Better Conditional Language Modeling

Chris Dyer

DeepMind

Carnegie Mellon University
Conditional LMs

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

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

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

What is the probability of the next word, given the history of previously generated words **and** conditioning context \( x \)?
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: RNN Decoder

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

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

The simplest model possible:
K&B 2013: Problems

- The bag of words assumption is really bad (part 1)

  Alice saw Bob.
  Bob saw Alice.

  I would like some fresh bread with aged cheese.
  I would like some aged bread with fresh cheese.

- We are putting a lot of information inside a single vector (part 2)
Sutskever et al. (2014)

LSTM encoder

\((c_0, h_0)\) are parameters

\((c_i, h_i) = \text{LSTM}(x_i, c_{i-1}, h_{i-1})\)

The encoding is \((c_\ell, h_\ell)\) where \(\ell = |x|\).

LSTM decoder

\[w_0 = \langle s \rangle\]

\[(c_{t+\ell}, h_{t+\ell}) = \text{LSTM}(w_{t-1}, c_{t+\ell-1}, h_{t+\ell-1})\]

\[u_t = Ph_{t+\ell} + b\]

\[p(W_t \mid x, w_{<t}) = \text{softmax}(u_t)\]
Sutskever et al. (2014)
Sutskever et al. (2014)

Beginnings

START

Aller  Anfang  ist  schwer  STOP
Beginnings are

Aller Anfang ist schwer STOP

Sutskever et al. (2014)
Beginnings are difficult

Sutskever et al. (2014)
Beginnings are difficult STOP

Aller Anfang ist schwer STOP

Sutskever et al. (2014)
Sutskever et al. (2014)

- **Good**
  - RNNs deal naturally with sequences of various lengths
  - LSTMs in principle can propagate gradients a long distance
  - Very simple architecture!

- **Bad**
  - The hidden state has to remember a lot of information!
Sutskever et al. (2014): Tricks

Beginnings are difficult STOP

START

Aller Anfang ist schwer STOP
Sutskever et al. (2014): Tricks

Read the input sequence “backwards”: +4 BLEU
Sutskever et al. (2014): Tricks

Use an ensemble of \( J \) independently trained models.

Ensemble of 2 models: +3 BLEU
Ensemble of 5 models: +4.5 BLEU

Decoder:

\[
(c^{(j)}_{t+\ell}, h^{(j)}_{t+\ell}) = \text{LSTM}^{(j)}(w_{t-1}, c^{(j)}_{t+\ell-1}, h^{(j)}_{t+\ell-1})
\]

\[
u_t^{(j)} = \text{Ph}_t^{(j)} + b^{(j)}
\]

\[
u_t = \frac{1}{J} \sum_{j'=1}^{J} u^{(j')}
\]

\[
p(W_t \mid x, w_{<t}) = \text{softmax}(u_t)
\]
Sutskever et al. (2014): Tricks

Use beam search: +1 BLEU

\[ x = \text{Bier trinke ich} \]

\[ \text{beer} \quad \text{drink} \quad \text{I} \]

\[ \langle s \rangle \logprob=0 \]

\[ w_0 \quad w_1 \quad w_2 \quad w_3 \]
Sutskever et al. (2014): Tricks

Use beam search: **+1 BLEU**

Make the beam really big: **-1 BLEU**

(Koehn and Knowles, 2017)
Conditioning with vectors

We are compressing a lot of information in a finite-sized vector.
Conditioning with vectors

We are compressing a lot of information in a finite-sized vector.

“You can't cram the meaning of a whole %&!#$ sentence into a single $&!#* vector!”

Prof. Ray Mooney
Conditioning with vectors

We are compressing a lot of information in a finite-sized vector.

Gradients have a long way to travel. Even LSTMs forget!
Conditioning with vectors

We are compressing a lot of information in a finite-sized vector.

Gradients have a long way to travel. Even LSTMs forget!

What is to be done?
Solving the vector bottleneck

- Represent a source sentence as a matrix
- Generate a target sentence from a matrix

- This will
  - Solve the capacity problem
  - Solve the gradient flow problem
Sentences as **matrices**

• Problem with the fixed-size vector model

• Sentences are of different sizes but vectors are of the same size

• Solution: use matrices instead

• Fixed number of rows, but number of columns depends on the number of words

• Usually $|\mathbf{f}| = \#\text{cols}$
Sentences as matrices

Ich möchte ein Bier
Sentences as matrices

Ich möchte ein Bier
Mach's gut
Ich möchte ein Bier
Mach’s gut
Die Wahrheiten der Menschen sind die unwiderlegbaren Irrtümer
Sentences as matrices

\[ \begin{array}{c}
\text{Ich möchte ein Bier} \\
\text{Mach’s gut} \\
\text{Die Wahrheiten der Menschen sind die unwiderlegbaren Irrtümer}
\end{array} \]

Question: How do we build these matrices?
Sentences as matrices
With concatenation

• Each word type is represented by an n-dimensional vector

• Take all of the vectors for the sentence and concatenate them into a matrix

• Simplest possible model

• So simple, no one has bothered to publish how well/badly it works!
Ich möchte ein Bier
Ich möchte ein Bier
Ich möchte ein Bier

\[ f_i = x_i \]

\( \mathbf{F} \in \mathbb{R}^{n \times |\mathbf{f}|} \)

Ich möchte ein Bier
Sentences as matrices
With bidirectional RNNs

- By far the most widely used matrix representation, due to Bahdanau et al (2015)
- One column per word
- Each column (word) has two halves concatenated together:
  - a “forward representation”, i.e., a word and its left context
  - a “reverse representation”, i.e., a word and its right context
- Implementation: **bidirectional RNNs** (GRUs or LSTMs) to read \( f \) from left to right and right to left, concatenate representations
Ich möchte ein Bier
Ich möchte ein Bier!
Ich möchte ein Bier!
Ich möchte ein Bier
Ich möchte ein Bier

\[ f_i = [ \overrightarrow{h_i}; \overleftarrow{h_i} ] \]
Ich möchte ein Bier
f_i = [h_i; h_i]

Ich möchte ein Bier!
Ich möchte ein Bier

\[ f_i = [\hat{h}_i; \vec{h}_i] \]

\[ F \in \mathbb{R}^{2n \times |f|} \]

Ich möchte ein Bier
Sentences as matrices
Where are we in 2018?

• There are lots of ways to construct $F$

• More exotic architectures coming out daily

• Increasingly common goal: get rid of $O(|f|)$ sequential processing steps, i.e., RNNs during training

• syntactic information can help (Sennrich & Haddow, 2016; Nadejde et al., 2017), but many more integration strategies are possible

• try something with phrase types instead of word types?

  Multi-word expressions are a pain in the neck.
Conditioning on matrices

- We have a matrix $F$ representing the input, now we need to generate from it
- Bahdanau et al. (2015) and Luong et al. (2015) concurrently proposed using attention for translating from matrix-encoded sentences
- High-level idea
  - Generate the output sentence word by word using an RNN
  - At each output position $t$, the RNN receives two inputs (in addition to any recurrent inputs)
    - a fixed-size vector embedding of the previously generated output symbol $e_{t-1}$
    - a fixed-size vector encoding a “view” of the input matrix
  - How do we get a fixed-size vector from a matrix that changes over time?
    - Bahdanau et al: do a weighted sum of the columns of $F$ (i.e., words) based on how important they are at the current time step. (i.e., just a matrix-vector product $Fa_t$)
    - The weighting of the input columns at each time-step ($a_t$) is called attention
Recall RNNs...
Recall RNNs...
Recall RNNs…
Recall RNNs...
Recall RNNs...
Recall RNNs...
Ich möchte ein Bier
Ich möchte ein Bier
Ich möchte ein Bier

Attention history:

\[ a_1 \]
Ich möchte ein Bier

Attention history:

\[ c_1 = F a_1 \]
Ich möchte ein Bier

\[ c_1 = F a_1 \]

**Attention history:**

\[
\begin{bmatrix}
    a_1 & & & \\
    & & & \\
    & & & \\
\end{bmatrix}
\]
Ich möchte ein Bier

c_1 = Fa_1

Attention history:
Ich möchte ein Bier
I'd like a beer.

Attention history:

Ich möchte ein Bier
Ich möchte ein Bier
Ich möchte ein Bier

Attention history:
Ich möchte ein Bier

Attention history:
I'd like a

Ich möchte ein Bier

Attention history:
Ich möchte ein Bier
I'd like a beer

Ich möchte ein Bier
Attention

- How do we know what to attend to at each time-step?
- That is, how do we compute $a_t$?
At each time step (one time step = one output word), we want to be able to “attend” to different words in the source sentence

- We need a weight for every column: this is an $|f|$-length vector $a_t$
- Here is a simplified version of Bahdanau et al.’s solution
  - Use an RNN to predict model output, call the hidden states $s_t$
    - $(s_t$ has a fixed dimensionality, call it $m)$
Computing attention

- At each time step (one time step = one output word), we want to be able to “attend” to different words in the source sentence
  - We need a weight for every column: this is an \(|f|\)-length vector \(a_t\)
  - Here is a simplified version of Bahdanau et al.’s solution
    - Use an RNN to predict model output, call the hidden states \(s_t\) \((s_t\) has a fixed dimensionality, call it \(m\))
    - At time \(t\) compute the \textbf{query key embedding} \(r_t = Vs_{t-1}\) \((V\) is a learned parameter)
Computing attention

- At each time step (one time step = one output word), we want to be able to “attend” to different words in the source sentence.
- We need a weight for every column: this is an $|f|$-length vector $a_t$.
- Here is a simplified version of Bahdanau et al.’s solution:
  - Use an RNN to predict model output, call the hidden states $s_t$ ($s_t$ has a fixed dimensionality, call it $m$).
  - At time $t$ compute the **query key embedding** $r_t = Vs_{t-1}$ (V is a learned parameter).
  - Take the dot product with every column in the source matrix to compute the **attention energy** $u_t = F^T r_t$ (called $e_t$ in the paper).
    (Since $F$ has $|f|$ columns, $u_t$ has $|f|$ rows).
Computing attention

- At each time step (one time step = one output word), we want to be able to “attend” to different words in the source sentence
  - We need a weight for every column: this is an $|f|$-length vector $a_t$
  - Here is a simplified version of Bahdanau et al.’s solution
    - Use an RNN to predict model output, call the hidden states $s_t$ ($s_t$ has a fixed dimensionality, call it $m$)
    - At time $t$ compute the **query key embedding** $r_t = Vs_{t-1}$ ($V$ is a learned parameter)
    - Take the dot product with every column in the source matrix to compute the **attention energy** $u_t = F^\top r_t$ (called $e_t$ in the paper)
      - (Since $F$ has $|f|$ columns, $u_t$ has $|f|$ rows)
    - Exponentiate and normalize to 1: $a_t = \text{softmax}(u_t)$
      - (called $\alpha_t$ in the paper)
Computing attention

- At each time step (one time step = one output word), we want to be able to “attend” to different words in the source sentence
  - We need a weight for every column: this is an $|f|$-length vector $a_t$
  - Here is a simplified version of Bahdanau et al.’s solution
    - Use an RNN to predict model output, call the hidden states $s_t$ ($s_t$ has a fixed dimensionality, call it $m$)
    - At time $t$ compute the query key embedding $r_t = Vs_{t-1}$ ($V$ is a learned parameter)
    - Take the dot product with every column in the source matrix to compute the attention energy $u_t = F^T r_t$ (called $e_t$ in the paper)
      (Since $F$ has $|f|$ columns, $u_t$ has $|f|$ rows)
    - Exponentiate and normalize to 1: $a_t = \text{softmax}(u_t)$ (called $\alpha_t$ in the paper)
    - Finally, the input source vector for time $t$ is $c_t = Fa_t$
Computing attention

• In the actual model, Bahdanau et al. replace the dot product between the columns of $F$ and $r_t$ with an MLP:

$$u_t = F^\top r_t$$

(simple model)
Computing attention
Nonlinear additive attention model

• In the actual model, Bahdanau et al. replace the dot product between the columns of $F$ and $r_t$ with an MLP:

\[
\begin{align*}
\mathbf{u}_t &= \mathbf{F}^\top \mathbf{r}_t & \text{(simple model)} \\
\mathbf{u}_t &= \mathbf{v}^\top \tanh(\mathbf{W} \mathbf{F} + \mathbf{r}_t) & \text{(Bahdanau et al)}
\end{align*}
\]
Computing attention
Nonlinear additive attention model

- In the actual model, Bahdanau et al. replace the dot product between the columns of $F$ and $r_t$ with an MLP:
  \[ u_t = F^\top r_t \] (simple model)
  \[ u_t = v^\top \tanh(WF + r_t) \] (Bahdanau et al)

- Here, $W$ and $v$ are learned parameters of appropriate dimension and $+$ “broadcasts” over the $|f|$ columns in $WF$

- This can learn more complex interactions
  - It is unclear if the added complexity is necessary for good performance
Putting it all together

\[
\begin{align*}
F &= \text{EncodeAsMatrix}(f) \quad \text{(Part 1 of lecture)} \\
e_0 &= \langle s \rangle \\
s_0 &= w \quad \text{(Learned initial state; Bahdanau uses } U \hat{h}_1 \text{)} \\
t &= 0 \\
\textbf{while } e_t \neq \langle /s \rangle : \\
& \quad t = t + 1 \\
& \quad r_t = Vs_{t-1} \\
& \quad u_t = v^\top \tanh(WF + r_t) \quad \text{(Compute attention; part 2 of lecture)} \\
& \quad a_t = \text{softmax}(u_t) \\
& \quad c_t = Fa_t \\
& \quad s_t = \text{RNN}(s_{t-1}, [e_{t-1}; c_t]) \quad \text{([}e_{t-1}\text{} \text{is a learned embedding of } e_t\text{)} \\
y_t &= \text{softmax}(Ps_t + b) \quad \text{(P and b are learned parameters)} \\
e_t \mid e_{<t} &\sim \text{Categorical}(y_t)
\end{align*}
\]
Putting it all together

\[ F = \text{EncodeAsMatrix}(f) \quad \text{(Part 1 of lecture)} \]
\[ e_0 = \langle s \rangle \]
\[ s_0 = w \quad \text{(Learned initial state; Bahdanau uses } U \hat{h}_1) \]
\[ t = 0 \]
\[ \text{while } e_t \neq \langle /s \rangle : \]
\[ t = t + 1 \]
\[ r_t = Vs_{t-1} \]
\[ u_t = \mathbf{v}^\top \tanh(W_F + r_t) \]
\[ a_t = \text{softmax}(u_t) \]
\[ c_t = Fa_t \]
\[ s_t = \text{RNN}(s_{t-1}, [e_{t-1}; c_t]) \quad \text{(} e_{t-1} \text{ is a learned embedding of } e_t) \]
\[ y_t = \text{softmax}(Ps_t + b) \quad \text{(} P \text{ and } b \text{ are learned parameters}) \]
\[ e_t \mid e_{<t} \sim \text{Categorical}(y_t) \]

\text{doesn't depend on output decisions}
Putting it all together

\[ F = \text{EncodeAsMatrix}(f) \quad \text{(Part 1 of lecture)} \]
\[ e_0 = \langle s \rangle \]
\[ s_0 = w \quad \text{(Learned initial state; Bahdanau uses } \hat{U} \tilde{h}_1) \]
\[ t = 0 \]
\[ X = WF \]

while \( e_t \neq \langle /s \rangle \):
\[ t = t + 1 \]
\[ r_t = Vs_{t-1} \]
\[ u_t = v^\top \tanh(WF + r_t) \]
\[ a_t = \text{softmax}(u_t) \]
\[ c_t = Fa_t \]
\[ s_t = \text{RNN}(s_{t-1}, [e_{t-1}; c_t]) \quad \text{(} e_{t-1} \text{ is a learned embedding of } e_t \text{)} \]
\[ y_t = \text{softmax}(Ps_t + b) \quad \text{(} P \text{ and } b \text{ are learned parameters)} \]
\[ e_t \mid e_{<t} \sim \text{Categorical}(y_t) \]
Putting it all together

\[ F = \text{EncodeAsMatrix}(f) \]  \hspace{1cm} \text{(Part 1 of lecture)}

\[ e_0 = \langle s \rangle \]

\[ s_0 = w \] (Learned initial state; Bahdanau uses \( U\hat{h}_1 \))

\[ t = 0 \]

\[ X = WF \]

while \( e_t \neq \langle /s \rangle \):

\[ t = t + 1 \]

\[ r_t = Vs_{t-1} \]

\[ u_t = v^\top \tanh(X + r_t) \]

\[ a_t = \text{softmax}(u_t) \]

\[ c_t = Fa_t \]

\[ s_t = \text{RNN}(s_{t-1}, [e_{t-1}; c_t]) \] \( (e_{t-1} \) is a learned embedding of \( e_t \))

\[ y_t = \text{softmax}(Ps_t + b) \] \( (P \) and \( b \) are learned parameters)

\[ e_t \mid e_{<t} \sim \text{Categorical}(y_t) \]
Attention in MT: Results

Add attention to seq2seq translation: +11 BLEU
The agreement on the European Economic Area was signed in August 1992.

It should be noted that the marine environment is the least known of environments.
A word about gradients
I'd like a beer

Ich möchte ein Bier
I'd like a beer

Attention history:

Ich möchte ein Bier
I'd like a beer

Ich möchte ein Bier
Attention and translation

• Cho’s question: does a translator read and memorize the input sentence/document and then generate the output?

• Compressing the entire input sentence into a vector basically says “memorize the sentence”

• Common sense experience says translators refer back and forth to the input. (also backed up by eye-tracking studies)

• Should humans be a model for machines?
Summary

- Attention
  - provides the ability to establish information flow directly from distant
  - closely related to “pooling” operations in convnets (and other architectures)
- Traditional attention model seems to only cares about “content”
  - No obvious bias in favor of diagonals, short jumps, fertility, etc.
  - Some work has begun to add other “structural” biases (Luong et al., 2015; Cohn et al., 2016), but there are lots more opportunities
  - Factorization into keys and values (Miller et al., 2016; Ba et al., 2016, Gulcehre et al., 2016)
- Attention weights provide interpretation you can look at
Questions?