DEEP CONTEXTUALIZED WORD REPRESENTATIONS

A (surprisingly) simple method for task-specific tuning of language embeddings

June 4, 2018

Chris Laver
RBC

1) **Overview**
2) Background
3) Model Formulation & Architecture
4) Results
5) Discussion

ELMo:
Embeddings from **L**anguage **Mo**dels

Deep Contextualized Word Representations

Neural Network

$y = f(word, context)$

Words as fundamental semantic unit

Embedding

The Paper in Question...

Deep Contextualized Word Representations
https://arxiv.org/abs/1802.05365

Matthew E. Peters†, Mark Neumann†, Mohit Iyyer†, Matt Gardner†, Christopher Clark∗, Kenton Lee∗, Luke Zettlemoyer†∗

†Allen Institute for Artificial Intelligence
∗Paul G. Allen School of Computer Science & Engineering, University of Washington

1) Overview
2) **Background**
3) Model Formulation & Architecture
4) Results
5) Discussion

Language Embeddings

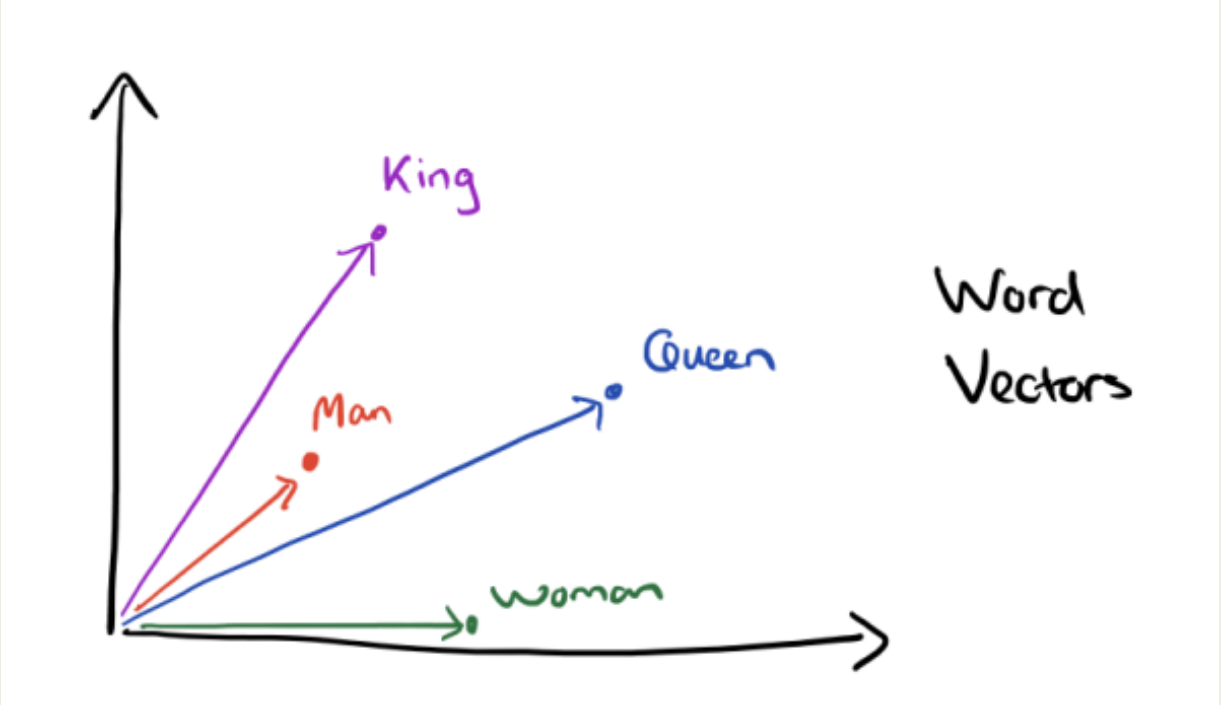
Natural language being hard to work with directly (dictionary cardinality, sparse representation, lack of operators), it's preferable to transform the representation to something more machine-readable

Ideally,

- Efficient (dense) representation
- Context-aware
- Valid vector math operators (semantic)

We have a number of good context-insensitive options (e.g. Word2Vec, GloVe, fastText)

Word2Vec



Mikolov, Chen, Corrado, Dean. **Efficient Estimation of Word Representations in Vector Space** (2013)

Word Embeddings in Context

"Chris attended the **play** at the Shaw Festival."

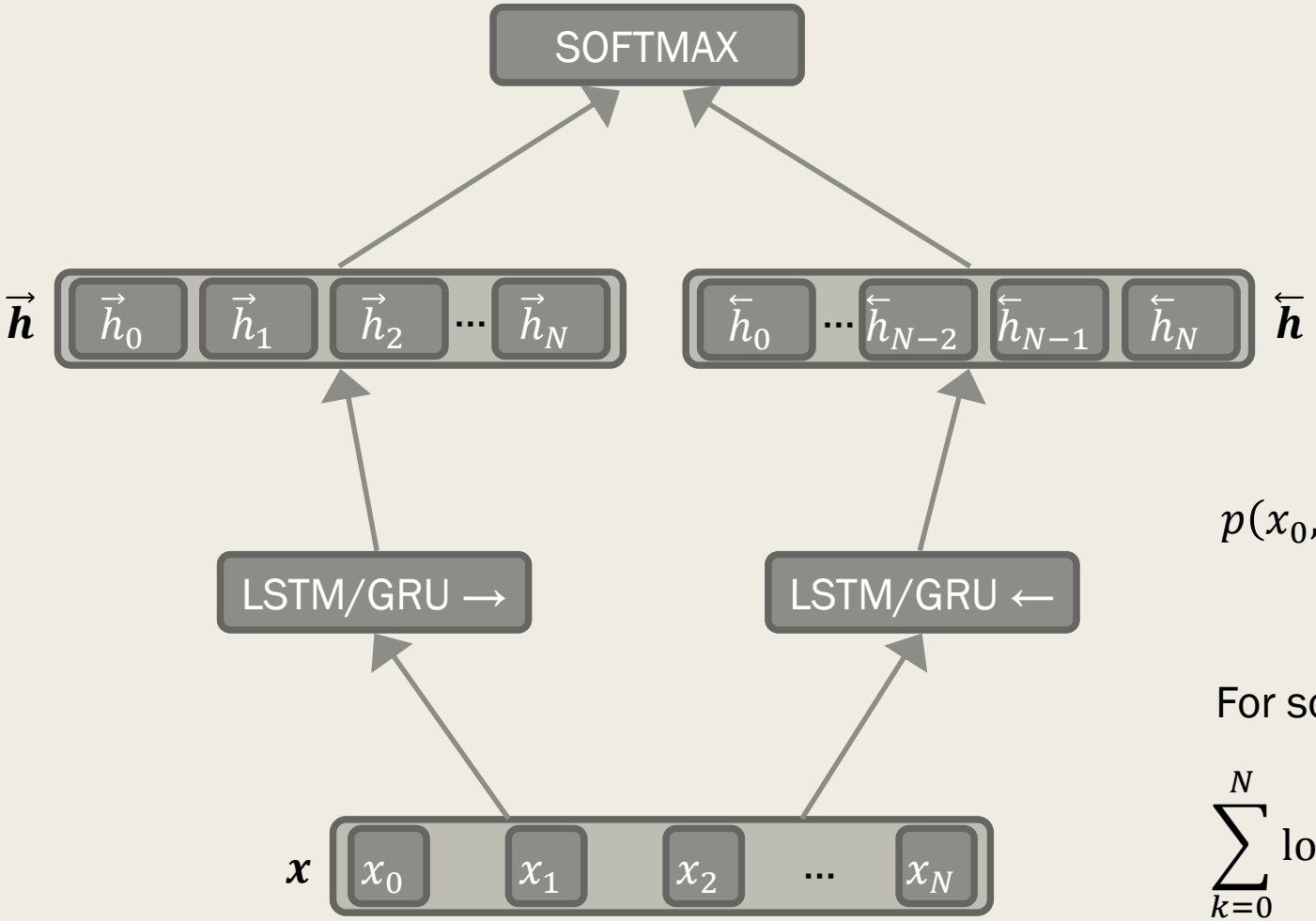
"Chris didn't want to **play** bridge."

In a context-insensitive embedding model, $f(\text{play})$ is the same for each sentence, despite the semantics being significantly different.

In a context-aware embedding model, we augment the word in question with the the context in which it appears.

$$y = f(\text{word}, \text{context})$$

A Language Model Architecture

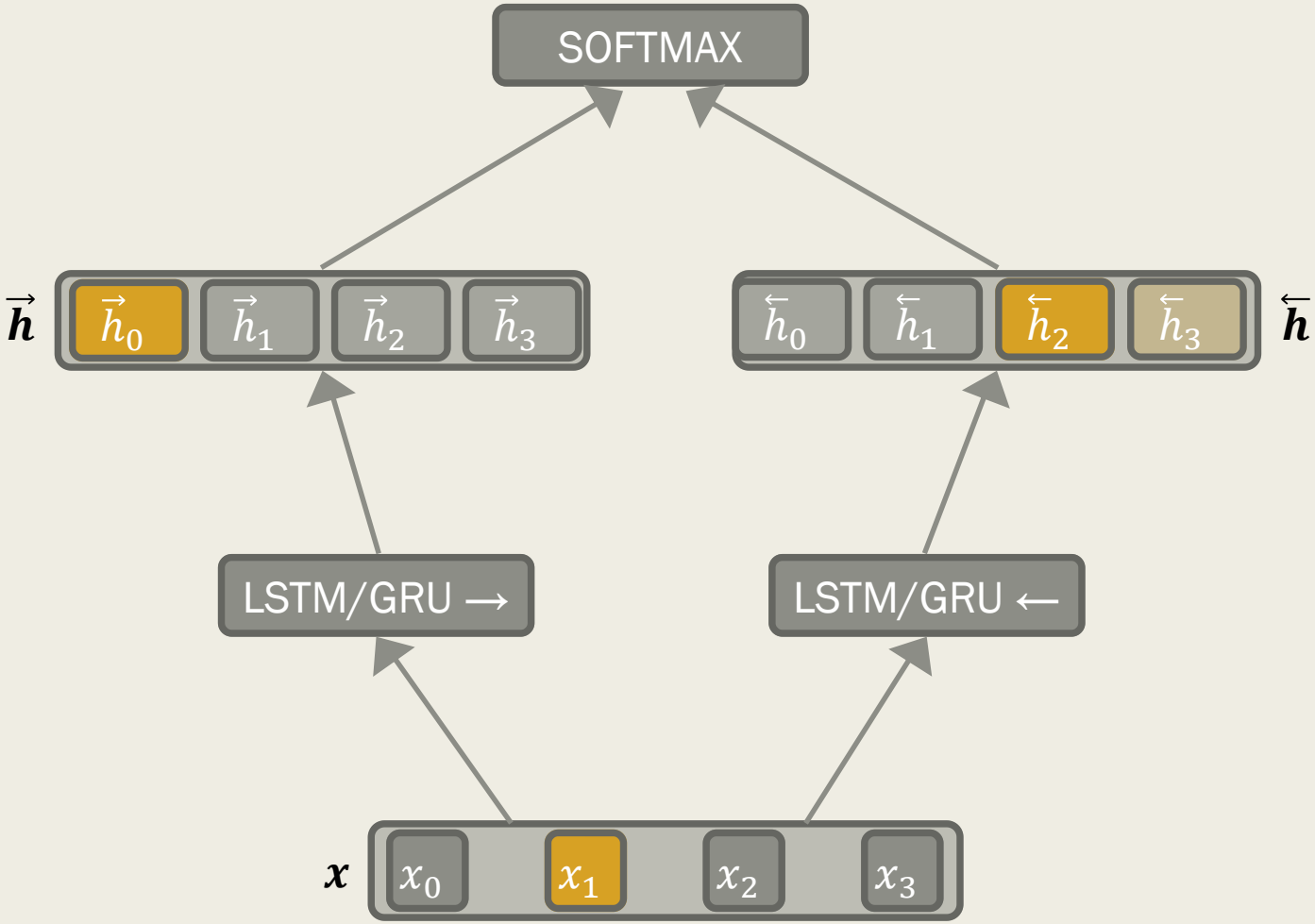


$$p(x_0, x_1, \dots, x_N) = \prod_{k=0}^{N} p(x_k | x_0, x_1, \dots, x_{k-1})$$

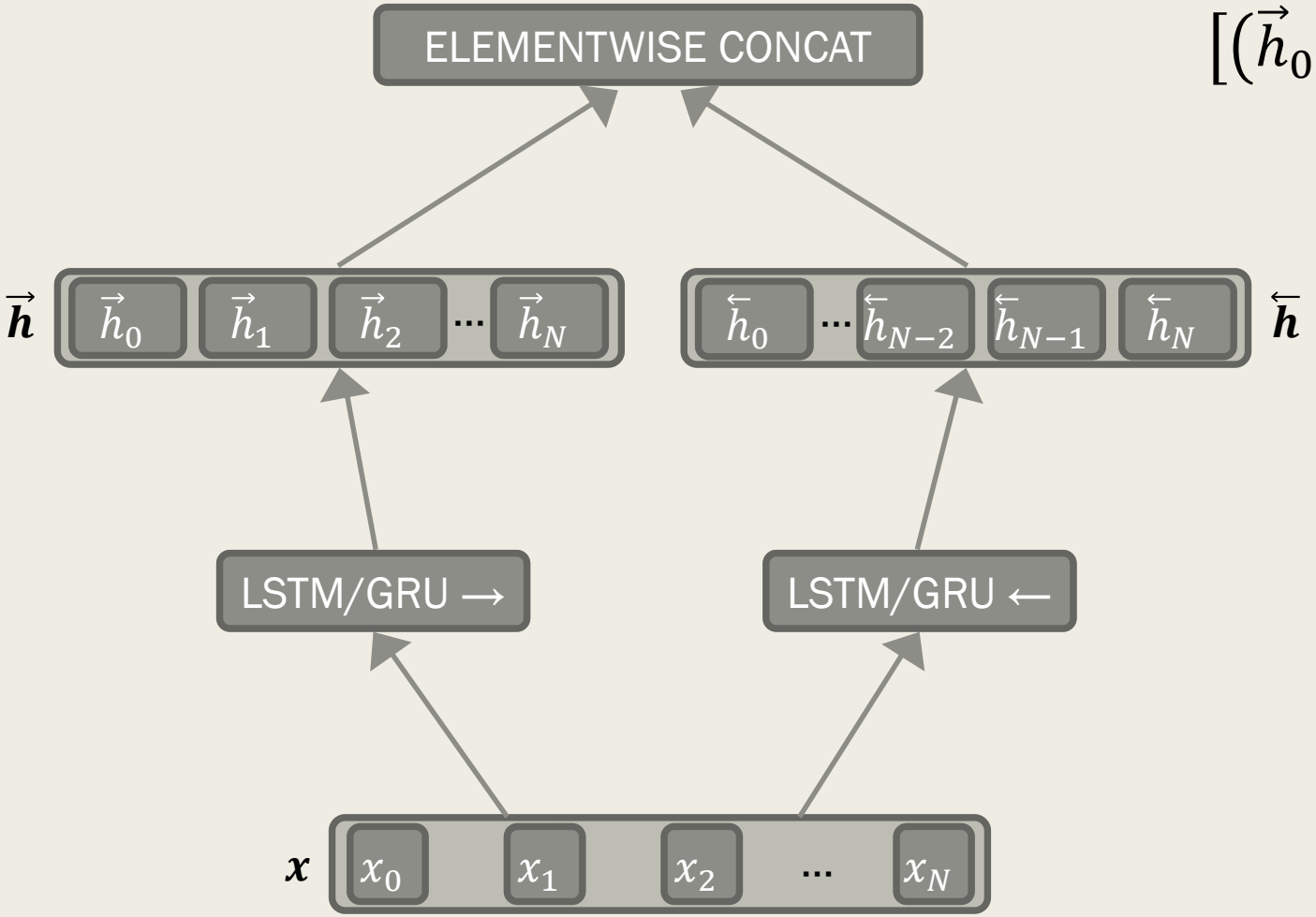
For some model parameterized by θ, maximize

$$\sum_{k=0}^{N} \log p(x_k | x_0, x_1, \dots, x_{k-1};\ \theta)$$

Training Example



A Language Model Architecture



$$[(\vec{h}_0;\overleftarrow{h}_0),(\vec{h}_1;\overleftarrow{h}_1),\dots,(\vec{h}_N;\overleftarrow{h}_N)]$$

$$y_k = (\vec{h}_k;\overleftarrow{h}_k)$$

1) Overview
2) Background
3) **Model Formulation & Architecture**
4) Results
5) Discussion

ELMo at a Glance

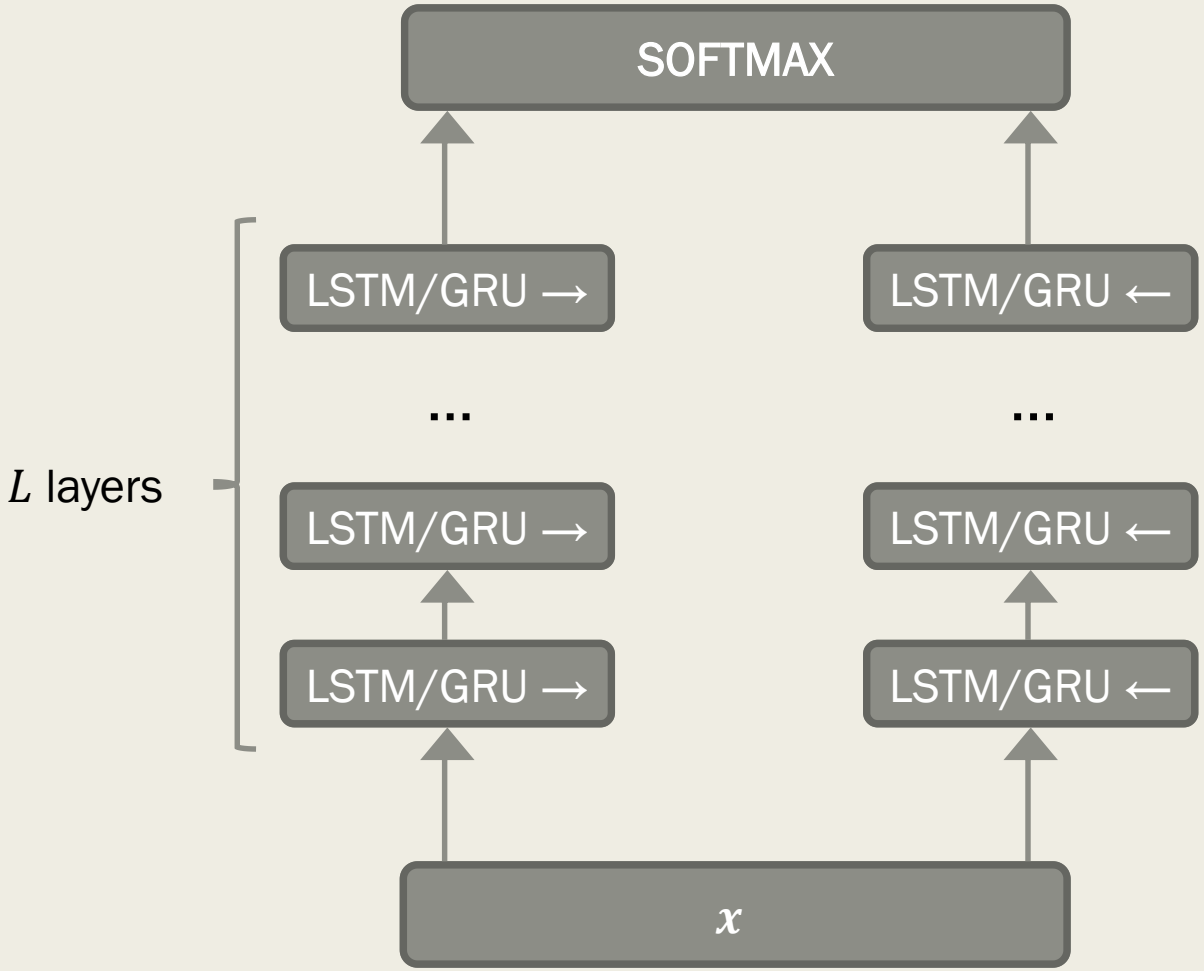
- Starting from a standard language model architecture…

  I. *Use multiple layers of recurrent units in the encoder*

  II. *Keep all the internal layer representations, in addition to the final recurrent layer*

  III. *For any downstream task, create the task-specific embeddings as a linear combination of all the internal layer representations*

Architecture



The representation for input token k is now

$$R_k = \left\{x_k, \vec{h}_{k,j}, \overleftarrow{h}_{k,j} \middle| j = 1, \dots, L\right\}$$

$$= \left\{h_{k,j} \middle| j = 0, \dots, L\right\}$$

given $h_{k,0} = x_k$ and $h_{k,j} = \left[\vec{h}_{k,j}; \overleftarrow{h}_{k,j}\right]$ otherwise

We can combine the internal representations as a (trainable, weighted) linear combination

$$\text{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}$$

s^{task} are softmax-normalized weights

Is this better...?

$$\mathrm{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}$$

The base case, when $\gamma^{task} = 1$ and $s_L^{task} = 1$, is equivalent to the vanilla language model presented earlier... so at a minimum, this is at least as representative as the vanilla model.

“Previous work has also shown that different layers of deep biRNNs encode different types of information” (related work section of paper)

ELMo in practice

In any situation where a language representation e would be used, we can substitute

$$[e; \mathrm{ELMo}^{task}]$$

To set γ^{task} and $\boldsymbol{s}^{task}$ we first freeze the parameters of the language model used to generate R_k, then train γ^{task} and $\boldsymbol{s}^{task}$ as parameters of the task-specific model in which we're including ELMo^{task}

Inputs

(Note: the technique discussed here is used in the implementation, but was not an innovation from this work)

- Using a one-hot encoding of words as input is still awkward
  - English, for example, has about 175K unique words, though only about 3K of them are common

Instead of words, use sub-word units (e.g. characters), and pass them through a CNN before presenting the now-dense representations to the architecture described here

- Two primary benefits
  - More compact representation
  - Out-of-vocabulary words can be handled

For more details on the implementation, consult
Kim, Jernite, Sontag, Rush. **Character-Aware Neural Language Models** (2015)

1) Overview
2) Background
3) Model Formulation & Architecture
4) **Results**
5) Discussion

Tasks

The work evaluated ELMo on some common NLP tasks, including

- Question answering
- Textual entailment
- Semantic role labeling
- Coreference resolution
- Named entity recognition
- Sentiment analysis

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

A Dubious Honour...

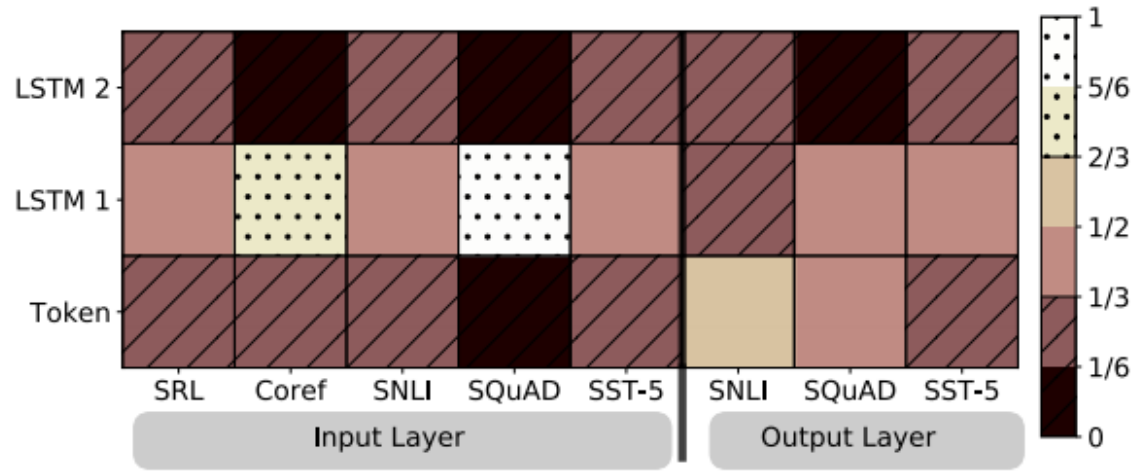


Figure 2: Visualization of softmax normalized biLM layer weights across tasks and ELMo locations. Normalized weights less then 1/3 are hatched with horizontal lines and those greater then 2/3 are speckled.

1) Overview
2) Background
3) Model Formulation & Architecture
4) Results
5) **Discussion**

Questions?

- Linear combination of internal representations… why linear?
- Input representations as words, characters, other sub-word units?
- Words as fundamental semantic unit?
- Measurement of quality for embedding tasks, related NLP tasks?

Related Work

Mikolov, Sutskever, Chen, Corrado, Dean. **Distributed representations of words and phrases and their compositionality** (2013)

Pennington, Socher, Manning. **GloVe: Global Vectors for Word Representation** (2014)

Joulin, Grave, Bojanowski, Mikolov. **Bag of Tricks for Efficient Text Classification** (2016)

Bahdanau, Cho, Bengio. **Neural Machine Translation by Jointly Learning to Align and Translate** (2014)

McCann, Bradbury, Xiong, Socher. **Learned in translation: Contextualized word vectors** (2017)

Lee, Cho, Hofmann. **Fully Character-Level Neural Machine Translation without Explicit Segmentation** (2017)