard solution: RNNs
Encoder

\[ c = \text{embed}(x) \]

\[ s = Vc \]
Encoder

\[ c = \text{embed}(x) \]

\[ s = Vc \]

Recurrent decoder

\[ h_t = g(W[h_{t-1}; w_{t-1}] + s + b) \]

\[ u_t = P h_t + b' \]

\[ p(W_t \mid x, w_{<t}) = \text{softmax}(u_t) \]
Kalchbrenner and Blunsom 2013

Encoder

\[ c = \text{embed}(x) \]

\[ s = Vc \]

Recurrent decoder

\[ h_t = g(W[h_{t-1}; w_{t-1}] + s + b)) \]

\[ u_t = Ph_t + b' \]

\[ p(W_t | x, w_{<t}) = \text{softmax}(u_t) \]

Recall unconditional RNN

\[ h_t = g(W[h_{t-1}; w_{t-1}] + b) \]
K&B 2013: Encoder

How should we define $c = \text{embed}(x)$?

The simplest model possible:

![Diagram showing the simplest model possible with $c = \sum x_i$, where $x_1, x_2, x_3, x_4, x_5, x_6$ are input nodes and $x_1, x_2, x_3, x_4, x_5, x_6$ are output nodes.]

What do you think of this model?
Encoder
\[ c = \text{embed}(x) \]
\[ s = Vc \]

Recurrent decoder
\[ h_t = \text{g}(W[h_{t-1}; w_{t-1}] + s + b) \]
\[ u_t = P h_t + b' \]
\[ p(W_t | x, w_{<t}) = \text{softmax}(u_t) \]

Recall unconditional RNN
\[ h_t = \text{g}(W[h_{t-1}; w_{t-1}] + b) \]
K&B 2013: RNN Decoder
K&B 2013: RNN Decoder
K&B 2013: RNN Decoder

\[ p(\text{tom} \mid s, \langle s \rangle) \]
K&B 2013: RNN Decoder

\[ p(tom \mid s, \langle s \rangle) \times p(likes \mid s, \langle s \rangle, tom) \]
K&B 2013: RNN Decoder

\[
p(tom \mid s, \langle s \rangle) \times p(likes \mid s, \langle s \rangle, tom) \\
\times p(beer \mid s, \langle s \rangle, tom, likes)
\]
K&B 2013: RNN Decoder

\[ p(\text{tom} \mid s, \langle s \rangle) \times p(\text{likes} \mid s, \langle s \rangle, \text{tom}) \]
\[ \times p(\text{beer} \mid s, \langle s \rangle, \text{tom}, \text{likes}) \]
\[ \times p(\langle \langle s \rangle \rangle \mid s, \langle s \rangle, \text{tom}, \text{likes}, \text{beer}) \]
A word about decoding

In general, we want to find the most probable (MAP) output given the input, i.e.

\[ w^* = \arg \max_w p(w \mid x) \]

\[ = \arg \max_w \sum_{t=1}^{|w|} \log p(w_t \mid x, w_{<t}) \]
In general, we want to find the most probable (MAP) output given the input, i.e.

$$w^* = \arg \max_w p(w \mid x)$$

$$= \arg \max_w \sum_{t=1}^{|w|} \log p(w_t \mid x, w_{<t})$$

This is, for general RNNs, a hard problem. We therefore approximate it with a **greedy search**:

$$w_1^* \approx \arg \max_{w_1} p(w_1 \mid x)$$

$$w_2^* \approx \arg \max_{w_2} p(w_2 \mid x, w_1^*)$$

$$\vdots$$

$$w_t^* \approx \arg \max_{w_t} p(w_t \mid x, w_{<t}^*)$$

**undecidable :(

A word about decoding

A slightly better approximation is to use a beam search with beam size $b$. Key idea: keep track of top $b$ hypotheses.

E.g., for $b=2$:

$$x = Bier \ trinke \ ich$$

$\langle s \rangle$

logprob=0

\[ w_0 \quad \quad w_1 \quad \quad w_2 \quad \quad w_3 \]
A word about decoding

A slightly better approximation is to use a \textbf{beam search} with beam size $b$. Key idea: keep track of top $b$ hypothesis.

E.g., for $b=2$:

$x = \textit{Bier trinke ich}$

\begin{itemize}
  \item \textit{beer} \quad \text{logprob}=-1.82
  \item \textit{I} \quad \text{logprob}=-2.11
\end{itemize}
A word about decoding

A slightly better approximation is to use a **beam search** with beam size $b$. Key idea: keep track of top $b$ hypothesis.

E.g., for $b=2$:

$x = Bier trinke ich$

\[
\begin{align*}
\langle s \rangle & \quad \text{logprob}=0 \\
\text{beer} & \quad \text{logprob}=-1.82 \\
\text{drink} & \quad \text{logprob}=-6.93 \\
I & \quad \text{logprob}=-2.11 \\
I & \quad \text{logprob}=-5.80
\end{align*}
\]
A word about decoding

A slightly better approximation is to use a beam search with beam size $b$. Key idea: keep track of top $b$ hypothesis.

E.g., for $b=2$:

$x = \text{Bier trinke ich}$

$\langle s \rangle$

logprob=0

$w_0$

beer

logprob=-1.82

$w_1$

I

logprob=-2.11

$w_2$

drink

logprob=-2.87

$w_3$

beer

logprob=-5.80

$w_4$

I

logprob=-6.93

$w_5$

drink

logprob=-8.66
A word about decoding

A slightly better approximation is to use a **beam search** with beam size $b$. Key idea: keep track of top $b$ hypothesis.

E.g., for $b=2$:

$x = \text{Bier trinke ich}$

- $\langle s \rangle$: logprob=0
- $I$: logprob=-2.11
- $\text{beer}$: logprob=-1.82
- $\text{drink}$: logprob=-2.87
- $\text{I}$: logprob=-5.80
- $\text{beer}$: logprob=-8.66
- $\text{drink}$: logprob=-6.93
A word about decoding

A slightly better approximation is to use a beam search with beam size $b$. Key idea: keep track of top $b$ hypothesis.

E.g., for $b=2$:

$x = \text{Bier trinke ich}$

\begin{align*}
\langle s \rangle & \quad \text{logprob}=0 \\
\text{beer} & \quad \text{logprob}=-1.82 \\
\text{I} & \quad \text{logprob}=-2.11 \\
\text{I} & \quad \text{logprob}=-5.80 \\
\text{beer} & \quad \text{logprob}=-6.93 \\
\text{beer} & \quad \text{logprob}=-8.66 \\
\text{drink} & \quad \text{logprob}=-2.87 \\
\text{drink} & \quad \text{logprob}=-6.28 \\
\text{like} & \quad \text{logprob}=-7.31 \\
\text{beer} & \quad \text{logprob}=-3.04 \\
\text{wine} & \quad \text{logprob}=-5.12
\end{align*}
A word about decoding

A slightly better approximation is to use a **beam search** with beam size $b$. Key idea: keep track of top $b$ hypothesis.

E.g., for $b=2$:

$$x = Bier \ trinke \ ich$$

$$\langle s \rangle \quad \text{logprob}=0$$

$$w_0 \quad \langle s \rangle \quad \text{logprob}=-1.82$$

$$w_1 \quad I \quad \text{logprob}=-2.11$$

$$w_2 \quad beer \quad \text{logprob}=-8.66$$

$$w_3 \quad drink \quad \text{logprob}=-2.87$$

$$w_0 \quad be\quad \text{logprob}=-2.87$$

$$w_1 \quad drink \quad \text{logprob}=-6.93$$

$$w_2 \quad like \quad \text{logprob}=-7.31$$

$$w_3 \quad beer \quad \text{logprob}=-6.28$$

$$w_0 \quad drink \quad \text{logprob}=-6.28$$

$$w_1 \quad like \quad \text{logprob}=-7.31$$

$$w_2 \quad beer \quad \text{logprob}=-3.04$$

$$w_3 \quad wine \quad \text{logprob}=-5.12$$

$x = Bier \ trinke \ ich$

beer drink I
A word about decoding

A slightly better approximation is to use a beam search with beam size $b$. Key idea: keep track of top $b$ hypothesis.

E.g., for $b=2$:

$x = Bier trinke ich$

\begin{align*}
\text{beer} &\quad \text{drink} &\quad \text{I} \\
\logprob = -2.11 &\quad \logprob = -5.80 &\quad \logprob = -7.31
\end{align*}
How well does this model do?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>order=5 Markov Kneser-Ney freq. est.</td>
<td>222</td>
<td>225</td>
</tr>
<tr>
<td>RNN LM</td>
<td>178</td>
<td>181</td>
</tr>
<tr>
<td>RNN LM + x</td>
<td>140</td>
<td>142</td>
</tr>
</tbody>
</table>
How well does this model do?
How well does this model do?
How well does this model do?
How well does this model do?

may i have a wake-up call at seven tomorrow morning?
How well does this model do?

where 's the currency exchange office?

CLM

货币 兑换处 在 哪里 ？
How well does this model do?

I'd like to have a room under thirty dollars a night.
How well does this model do?

(Literal: I will feel bad if you do not find a solution.)
How well does this model do?

(Literal: I will feel bad if you do not find a solution.)
Summary

• Conditional language modeling provides a convenient formulation for a lot of practical applications

• Two big problems:
  • Model expressivity
  • Decoding difficulties

• Next time
  • A better encoder for vector to sequence models
  • “Attention” for better learning
  • Lots of results on machine translation
Questions?