Neural Speed Reading via Skim-RNN

Minjoon Seo$^{1,2,*}$, Sewon Min$^{3,*}$, Ali Farhadi$^{2,4,5}$, Hannaneh Hajishirzi$^2$
NAVER Clova$^1$, University of Washington$^2$, Seoul National University$^3$, Allen Institute for AI$^4$, XNOR.AI$^5$
May 4, 2018

* denotes equal contribution.
RNN for “reading”

Intelligent

and

invigorating

film
RNN for “reading”

• RNN has become the standard model for reading text
  • Machine Translation
  • Question Answering
  • Classification
  • Syntactic Parsing
  • Natural Language Inference
  • ...
RNNs are slow…

• RNNs are slow on GPUs
  • RNNs cannot be parallelized

• CPUs are slow too
  • So many (millions) FLOP in neural networks

• Recent works to replace RNNs:
  • Google’s Attention-based MT (2017)
  • Facebook’s CNN-based MT (2017)

FLOP = Floating-point operations i.e. # of computations
RNNs are slow...

- RNNs are slow on GPUs
  - RNNs cannot be parallelized

- CPUs are slow too
  - So many (millions) FLOP in neural networks

- Recent works to replace RNNs:
  - Google’s Transformer (2017)
  - Facebook’s CNN-based MT (2017)
Inference speed on CPUs

• CPUs are often more desirable options than GPUs for production
• Small devices often times only have CPUs
• Latency-critical applications
  • CPUs can have lower latency than GPUs.

How can we make RNNs faster on CPUs?
Speed Reading

• “Readers make longer pauses at points where processing loads are greater … as a function of the involvement of the various levels of processing” (Just & carpenter, 1980).

• ‘Reading’ is similar to matrix multiplication in RNN.
  • **Skim**: use small matrix multiplication.
  • **Fully read**: use big matrix multiplication.

Sentiment Classification with Skim-RNN

$h_0$

$x_1$

“Intelligent”
Sentiment Classification with Skim-RNN

Intelligent
Sentiment Classification with Skim-RNN

$h_0$

$Q_1=1$

READ

$x_1$

“Intelligent”
Sentiment Classification with Skim-RNN

\[ Q_1 = 1 \]

"Intelligent"
Sentiment Classification with Skim-RNN

Big RNN

$h_0$ $h_1$

$x_1$ "Intelligent"

$x_2$ "and"

$Q_1 = 1$
Sentiment Classification with Skim-RNN

Intelligent

"and"

Q₁=1

Q₂=2

READ

SKIM

h₀

h₁

x₁

x₂

Big RNN

x₁

"Intelligent"

x₂

"and"
Sentiment Classification with Skim-RNN

Intelligent and

$Q_1 = 1$

$Q_2 = 2$

$\text{READ} \ x_1$

$\text{SKIM} \ x_2$

Big RNN

Small RNN
Sentiment Classification with Skim-RNN

```
Intelligent
\&=1
\&=2
```

```
Big RNN
```

```
Small RNN
COPY
```

```
x_1
"Intelligent"
```

```
x_2
"and"
```

```
h_0
```

```
h_1
```

```
h_2
```
Intelligent and invigorating film
Similar to Pulse Width Modulation

50% duty cycle

75% duty cycle

25% duty cycle
Skim-RNN

• Consists of two RNNs:
  • **Big RNN**: hidden state size = $d$
  • **Small RNN**: hidden state size = $d'$
  • $d' >> d'$ (e.g. $d = 100$, $d' = 5$)

• Hidden state is shared between the RNNs.
• Big RNN updates the entire hidden state.
• Small RNN updates only a small portion of the hidden state.
• When using small RNN, the inference requires smaller # of FLOP.
  • $O(d^2) >> O(d'd)$
• Dynamically makes decision on which size of RNN to use
Decision ($Q_t$) as a random variable

$x_t$: Input  \hspace{1cm} h_{t-1}$: Previous hidden state

$p_t = \text{softmax}(\alpha(x_t, h_{t-1}))$

$Q_t = \text{Multinomial}(p_t)$

$Q = [Q_1, Q_2, \ldots, Q_T]$

$E[L(\theta)] = \sum_{Q\in Q} L(\theta; Q)Pr(Q)$

But the sample space is exponentially large!
How to train?

• Computing gradient is intractable

• Policy gradient (Williams, 1992)
  • REINFORCE
  • Unbiased gradient estimation
REINFORCE (Williams, 1992)

\[ \mathbb{E}[L(\theta)] = \sum_{Q \in \mathcal{Q}} L(\theta; Q) \Pr(Q) \]

\[ \nabla \log \mathbb{E}[L(\theta)] = \mathbb{E}[\nabla \log L(\theta; Q) + \log L(\theta; Q) \nabla \log \Pr(Q)] \]

Gradient can be *sampled*

But the sample space is exponentially large!
How to train?

• Computing gradient is intractable

• Policy gradient (Williams, 1992)
  • REINFORCE
  • Unbiased estimation
  • High variance; hard to train

• Gumbel-Softmax (Jang et al., 2017)
  • Biased estimation
  • Low variance; good empirical results
  • Fully differentiable during training via reparameterization
Gumbel-Softmax Reparameterization (Jang et al., 2017)

- Start with
  - soft decision (attention) $p$
  - Random variable $g$ from Gumbel distribution

\[
\begin{align*}
\mathbf{r}_t^i &= \frac{\exp((\log(p_t^i) + g_t^i)/\tau)}{\sum_j \exp((\log(p_t^j) + g_t^j)/\tau)} \\
\mathbf{h}_t &= \sum_i \mathbf{r}_t^i \tilde{\mathbf{h}}_t^i
\end{align*}
\]

- Slowly anneal (decrease $\tau$), making the distribution more discrete
- Near 0 temperature, identical to categorical distribution
- Sampling $g$ gives stochasticity
- Reparameterization allows differentiation with stochasticity
- *Shake & Anneal*
Experiments

• 4 Classification Tasks
  • Stanford Sentiment Treebank (SST)
  • Rotten Tomatoes (RT)
  • IMDb
  • AG News

• 2 Question Answering Tasks
  • Stanford Question Answering Dataset (SQuAD)
  • Children Book Test (CBT)
Baselines

• Regular RNN (LSTM)

• LSTM-Jump (Yu et al., 2017)
  • Learns to *skip* inputs

• Variable-Computation RNN (VCRNN) (Jernite et al, 2017)
  • Variable number of hidden state units to update
# SST and RT Results

<table>
<thead>
<tr>
<th>Model</th>
<th>SST</th>
<th>Rotten Tomatoes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (LSTM)</td>
<td>86.4%</td>
<td>82.5%</td>
</tr>
<tr>
<td>LSTM-Jump</td>
<td>-</td>
<td>79.3% / 1.6x Speed</td>
</tr>
<tr>
<td>VCRNN</td>
<td>81.9% / 2.6x FLOP</td>
<td>-</td>
</tr>
<tr>
<td>Skim-RNN</td>
<td>86.4% / 3.0x FLOP</td>
<td>84.2% / 1.3x Speed</td>
</tr>
</tbody>
</table>
Question Answering Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>EM</th>
<th>FLOP-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (LSTM+Att)</td>
<td>75.5%</td>
<td>67.0%</td>
<td>1.0x</td>
</tr>
<tr>
<td>VCRNN</td>
<td>74.9%</td>
<td>65.4%</td>
<td>1.0x</td>
</tr>
<tr>
<td>Skim-RNN</td>
<td>75.0%</td>
<td>66.0%</td>
<td>2.3x</td>
</tr>
</tbody>
</table>
F1 vs FLOP across diff configs.
Visualization on IMDb Sentiment Classification

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>I <strong>liked</strong> this movie, not because Tom Selleck was in it, but <strong>because</strong> it was a <strong>good</strong> story about baseball and it also had a semi-over <strong>dramatized</strong> view of some of the issues that a BASEBALL player coming to the end of their time in Major League sports must face. I also <strong>greatly enjoyed</strong> the cultural differences in American and Japanese baseball and the small facts on how the games are played differently. <strong>Overall</strong>, it is a <strong>good movie</strong> to watch on Cable TV or rent on a cold winter’s night and watch about the &quot;Dog Day’s&quot; of summer and know that spring training is only a few months away. A <strong>good movie</strong> for a baseball fan as well as a good &quot;DATE&quot; movie. Trust me on that one! <em>Wink</em></td>
<td><strong>No! no - No - NO!</strong> My <strong>entire</strong> being is <strong>revolting</strong> against this <strong>dreadful</strong> remake of a classic movie. I <strong>knew</strong> we were heading for trouble from the moment Meg Ryan appeared on screen with her <strong>ridiculous</strong> hair and clothing - literally looking like a scarecrow in that garden she was digging. Meg Ryan playing Meg Ryan - how <strong>tiresome</strong> is that?! And it <strong>got worse</strong> ... so much <strong>worse</strong>. The <strong>horribly cliché</strong> lines, the stock characters, the increasing sense I was <strong>watching</strong> a spin-off of &quot;The First Wives Club&quot; and the ultimate <strong>hackneyed</strong> schtick in the delivery room. How many times have I seen this movie? Only once, but it <strong>feel</strong> like a dozen times - <strong>nothing</strong> original or fresh about it. For shame!</td>
</tr>
</tbody>
</table>

*Black words are skimmed (small RNN), blue words are fully read.*
Visualization on Stanford Question Answering Dataset

Q What is one straightforward case of a probabilistic test?

A particularly simple example of a probabilistic test is the Fermat primality test, which relies on the fact (Fermat’s little theorem) that $np \equiv n \pmod{p}$ for any $n$ if $p$ is a prime number. If you have a number $b$ that we want to test for primality, then we work out $nb \pmod{b}$ for a random value of $n$ as our test. A flaw with this test is that there are some composite numbers (the Carmichael numbers) that satisfy the Fermat identity even though they are not prime, so the test has no way of distinguishing between prime numbers and Carmichael numbers. Carmichael numbers are substantially rarer than prime numbers, though, so this test can be useful for practical purposes. More powerful extensions of the Fermat primality test, such as Baillie-PSW, Miller-Rabin, and Solovay-Strassen tests, are guaranteed to fail at least some of the time when applied to a composite number.

*Black words are skimmed (small RNN), blue words are fully read.
Layer-wise Skim Visualization (SQuAD)

Most RNN steps at higher layer is redundant!
Dynamically controlling # of FLOP

F1 vs. Flop-R (Float operation Reduction)

- $d' = 10$
- $d' = 0$
Conclusion

• **Skim-RNN**: switching between two different-size RNNs with shared hidden state.
  - Can be generalized to multiple RNNs.
• Speed gain can be substantial.
• Especially useful for **latency**.
Future Work

• Using multiple granularities of RNNs (not just two)
• Extension to latency-critical applications
  • Speech
  • Video
• Low-level implementation
Thanks! Questions?

- minjoon.seo@navercorp.com
- http://seominjoon.github.io