Seq2Seq + Attention/Copy

Wei Xu

(many slides from Greg Durrett)
This Week

- Sequence-to-Sequence Model
- Attention Mechanism
- Copy Mechanism
- Transformer Architecture
Recall: CNNs vs. LSTMs

- Both LSTMs and convolutional layers transform the input using context.
- LSTM: “globally” looks at the entire sentence (but local for many problems).
- CNN: local depending on filter width + number of layers.
LSTMs

\[ c_j = c_{j-1} \odot f + g \odot i \]
\[ f = \sigma(x_j W^{xf} + h_{j-1} W^{hf}) \]
\[ g = \tanh(x_j W^{xg} + h_{j-1} W^{hg}) \]
\[ i = \sigma(x_j W^{xi} + h_{j-1} W^{hi}) \]
\[ h_j = \tanh(c_j) \odot o \]
\[ o = \sigma(x_j W^{xo} + h_{j-1} W^{ho}) \]

- \( f, i, o \) are gates that control information flow
- \( g \) reflects the main computation of the cell

Hochreiter & Schmidhuber (1997)

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Gradient still diminishes, but in a controlled way and generally by less — usually initialize forget gate = 1 to remember everything to start.
Encoder-Decoder

- Encode a sequence into a fixed-sized vector
- Now use that vector to produce a series of tokens as output from a separate LSTM decoder
- Machine translation, NLG, summarization, dialog, and many other tasks (e.g., semantic parsing, syntactic parsing) can be done using this framework.

Sutskever et al. (2014)
It’s not an ACL tutorial on vector representations of meaning if there isn’t at least one Ray Mooney quote.

- Is this true? Sort of... we’ll come back to this later
Model

- Generate next word conditioned on previous word as well as hidden state
- W size is $|\text{vocab}| \times |\text{hidden state}|$, softmax over entire vocabulary

$$P(y_i|x, y_1, \ldots, y_{i-1}) = \text{softmax}(W\bar{h})$$

$$P(y|x) = \prod_{i=1}^{n} P(y_i|x, y_1, \ldots, y_{i-1})$$

Decoder has separate parameters from encoder, so this can learn to be a language model (produce a plausible next word given current one)
Inference

- Generate next word conditioned on previous word as well as hidden state

- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state

- Need to actually evaluate computation graph up to this point to form input for the next state

- Decoder is advanced one state at a time until [STOP] is reached
Implementing seq2seq Models

- Encoder: consumes sequence of tokens, produces a vector. Analogous to encoders for classification/tagging tasks.

- Decoder: separate module, single cell. Takes two inputs: hidden state (vector $h$ or tuple $(h, c)$) and previous token. Outputs token + new state.
Training

- Objective: maximize

\[
\sum_{(x,y)} \sum_{i=1}^{n} \log P(y_i^* | x, y_1^*, \ldots, y_{i-1}^*)
\]

- One loss term for each target-sentence word, feed the correct word regardless of model’s prediction
Training: Scheduled Sampling

- Model needs to do the right thing even with its own predictions

- Scheduled sampling: with probability $p$, take the gold as input, else take the model’s prediction

- Starting with $p = 1$ and decaying it works best

Bengio et al. (2015)
Sentence lengths vary for both encoder and decoder:

- Typically pad everything to the right length

Encoder: Can be a CNN/LSTM/…

Decoder: Execute one step of computation at a time, so computation graph is formulated as taking one input + hidden state. Until reach `<STOP>`.

Beam search: can help with lookahead. Finds the (approximate) highest scoring sequence:

\[
\arg\max_{y} \prod_{i=1}^{n} P(y_i | x, y_1, \ldots, y_{i-1})
\]
Implementation Details

Fairseq:

- GETTING STARTED
  - Evaluating Pre-trained Models
  - Training a New Model
  - Advanced Training Options
  - Command-line Tools
- EXTENDING FAIRSEQ
  - Overview

Tutorial: Simple LSTM

1. Building an Encoder and Decoder
2. Registering the Model
3. Training the Model
4. Making generation faster

Tutorial: Classifying Names with a Character-Level RNN

Library Reference

Tasks

Models

Decoder

Our Decoder will predict the next word, conditioned on the Encoder's final hidden state and an embedded representation of the previous target word – which is sometimes called teacher forcing. More specifically, we'll use a `torch.nn.LSTM` to produce a sequence of hidden states that we'll project to the size of the output vocabulary to predict each target word.

```python
import torch
from fairseq.models import FairseqDecoder

class SimpleLSTMDriver(FairseqDecoder):
    def __init__(
        self, dictionary, encoder_hidden_dim=128, embed_dim=128, hidden_dim=128,
        dropout=0.1,
    );
    super().__init__(dictionary)

    # Our decoder will embed the inputs before feeding them to the LSTM.
    self.embed_tokens = nn.Embedding(
        num_embeddings=len(dictionary),
        embedding_dim=embed_dim,
        padding_idx=dictionary.pad(),
    )
    self.dropout = nn.Dropout(p=dropout)

    # We'll use a single-layer, unidirectional LSTM for simplicity.
    self.lstm = nn.LSTM(
        input_size=encoder_hidden_dim + embed_dim,
        hidden_size=hidden_dim,
        num_layers=1,
        bidirectional=False,
    )

    # Define the output projection.
    self.output_projection = nn.Linear(hidden_dim, len(dictionary))
```
Beam Search

- Maintain decoder state, token history in beam

- Keep both *film* states! Hidden state vectors are different
Can use for other semantic parsing-like tasks

Predict regex from text

Problem: requires a lot of data: 10,000 examples needed to get ~60% accuracy on pretty simple regexes

Locascio et al. (2016)
Semantic Parsing as Translation

“what states border Texas”

\[ \lambda x \; \text{state}(x) \land \text{borders}(x, \text{e89}) \]

- Write down a linearized form of the semantic parse, train seq2seq models to directly translate into this representation
- No need to have an explicit grammar, simplifies algorithms
- Might not produce well-formed logical forms, might require lots of data

Semantic Parsing/Lambda Calculus: https://www.youtube.com/watch?v=OocGXG-BY6k&t=200s

Jia and Liang (2015)
SQL Generation

- Convert natural language description into a SQL query against some DB
- How to ensure that well-formed SQL is generated?
  - Three components
- How to capture column names + constants?
  - Pointer mechanisms

Question:
How many CFL teams are from York College?

SQL:
```
SELECT COUNT CFL Team FROM CFLDraft WHERE College = "York"
```
Attention
Problems with Seq2seq Models

- Encoder-decoder models like to repeat themselves:

  *Un garçon joue dans la neige* → *A boy plays in the snow* boy plays boy plays

- Often a byproduct of training these models poorly. Input is forgotten by the LSTM so it gets stuck in a “loop” of generation the same output tokens again and again.

- Need some notion of input coverage or what input words we’ve translated
Problems with Seq2seq Models

- Bad at long sentences: 1) a fixed-size hidden representation doesn’t scale; 2) LSTMs still have a hard time remembering for really long sentences

RNNenc: the model we’ve discussed so far

RNNsearch: uses attention

Bahdanau et al. (2014)
Problems with Seq2seq Models

- Unknown words:

  *en*: The *ecotax* portico in *Pont-de-Buis*, ... [truncated] ..., was taken down on Thursday morning

  *fr*: Le *portique écotaxe* de *Pont-de-Buis*, ... [truncated] ..., a été *démonté* jeudi matin

  *nn*: Le *unk* de *unk à unk*, ... [truncated] ..., a été pris le jeudi matin

- Encoding these rare words into a vector space is really hard

- In fact, we don’t want to encode them, we want a way of directly looking back at the input and copying them (Pont-de-Buis)

  Jean et al. (2015), Luong et al. (2015)
Aligned Inputs

- Suppose we knew the source and target would be word-by-word translated

- Can look at the corresponding input word when translating — this could scale!

- Much less burden on the hidden state

- How can we achieve this without hardcoding it?
At each decoder state, compute a distribution over source inputs based on current decoder state

Use that in output layer
Attention

- For each decoder state, compute weighted sum of input states

- No attn:  \( P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \text{softmax}(W\bar{h}_i) \)

\[
P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \text{softmax}(W[c_i; \bar{h}_i])
\]

\[
c_i = \sum_j \alpha_{ij} h_j
\]

\[
\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}
\]

\[
e_{ij} = f(\bar{h}_i, h_j)
\]

- Weighted sum of input hidden states (vector)

- Some function f (next slide)
Attention

\[ c_i = \sum_j \alpha_{ij} h_j \]
\[ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})} \]
\[ e_{ij} = f(\bar{h}_i, h_j) \]
\[ f(\bar{h}_i, h_j) = \tanh(W[\bar{h}_i, h_j]) \]

- Bahdanau+ (2014): additive
  \[ f(\bar{h}_i, h_j) = \bar{h}_i \cdot h_j \]
- Luong+ (2015): dot product
  \[ f(\bar{h}_i, h_j) = \bar{h}_i^\top W h_j \]
- Luong+ (2015): bilinear

Note that this all uses outputs of hidden layers
What can attention do?

- Learning to copy — how might this work?

- LSTM can learn to count with the right weight matrix

- This is effectively position-based addressing

Luong et al. (2015)
What can attention do?

- Learning to subsample tokens

- Need to count (for ordering) and also determine which tokens are in/out

- Content-based addressing

Luong et al. (2015)
- Encoder hidden states capture contextual source word identity

- Decoder hidden states are now mostly responsible for selecting what to attend to

- Doesn’t take a complex hidden state to walk monotonically through a sentence and spit out word-by-word translations
Batching Attention

token outputs: batch size x sentence length x dimension

sentence outputs: batch size x hidden size

hidden state: batch size x hidden size

\[
e_{ij} = f(\tilde{h}_i, h_j)
\]

\[
\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}
\]

attention scores = batch size x sentence length

c = batch size x hidden size

\[
c_i = \sum_j \alpha_{ij} h_j
\]

Make sure tensors are the right size!

Luong et al. (2015)
Results

- Machine translation: BLEU score of 14.0 on English-German -> 16.8 with attention, 19.0 with smarter attention (constrained to a small windows)

- Summarization/headline generation: bigram recall from 11% -> 15%

- Semantic parsing: ~30% accuracy -> 70+% accuracy on Geoquery

Luong et al. (2015)
Chopra et al. (2016)
Jia and Liang (2016)
Copying Input/Pointers
Want to be able to copy named entities like Pont-de-Buis

\[ P(y_i|x, y_1, \ldots, y_{i-1}) = \text{softmax}(W[c_i; \bar{h}_i]) \]

from attention from RNN hidden state

Problem: target word has to be in the vocabulary, a8enPon + RNN need to generate good embedding to pick it

Jean et al. (2015), Luong et al. (2015)
Pointer Network

- **Standard decoder** ($P_{\text{vocab}}$): softmax over vocabulary

  \[ P(y_i|x, y_1, \ldots, y_{i-1}) = \text{softmax}(W[c_i; \bar{h}_i]) \]

- **Pointer network** ($P_{\text{pointer}}$): predict from source words, instead of target vocabulary

  \[ P_{\text{pointer}}(y_i|x, y_1, \ldots, y_{i-1}) \propto \begin{cases} h_j^T V \bar{h}_i & \text{if } y_i = w_j \\ 0 & \text{otherwise} \end{cases} \]
Define the decoder model as a mixture model of $P_{\text{vocab}}$ and $P_{\text{pointer}}$

$$P(y_i | x, y_1, \ldots, y_{i-1}) = P(\text{copy})P_{\text{pointer}} + (1 - P(\text{copy}))P_{\text{vocab}}$$

- Predict $P(\text{copy})$ based on decoder state, input, etc.
- Marginalize over copy variable during training and inference
- Model will be able to both generate and copy, flexibly adapt between the two

Gulcehre et al. (2016), Gu et al. (2016)
Copying

en: The ecotax portico in Pont-de-Buis, … [truncated] …
fr: Le portique écotaxe de Pont-de-Buis, … [truncated]
nn: Le unk de unk à unk, … [truncated] …, a été pris

Vocabulary contains “normal” vocab as well as words in input. Normalizes over both of these:

\[
P(y_i = w | x, y_1, \ldots, y_{i-1}) \propto \begin{cases} 
\exp W_w[c_i; \bar{h}_i] & \text{if } w \text{ in vocab} \\
\bar{h}_i^T V \bar{h}_i & \text{if } w = x_j
\end{cases}
\]

Bilinear function of input representation + output hidden state
Copying in Summarization

See et al. (2017)
## Copying in Summarization

<table>
<thead>
<tr>
<th>Model Type</th>
<th>ROUGE 1</th>
<th>ROUGE 2</th>
<th>ROUGE L</th>
<th>METEOR exact match</th>
<th>METEOR + stem/syn/para</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstractive model (Nallapati et al., 2016)*</td>
<td>35.46</td>
<td>13.30</td>
<td>32.65</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>seq-to-seq + attn baseline (150k vocab)</td>
<td>30.49</td>
<td>11.17</td>
<td>28.08</td>
<td>11.65</td>
<td>12.86</td>
</tr>
<tr>
<td>seq-to-seq + attn baseline (50k vocab)</td>
<td>31.33</td>
<td>11.81</td>
<td>28.83</td>
<td>12.03</td>
<td>13.20</td>
</tr>
<tr>
<td>pointer-generator</td>
<td>36.44</td>
<td>15.66</td>
<td>33.42</td>
<td>15.35</td>
<td>16.65</td>
</tr>
<tr>
<td>pointer-generator + coverage</td>
<td><strong>39.53</strong></td>
<td><strong>17.28</strong></td>
<td><strong>36.38</strong></td>
<td>17.32</td>
<td>18.72</td>
</tr>
<tr>
<td>lead-3 baseline (ours)</td>
<td>40.34</td>
<td>17.70</td>
<td>36.57</td>
<td>20.48</td>
<td>22.21</td>
</tr>
<tr>
<td>lead-3 baseline (Nallapati et al., 2017)*</td>
<td>39.2</td>
<td>15.7</td>
<td>35.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>extractive model (Nallapati et al., 2017)*</td>
<td>39.6</td>
<td>16.2</td>
<td>35.3</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

See et al. (2017)
Copying in Summarization

Original Text (truncated): Lagos, Nigeria (CNN) a day after winning Nigeria’s presidency, Muhammadu Buhari told CNN’s Christiane Amanpour that he plans to aggressively fight corruption that has long plagued Nigeria and go after the root of the nation’s unrest. Buhari said he’ll “rapidly give attention” to curbing violence in the northeast part of Nigeria, where the terrorist group Boko Haram operates. By cooperating with neighboring nations Chad, Cameroon and Niger, he said his administration is confident it will be able to thwart criminals and others contributing to Nigeria’s instability. For the first time in Nigeria’s history, the opposition defeated the ruling party in democratic elections. Buhari defeated incumbent Goodluck Jonathan by about 2 million votes, according to Nigeria’s independent national electoral commission. The win comes after a long history of military rule, coups and botched attempts at democracy in Africa’s most populous nation.

Baseline Seq2Seq + Attention: UNK UNK says his administration is confident it will be able to destabilize Nigeria’s economy. UNK says his administration is confident it will be able to thwart criminals and other Nigerians. He says the country has long Nigeria and Nigeria’s economy.

Pointer-Gen: Muhammadu Buhari says he plans to aggressively fight corruption in the northeast part of Nigeria. He says he’ll “rapidly give attention” to curbing violence in the northeast part of Nigeria. He says his administration is confident it will be able to thwart criminals.

Pointer-Gen + Coverage: Muhammadu Buhari says he plans to aggressively fight corruption that has long plagued Nigeria. He says his administration is confident it will be able to thwart criminals. The win comes after a long history of military rule, coups and botched attempts at democracy in Africa’s most populous nation.

Figure 1: Comparison of output of 3 abstractive summarization models on a news article. The baseline model makes factual errors, a nonsensical sentence and struggles with OOV words Muhammadu Buhari. The pointer-generator model is accurate but repeats itself. Coverage eliminates repetition. The final summary is composed from several fragments.
Transformers
Sentence Encoders

- LSTM abstraction: maps each vector in a sentence to a new, context-aware vector
  
- CNNs do something similar with filters

- Attention can give us a third way to do this

Vaswani et al. (2017)
Self-Attention

- Assume we’re using GloVe — what do we want our neural network to do?

\[ \text{The ballerina is very excited that she will dance in the show.} \]

- What words need to be contextualized here?
  - Pronouns need to look at antecedents
  - Ambiguous words should look at context
  - Words should look at syntactic parents/children

- Problem: LSTMs and CNNs don’t do this

Vaswani et al. (2017)
Self-Attention

- Want:
  \[ \text{The ballerina is very excited that she will dance in the show.} \]

- LSTMs/CNNs: tend to look at local context
  \[ \text{The ballerina is very excited that she will dance in the show.} \]

- To appropriately contextualize embeddings, we need to pass information over long distances dynamically for each word

Vaswani et al. (2017)
Self-Attention

- Each word forms a “query” which then computes attention over each word
  \[
  \alpha_{i,j} = \text{softmax}(x_i^\top x_j) \quad \text{scalar}
  \]
  \[
  x'_i = \sum_{j=1}^{n} \alpha_{i,j} x_j \quad \text{vector = sum of scalar * vector}
  \]

- Multiple “heads” analogous to different convolutional filters. Use parameters \( W_k \) and \( V_k \) to get different attention values + transform vectors
  \[
  \alpha_{k,i,j} = \text{softmax}(x_i^\top W_k x_j) \quad x'_{k,i} = \sum_{j=1}^{n} \alpha_{k,i,j} V_k x_j
  \]

Vaswani et al. (2017)
What can self-attention do?

\[ \text{The ballerina is very excited that she will dance in the show.} \]

- Attend nearby + to semantically related terms
- This is a demonstration, we will revisit what these models actually learn when we discuss BERT
- Why multiple heads? Softmaxes end up being peaked, single distribution cannot easily put weight on multiple things

Vaswani et al. (2017)
Transformer Uses

- **Supervised:** transformer can replace LSTM as encoder, decoder, or both; will revisit this when we discuss MT

- **Unsupervised:** transformers work better than LSTM for unsupervised pre-training of embeddings: predict word given context words

- **BERT (Bidirectional Encoder Representations from Transformers):** pretraining transformer language models similar to ELMo

- **Stronger than similar methods, SOTA on ~11 tasks (including NER — 92.8 F1)**
Takeaways

- Attention is very helpful for seq2seq models

- Used for tasks including data-to-text generation and summarization

- Explicitly copying input can be beneficial as well

- Transformers are strong models we’ll come back to later, if time