Neural Speed Reading

Minjoon Seo[1,2*], Sewon Min[3*], Ali Farhadi[2,4,5], Hannaneh Hajishirzi[2]

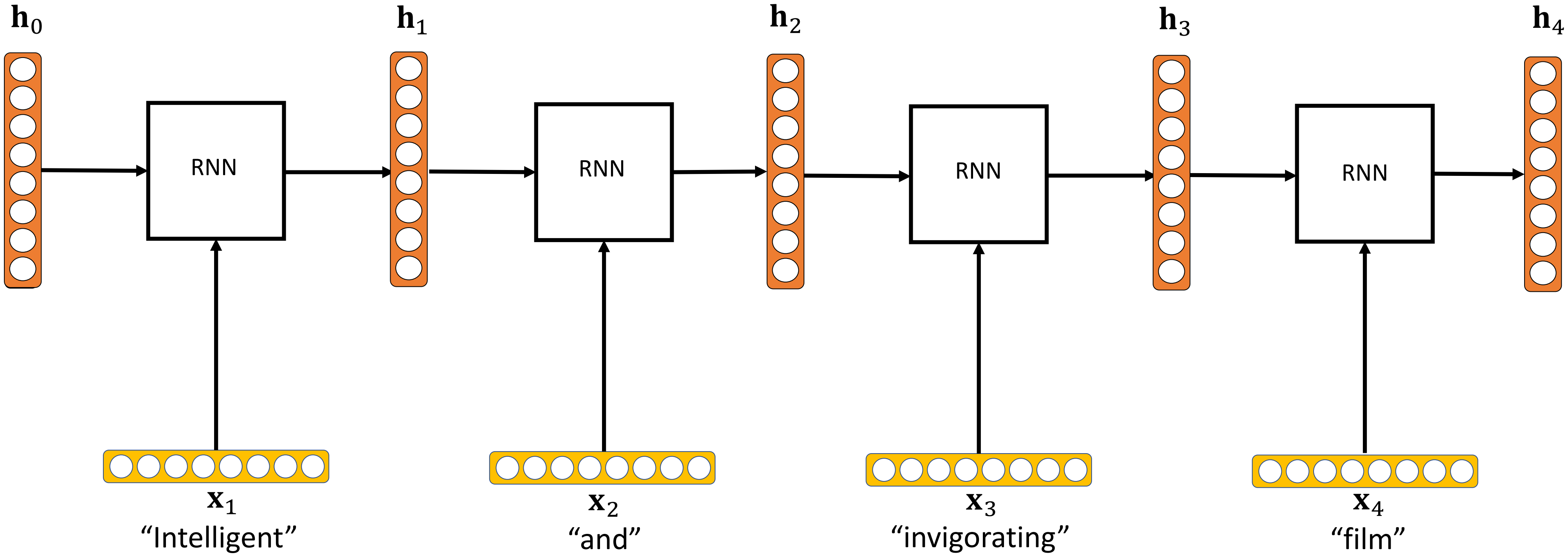
NAVER Clova[1], University of Washington[2], Seoul National University[3],

Allen Institute for AI[4], XNOR.AI[5]

Feb 15, 2018

** denotes equal contribution.*

Sentiment Classification with Recurrent Neural Network (RNN)



RNNs are slow…

- RNNs cannot be parallelized over time
  - Time complexity is linear in the length of sequence.
  - GPUs cannot take full advantage of parallelization.
  - Recent works to overcome slow RNNs: Google’s Transformer (2017), Facebook’s CNN-based MT (2017).

RNNs are slow…

- RNNs cannot be parallelized over time
  - Time complexity is linear in the length of sequence.
  - GPUs cannot take full advantage of parallelization.
  - Recent works to overcome slow RNNs: Google’s Transformer (2017), Facebook’s CNN-based MT (2017).

- CPUs are *not* great for RNNs either
  - Neural networks require so many computations.

(very rough) FLOP Complexity of RNN

- Hidden state size = d (= input size)

- $$\mathbf{h}_{t+1} = \underbrace{\sigma}_{d}\Big(\underbrace{\underbrace{\mathbf{W}^{(\mathbf{x})}\mathbf{x}_t}_{d^2} + \underbrace{\mathbf{W}^{(\mathbf{h})}\mathbf{h}_t}_{d^2}}_{d} \underbrace{+ \mathbf{b}}_{d}\Big)$$

- Total number of operations: $3d + 2d^2$
- If d is sufficiently large, **matrix multiplication** is the bottleneck.

Can we improve inference speed on CPUs?

- CPUs are often more desirable options for production
- Small devices (often CPU-only)
- Latency-critical applications
  - CPUs *can* have lower latency.

How do humans ‘speed-read’?

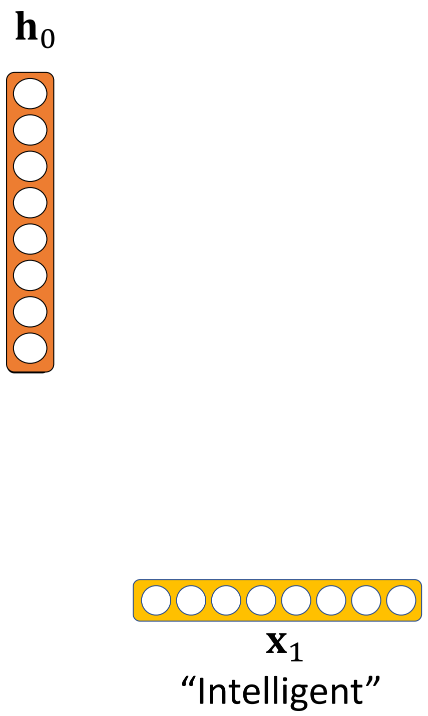
- Speed readers **skim** unimportant parts and **fully read** important text (Just and Carpenter, 1987).

How do humans ‘speed-read’?

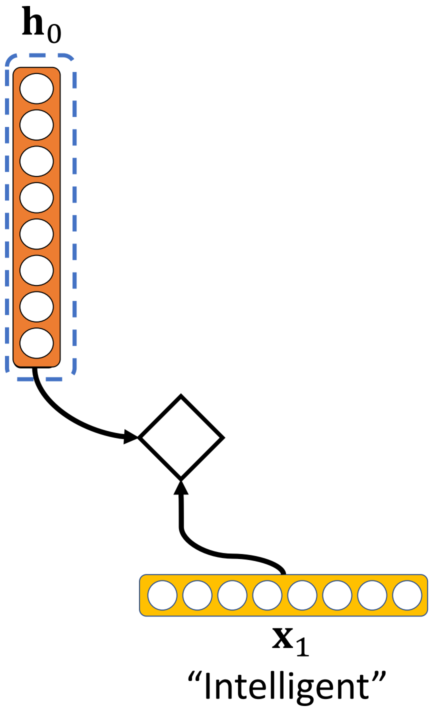
- Speed readers **skim** unimportant parts and **fully read** important text (Just and Carpenter, 1987).

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

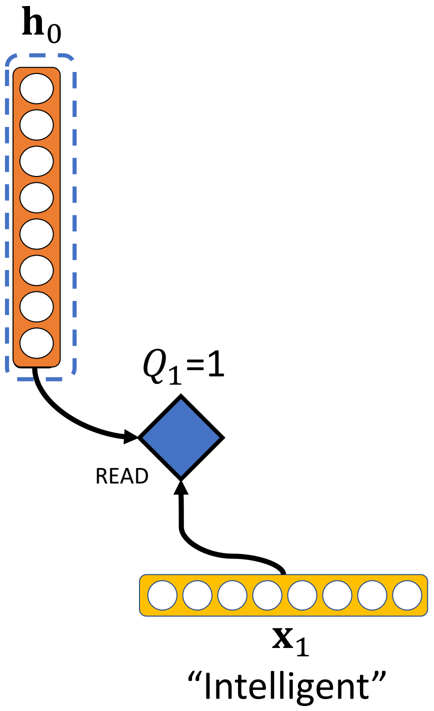
Sentiment Classification with Skim-RNN



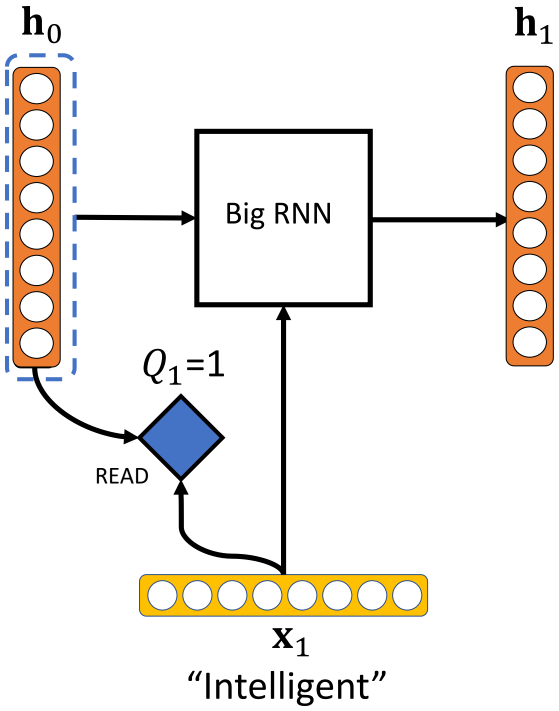
Sentiment Classification with Skim-RNN



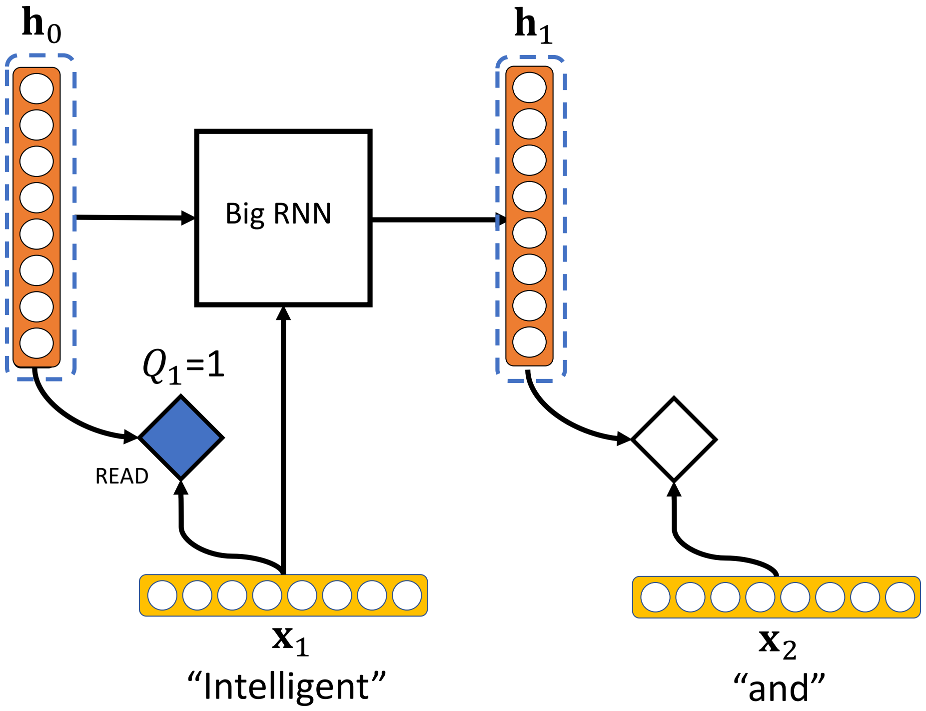
Sentiment Classification with Skim-RNN



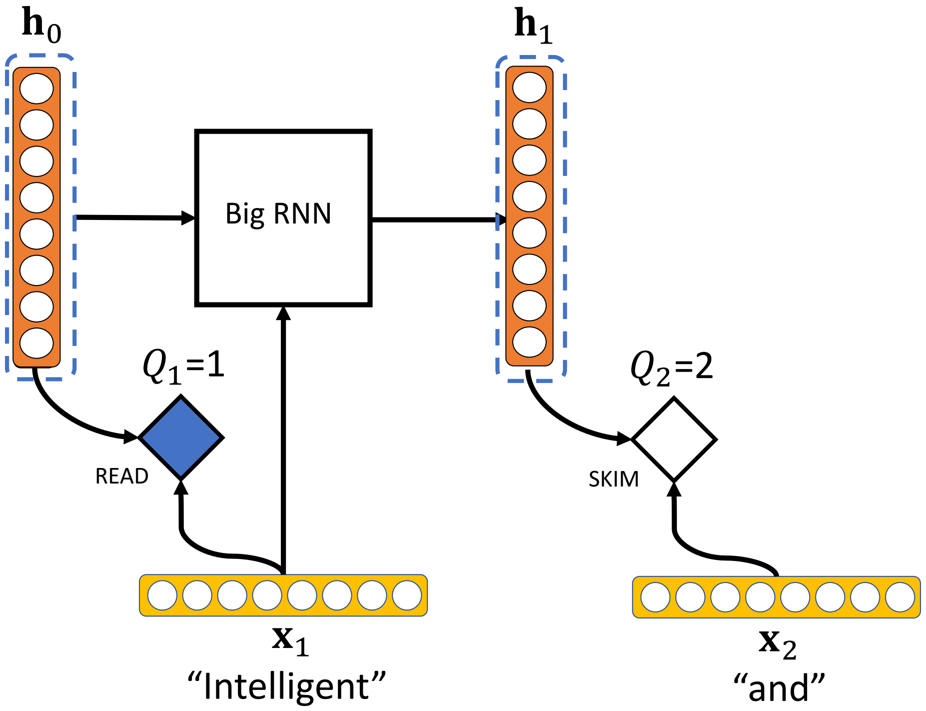
Sentiment Classification with Skim-RNN



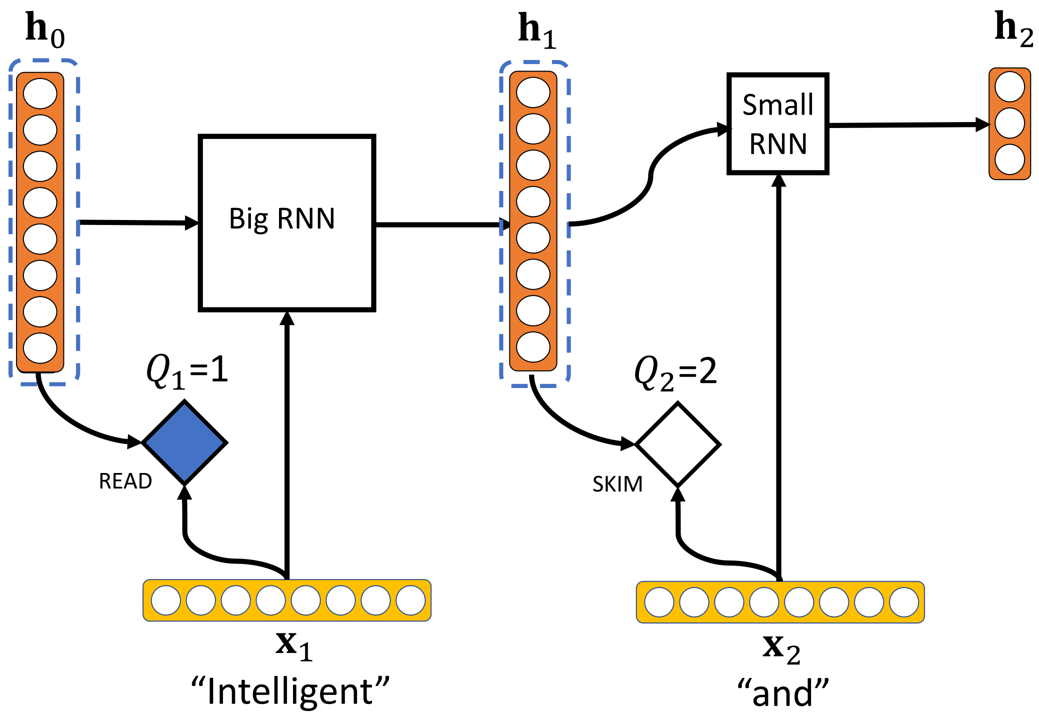
Sentiment Classification with Skim-RNN



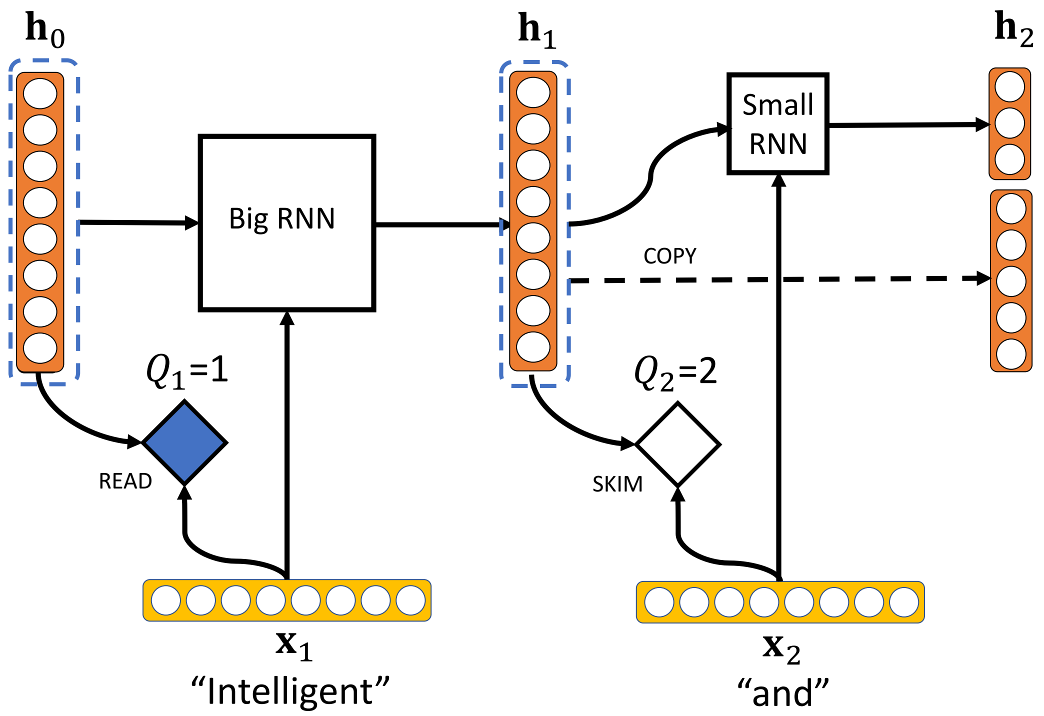
Sentiment Classification with Skim-RNN



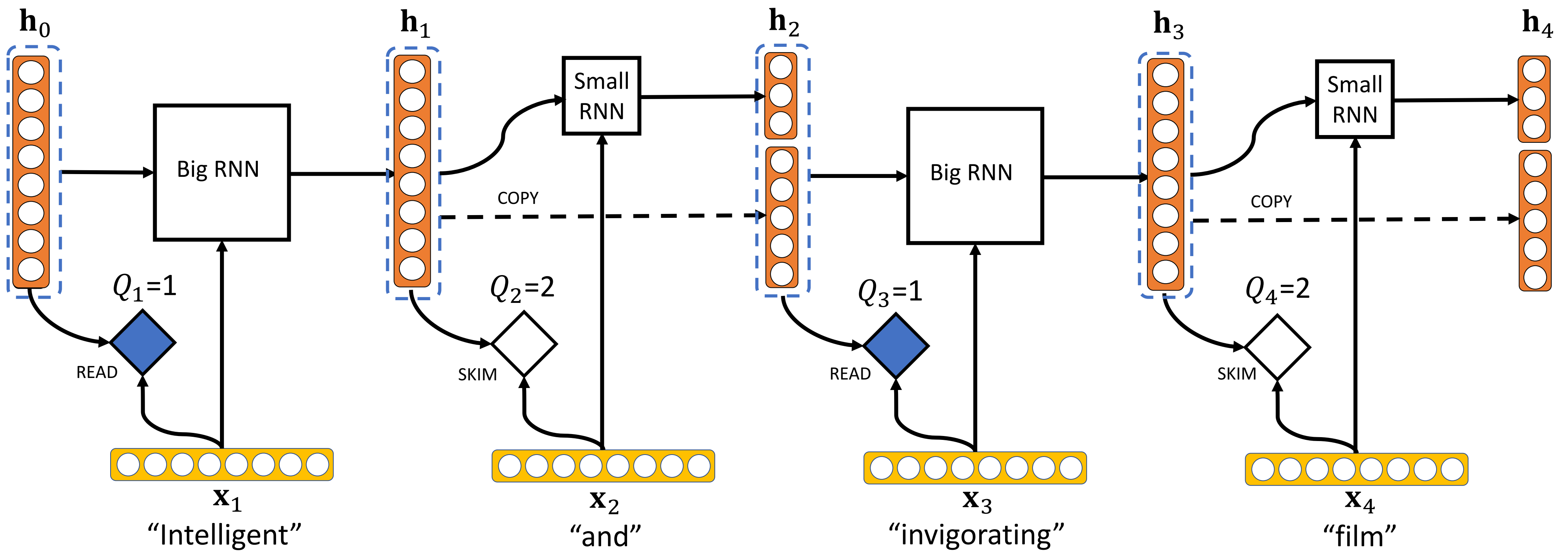
Sentiment Classification with Skim-RNN



Sentiment Classification with Skim-RNN



Sentiment Classification with Skim-RNN



Skim-RNN

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

Skim-RNN

- Consists of two RNNs:
  - **Big RNN**: hidden state size = d
  - **Small RNN**: hidden state size = d'
  - $d \gg d'$ (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.

Skim-RNN

- Consists of two RNNs:
  - **Big RNN**: hidden state size = d
  - **Small RNN**: hidden state size = d'
  - $d \gg d'$ (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.
  - $O(d^2) \gg O(d'd)$
- Dynamically makes decision on which size of RNN to use

How to train?

- Decisions (which RNN to use) are so non-differentiable

- 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

- Start with soft decision (attention) **p**

$$\mathbf{r}_t^i = \frac{\exp((\log(\mathbf{p}_t^i) + g_t^i)/\tau)}{\sum_j \exp((\log(\mathbf{p}_t^j) + g_t^j)/\tau)} \qquad\qquad \mathbf{h}_t = \sum_i \mathbf{r}_t^i \tilde{\mathbf{h}}_t^i$$

- Slowly decrease temperature (τ), making the distribution more discrete
- Sampling with the attention weights (**r**) approximate true distribution
- Reparameterization (g) allows differentiation with stochasticity

LSTM-Jump (Yu et al., 2017)

- Orthogonal
  - *Skipping*: you decide to skip next words / sentences before reading at all.
  - *Skimming*: you use very small RNN to read unimportant words fast.
  - Both can be combined.
- Concurrent work

LSTM-Jump (Yu et al., 2017)

- Orthogonal
  - *Skipping*: you decide to skip next words / sentences before reading at all.
  - *Skimming*: you use very small RNN to read unimportant words fast.
  - Both can be combined.
- Concurrent work
- Skim-RNN has output for every time step
  - Useful for applications that need output every time.
  - Easy to replace existing RNNs.
- LSTM-Jump has GPU advantage

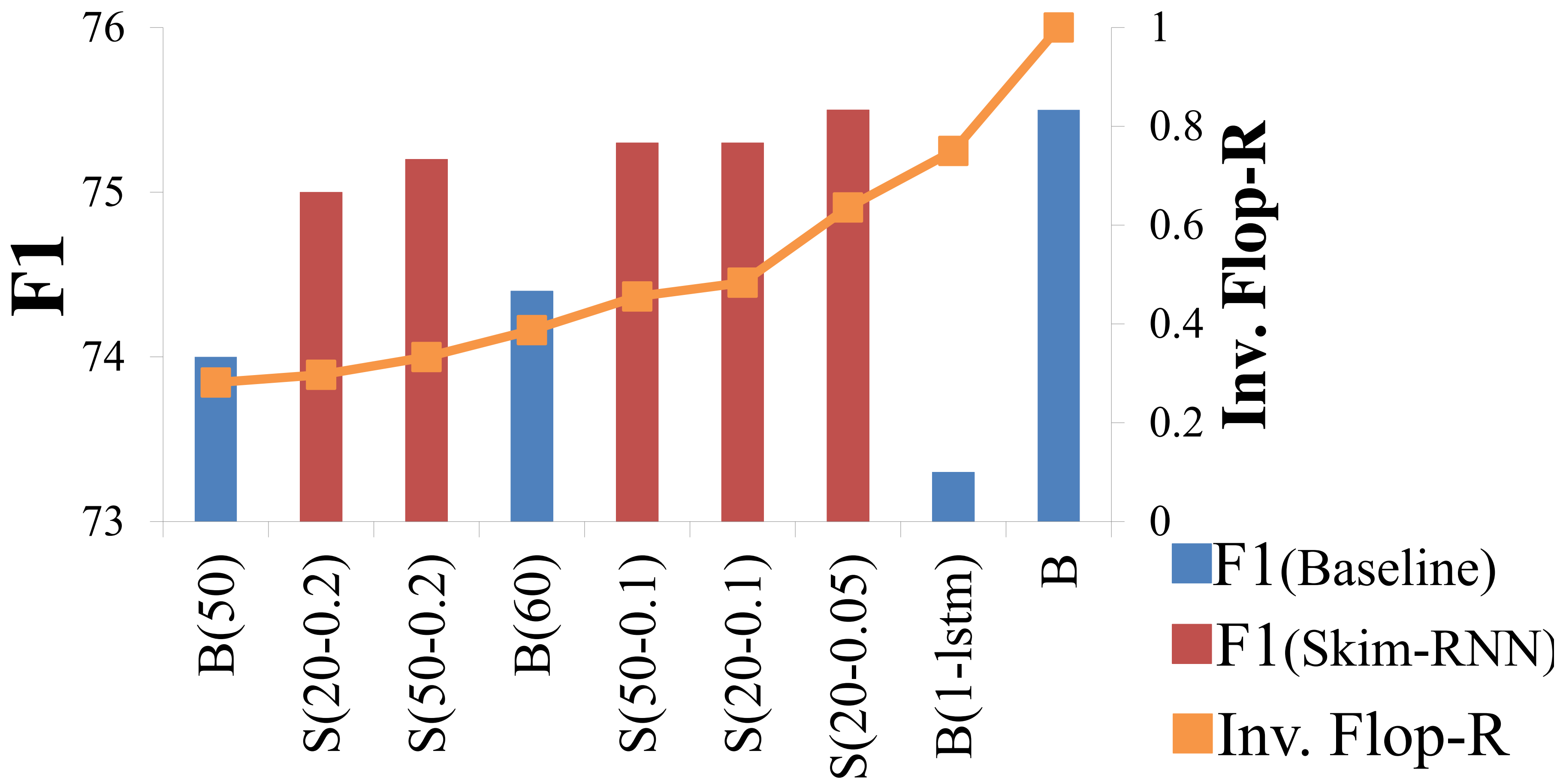
Classification Results

LSTM Model	d'/γ	SST Acc	SST Sk	SST Flop-r	SST Sp	Rotten Tomatoes Acc	Rotten Tomatoes Sk	Rotten Tomatoes Flop-r	Rotten Tomatoes Sp	IMDb Acc	IMDb Sk	IMDb Flop-r	IMDb Sp	AGNews Acc	AGNews Sk	AGNews Flop-r	AGNews Sp
Standard		86.4	-	1.0x	1.0x	82.5	-	1.0x	1.0x	91.1	-	1.0x	1.0x	93.5	-	1.0x	1.0x
Skim	5/0.01	**86.4**	58.2	2.4x	1.4x	**84.2**	52.0	2.1x	1.3x	89.3	79.2	4.7x	2.1x	**93.6**	30.3	1.4x	1.0x
Skim	10/0.01	85.8	61.1	2.5x	1.5x	82.5	58.5	2.4x	1.4x	**91.2**	83.9	5.8x	2.3x	93.5	33.7	1.5x	1.0x
Skim	5/0.02	85.6	62.3	2.6x	1.5x	81.8	63.7	2.7x	1.5x	88.7	63.2	2.7x	1.5x	93.3	36.4	1.6x	1.0x
Skim	10/0.02	**86.4**	68.0	3.0x	1.7x	82.5	63.0	2.6x	1.5x	90.9	90.7	9.5x	2.7x	92.5	10.6	1.1x	0.8x
LSTM-Jump		-	-	-	-	79.3	-	-	1.6x	89.4	-	-	1.6x	89.3	-	-	1.1x
SOTA		89.5	-	-	-	83.4	-	-	-	94.1	-	-	-	93.4	-	-	-

Question Answering Results

Model	γ	F1	EM	Sk	Flop-r
LSTM+Att (1 layer)	-	73.3	63.9	-	1.3x
LSTM+Att ($d = 50$)	-	74.0	64.4	-	3.6x
LSTM+Att	-	75.5	**67.0**	-	1.0x
Sk-LSTM+Att ($d' = 0$)	0.1	**75.7**	66.7	37.7	1.4x
Sk-LSTM+Att ($d' = 0$)	0.2	75.6	66.4	49.7	1.6x
Sk-LSTM+Att	0.05	75.5	66.0	39.7	1.4x
Sk-LTM+Att	0.1	75.3	66.0	56.2	1.7x
Sk-LSTM+Att	0.2	75.0	66.0	76.4	2.3x
BiDAF ($d = 30$)	-	74.6	64.0	-	9.1x
BiDAF ($d = 50$)	-	75.7	65.5	-	3.7x
BiDAF	-	**77.3**	**67.7**	-	1.0x
Sk-BiDAF	0.01	76.9	67.0	74.5	2.8x
Sk-BiDAF	0.001	77.1	67.4	47.1	1.7x
SOTA (Wang et al., 2017)		79.5	71.1	-	-

Comparing F1 & FLOP across diff configs.



Visualization on IMDb Sentiment Classification

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

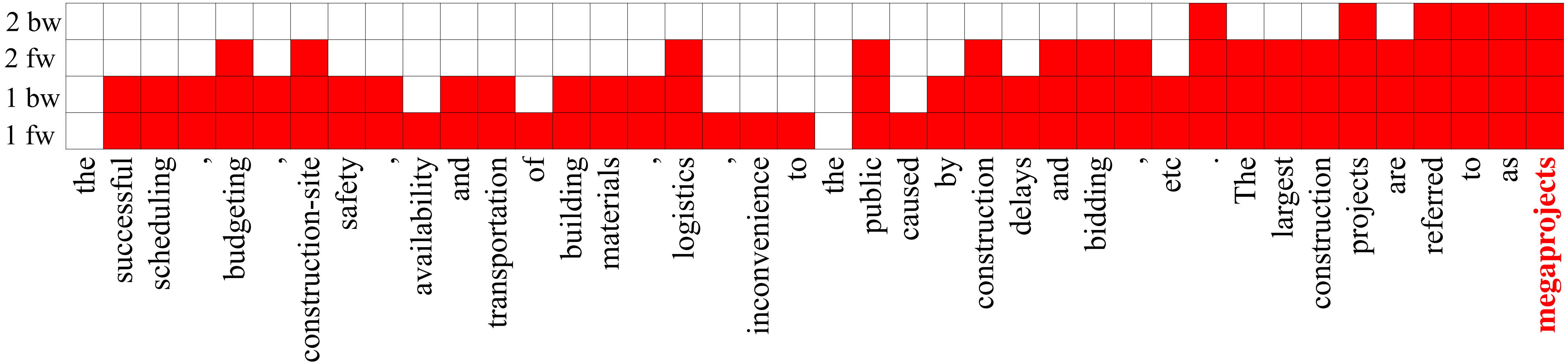
*Black words are skimmed (small RNN), blue words are fully read.

Visualization on Stanford Question Answering Dataset

Q	What is one straightforward case of a probabilistic test?
C	A particularly **simple** example of a **probabilistic test** is the **Fermat primality** **test**, which **relies** on the fact (**Fermat**'s little theorem) that $np \equiv n$ (**mod** 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 (**mod** 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.

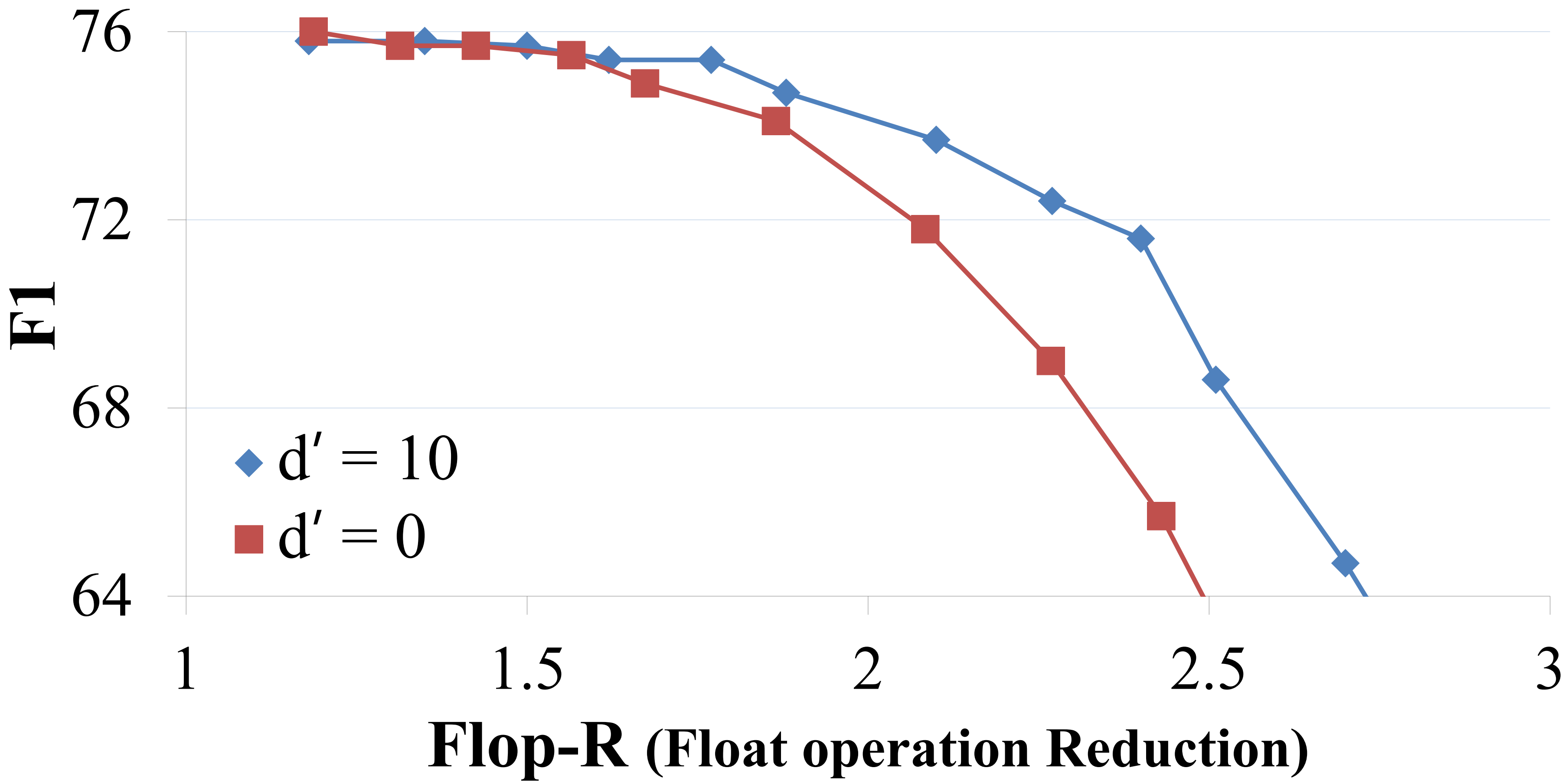
*Black words are skimmed (small RNN), blue words are fully read.

Layer-wise Skim Visualization (SQuAD)

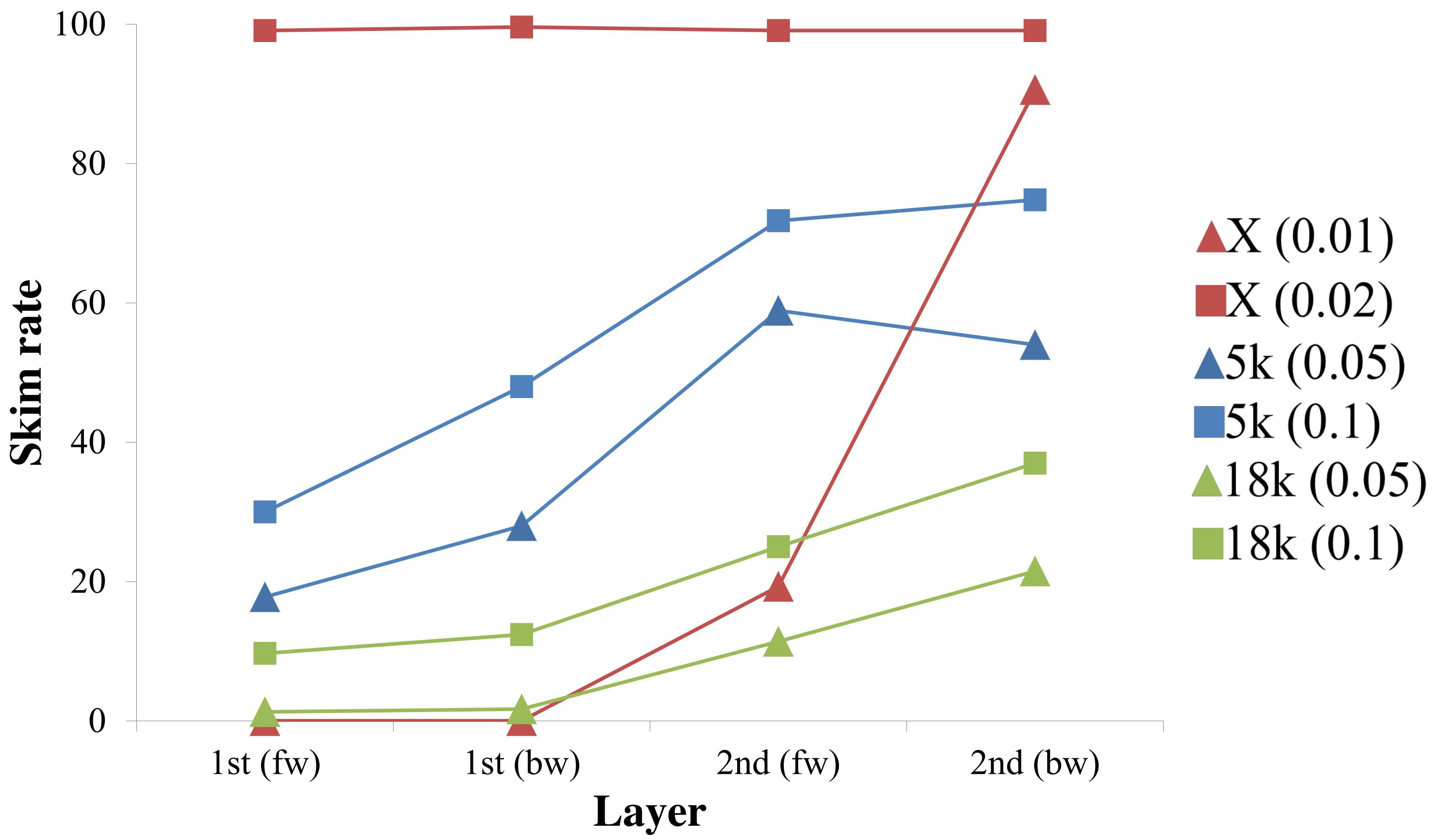


Most RNN steps at higher layer is redundant!

Dynamically controlling # of FLOP



Stability of Pretraining



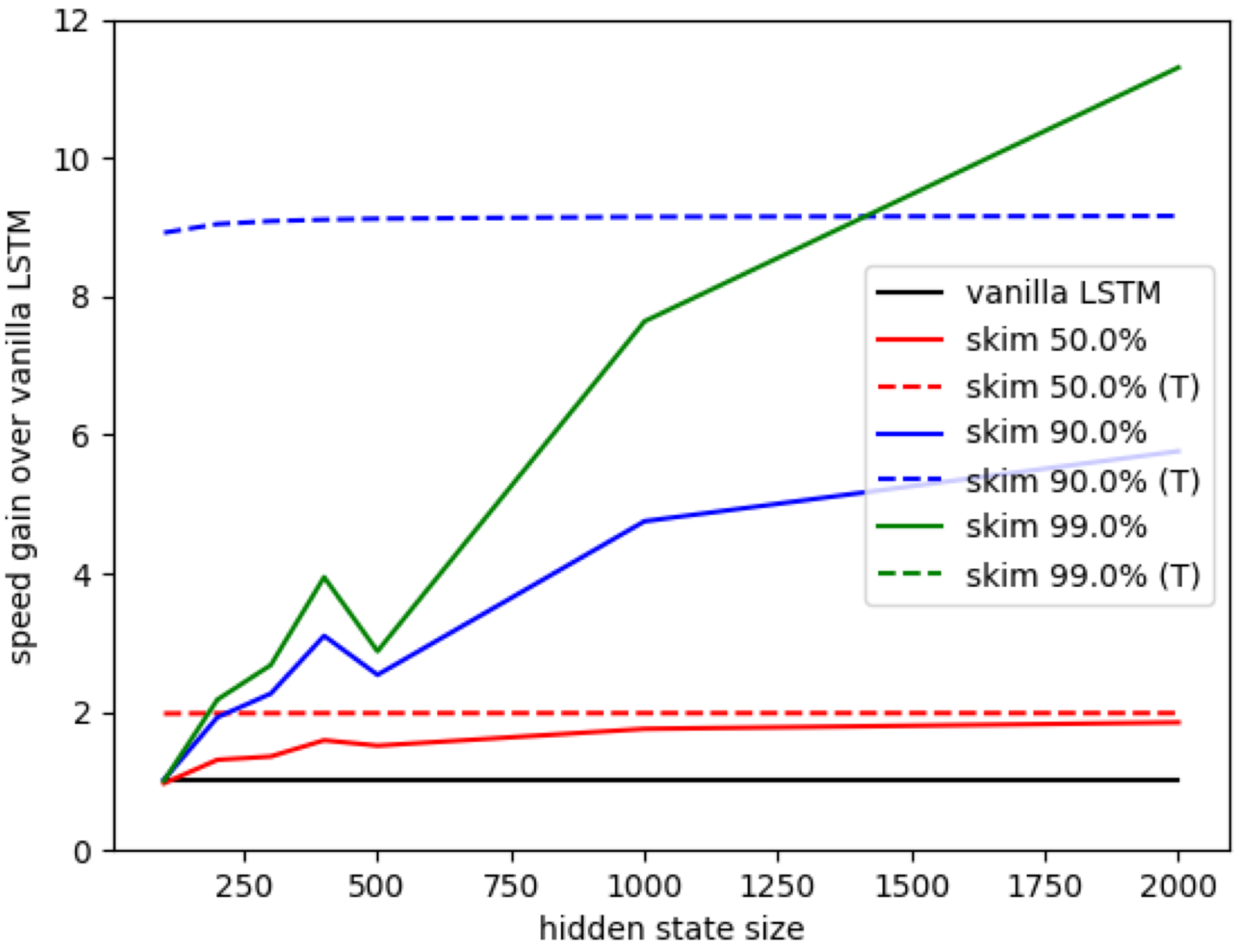
FLOP = Speed?

- On GPU: empirically NO
- On CPU: conditionally YES, with low-level programming

TensorFlow or PyTorch?

- **NumPy** is faster for small matrices
  - On CPUs (NumPy has no GPU compatibility)
  - Batch size = 1 (latency, not throughput)
  - d < 220 for TensorFlow
  - d < 700 for PyTorch
- TensorFlow and PyTorch seem to have more overheads
- All benchmarks are based on NumPy

Theoretical and Actual Speed Gains on NumPy



Conclusion

- **Skim-RNN**: switching between two different-size RNNs with shared hidden state.
  - Can be generalized to multiple RNNs.
- Speed gain can be substantial.
- More beneficial with larger hidden state size.
- Especially useful for **latency.**
  - To get throughput advantage, will need to go low-level.

Future Work

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

Thanks!

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