Seq2Seq + Attention

Wei Xu

(many slides from Greg Durrett)
This Lecture

- 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.
Encoder-Decoder

- Encode a sequence into a fixed-sized vector

 Encode the movie was great

 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)
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
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)
Implementation Details

- 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:

\[
\arg \max_y \prod_{i=1}^{n} P(y_i | x, y_1, \ldots, y_{i-1})
\]
Beam Search

- Maintain decoder state, token history in beam

- Keep both film states! Hidden state vectors are different

```
<s>
la: 0.4
le: 0.3
les: 0.1

log(0.4)
log(0.3)
log(0.1)

film: 0.4

log(0.3)+log(0.8)
log(0.4)+log(0.4)
```
Regex Prediction

- Seq2seq models can be used for many other 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”

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

- 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.

Jia and Liang (2015)

Semantic Parsing/Lambda Calculus: [https://www.youtube.com/watch?v=OocGXG-BY6k&t=200s](https://www.youtube.com/watch?v=OocGXG-BY6k&t=200s)
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"
```

Zhong et al. (2017)
Attention
Recap: Seq2Seq Model

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

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

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

```
  le  film
```

```
  Encoder
  the movie was great

  Decoder
  le
```
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: \[ P(y_i | x, y_1, \ldots, y_{i-1}) = \text{softmax}(W\vec{h}_i) \]

- 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

hidden state: batch size x hidden size

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

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

attention scores = batch size x sentence length

$c_i = \sum_j \alpha_{ij} h_j$

c = batch size x hidden size

Luong et al. (2015)

- Make sure tensors are the right size!
“Early” Neural MT


PyTorch first released in 2016.

Effective Approaches to Attention-based Neural Machine Translation

Minh-Thang Luong  Hieu Pham  Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
{lmtthang,hypham,manning}@stanford.edu

Abstract

An attentional mechanism has lately been used to improve neural machine translation (NMT) by selectively focusing on parts of the source sentence during translation. However, there has been little work exploring useful architectures for attention-based NMT. This paper examines two simple and effective classes of attentional mechanisms: a global approach which always attends to all source words and a local one that only looks at a subset of source words at a time. We demonstrate the effectiveness of both approaches on the

![Diagram](image)

Figure 1: Neural machine translation – a stacking recurrent architecture for translating a source sequence A B C D into a target sequence X Y Z. Here, \(<\text{eos}>\) marks the end of a sentence.

This approach is motivated by (a) the benefits of using plain SGD, (b) the need to optimize over all possible translations, (c) a simple learning rate schedule is employed – we start with a learning rate of 1; after 5 epochs, we begin to halve the learning rate every epoch, (d) our mini-batch size is 128, and (e) the normalized gradient is rescaled whenever its norm exceeds 5. Additionally, we also use dropout with probability 0.2 for our LSTMs as suggested by (Zaremba et al., 2015). For dropout models, we train for 12 epochs and start halving the learning rate after 8 epochs. For local attention models, we empirically set the window size $D = 10$.

Our code is implemented in MATLAB. When running on a single GPU device Tesla K40, we achieve a speed of 1K target words per second. It takes 7–10 days to completely train a model.

Luong et al. (2015)
Neural MT Details
Encoder-Decoder MT

- Sutskever seq2seq paper: first major application of LSTMs to NLP
- Basic encoder-decoder with beam search

<table>
<thead>
<tr>
<th>Method</th>
<th>test BLEU score (ntst14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bahdanau et al. [2]</td>
<td>28.45</td>
</tr>
<tr>
<td>Baseline System [29]</td>
<td>33.30</td>
</tr>
<tr>
<td>Single forward LSTM, beam size 12</td>
<td>26.17</td>
</tr>
<tr>
<td>Single reversed LSTM, beam size 12</td>
<td>30.59</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 1</td>
<td>33.00</td>
</tr>
<tr>
<td>Ensemble of 2 reversed LSTMs, beam size 12</td>
<td>33.27</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 2</td>
<td>34.50</td>
</tr>
<tr>
<td>Ensemble of 5 reversed LSTMs, beam size 12</td>
<td><strong>34.81</strong></td>
</tr>
</tbody>
</table>

- SOTA = 37.0 — not all that competitive...
Results: WMT English-French

- 12M sentence pairs

Classic phrase-based system: \( \sim 33 \) BLEU, uses additional target-language data
  
  Rerank with LSTMs: \( 36.5 \) BLEU (long line of work here; Devlin+ 2014)

Sutskever+ (2014) seq2seq single: \( 30.6 \) BLEU

Sutskever+ (2014) seq2seq ensemble: \( 34.8 \) BLEU

Luong+ (2015) seq2seq ensemble with attention and rare word handling: \( 37.5 \) BLEU

- But English-French is a really easy language pair and there’s *tons* of data for it! Does this approach work for anything harder?
Results: WMT English-German

- 4.5M sentence pairs

Classic phrase-based system: **20.7** BLEU

Luong+ (2014) seq2seq: **14** BLEU

Luong+ (2015) seq2seq ensemble with rare word handling: **23.0** BLEU

- Not nearly as good in absolute BLEU, but not really comparable across languages

- French, Spanish = easiest
  - German, Czech = harder
  - Japanese, Russian = hard (grammatically different, lots of morphology... )
### MT Examples

<table>
<thead>
<tr>
<th>src</th>
<th>In einem Interview sagte Bloom jedoch, dass er und Kerr sich noch immer lieben.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref</td>
<td>However, in an interview, Bloom has said that he and <em>Kerr</em> still love each other.</td>
</tr>
<tr>
<td>best</td>
<td>In an interview, however, Bloom said that he and <em>Kerr</em> still love.</td>
</tr>
<tr>
<td>base</td>
<td>However, in an interview, Bloom said that he and <em>Tina</em> were still <em>&lt;unk&gt;</em>.</td>
</tr>
</tbody>
</table>

- **best** = with attention, **base** = no attention

- NMT systems can hallucinate words, especially when not using attention — phrase-based doesn’t do this

Luong et al. (2015)
| src | Wegen der von Berlin und der Europäischen Zentralbank verhängten strengen Sparpolitik in Verbindung mit der Zwangsjacke, in die die jeweilige nationale Wirtschaft durch das Festhalten an der gemeinsamen Währung genötigt wird, sind viele Menschen der Ansicht, das Projekt Europa sei zu weit gegangen |
| ref | The *austerity imposed by Berlin and the European Central Bank, coupled with the straitjacket imposed on national economies through adherence to the common currency*, has led many people to think Project Europe has gone too far. |
| best | Because of the strict *austerity measures imposed by Berlin and the European Central Bank in connection with the straitjacket* in which the respective national economy is forced to adhere to the common currency, many people believe that the European project has gone too far. |
| base | Because of the pressure *imposed by the European Central Bank and the Federal Central Bank with the strict austerity* imposed on the national economy in the face of the single currency, many people believe that the European project has gone too far. |

- best = with attention, base = no attention

Luong et al. (2015)
### MT Examples

<table>
<thead>
<tr>
<th>Source</th>
<th>such changes in reaction conditions include, but are not limited to, an increase in temperature or change in pH.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>所(such)述(said)反应(reaction)条件(condition)的(of)改变(change)包括(include)但(not)限于(limit)温度(temperature)的(of)增加(increase)或(or)pH值(value)的(of)改变(change)。</td>
</tr>
<tr>
<td>PBMT</td>
<td>中(in)的(of)这种(such)变化(change)的(of)反应(reaction)条件(condition)包括(include),但(not)限于(limit),增加(increase)的(of)温度(temperature)或(or)pH变化(change)。</td>
</tr>
<tr>
<td>NMT</td>
<td>这种(such)反应(reaction)条件(condition)的(of)变化(change)包括(include)但(not)限于(limit)pH或(or)pH的(of)变化(change)。</td>
</tr>
</tbody>
</table>

- NMT can repeat itself if it gets confused (pH or pH)
- Phrase-based MT often gets chunks right, may have more subtle ungrammaticalities

---

Zhang et al. (2017)
Handling Rare Words

- Words are a difficult unit to work with: copying can be cumbersome, word vocabularies get very large.

- Character-level models don’t work well.

- Solution: “word pieces” (which may be full words but may be subwords).

  Input: _the _eco tax _port i co _in _Po nt - de - Bu is ...

  Output: _le _port ique _éco taxe _de _Pont - de - Bui s

- Can help with transliteration; capture shared linguistic characteristics between languages (e.g., transliteration, shared word root, etc.)

  Wu et al. (2016)
Byte Pair Encoding (BPE)

- Start with every individual byte (basically character) as its own symbol

```python
for i in range(num_merges):
    pairs = get_stats(vocab)
    best = max(pairs, key=pairs.get)
    vocab = merge_vocab(best, vocab)
```

- Count bigram character cooccurrences
- Merge the most frequent pair of adjacent characters

- Do this either over your vocabulary (original version) or over a large corpus (more common version)
- Final vocabulary size is often in 10k ~ 30k range for each language
- Most SOTA NMT systems use this on both source + target

Sennrich et al. (2016)
Word Pieces

while voc size < target voc size:

Build a language model over your corpus

Merge pieces that lead to highest improvement in language model perplexity

- SentencePiece library from Google: unigram LM
- Result: way of segmenting input appropriate for translation

Schuster and Nakajima (2012), Wu et al. (2016), Kudo and Richardson (2018)
Google’s NMT System

- 8-layer LSTM encoder-decoder with attention, word piece vocabulary of 8k-32k

Wu et al. (2016)
Google’s NMT System

English-French:
Google’s phrase-based system: 37.0 BLEU
Luong+ (2015) seq2seq ensemble with rare word handling: 37.5 BLEU
Google’s 32k word pieces: 38.95 BLEU

English-German:
Google’s phrase-based system: 20.7 BLEU
Luong+ (2015) seq2seq ensemble with rare word handling: 23.0 BLEU
Google’s 32k word pieces: 24.2 BLEU

Wu et al. (2016)
Figure 6: Histogram of side-by-side scores on 500 sampled sentences from Wikipedia and news websites for a typical language pair, here English → Spanish (PBMT blue, GNMT red, Human orange). It can be seen that there is a wide distribution in scores, even for the human translation when rated by other humans, which shows how ambiguous the task is. It is clear that GNMT is much more accurate than PBMT.

Wu et al. (2016)
Google’s NMT System

<table>
<thead>
<tr>
<th>Source</th>
<th>She was spotted three days later by a dog walker trapped in the quarry</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBMT</td>
<td>Elle a été repéré trois jours plus tard par un promeneur de chien piégé dans la carrière</td>
</tr>
<tr>
<td>GNMT</td>
<td>Elle a été repérée trois jours plus tard par un traîneau à chiens piégé dans la carrière.</td>
</tr>
<tr>
<td>Human</td>
<td>Elle a été repérée trois jours plus tard par une personne qui promenait son chien coincée dans la carrière</td>
</tr>
</tbody>
</table>

Gender is correct in GNMT but not in PBMT

“sled”

“walker”

The right-most column shows the human ratings on a scale of 0 (complete nonsense) to 6 (perfect translation)

Wu et al. (2016)