COLX-585: Trends in Computational Linguistics

Muhammad Abdul-Mageed

muhammad.mageed@ubc.ca

Natural Language Processing Lab

The University of British Columbia

Table of Contents

1. Token Representation
2. Forward and Backward LMs
3. Exploiting Pre-Trained LMs
4. ELMo
5. CoVe

Bidirectional LMs

Semi-supervised sequence tagging with bidirectional language models

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, Russell Power
Allen Institute for Artificial Intelligence
{matthewp,waleeda,chandrab,russellp}@allenai.org

Abstract

Pre-trained word embeddings learned from unlabeled text have become a standard component of neural network architectures for NLP tasks. However, in most cases, the recurrent network that operates on word-level representations to produce context sensitive representations is trained on relatively little labeled data. In this paper, we demonstrate a general semi-supervised approach for adding pre-trained context embeddings from bidirectional language models to NLP systems and apply it to sequence labeling tasks. We evaluate our model on two standard datasets for named entity recognition (NER) and chunking, and in both cases achieve state of the art results, surpassing previous systems that use other forms of transfer or joint learning with additional labeled data and task specific gazetteers.

current neural network (RNN) that encodes token sequences into a context sensitive representation before making token specific predictions (Yang et al., 2017; Ma and Hovy, 2016; Lample et al., 2016; Hashimoto et al., 2016).

Although the token representation is initialized with pre-trained embeddings, the parameters of the bidirectional RNN are typically learned only on labeled data. Previous work has explored methods for jointly learning the bidirectional RNN with supplemental labeled data from other tasks (e.g., Søgaard and Goldberg, 2016; Yang et al., 2017).

In this paper, we explore an alternate semi-supervised approach which does not require additional labeled data. We use a neural language model (LM), pre-trained on a large, unlabeled corpus to compute an encoding of the context at each position in the sequence (hereafter an *LM embedding*) and use it in the supervised sequence tagging model. Since the LM embeddings are used to compute the probability of future words in a neural LM, they are likely to encode both the semantic

Context-Independent Token Representation

- Given a sequence of tokens $(t_1, t_2, \ldots, t_N)$, we can compute context-independent token representation $\mathbf{x}_k$
- Concatenate a character based representation $\mathbf{c}_k$ (acquired via e.g., CNN) with a token embedding $\mathbf{w}_k$ (acquired from a look-up embedding table, e.g., word2vec)

1: Token Representation

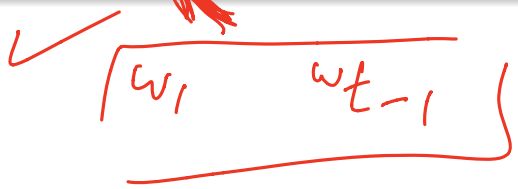
$$c_k = C(t_k; \theta_c)$$
$$w_k = E(t_k; \theta_w)$$
$$x_k = [c_k; w_k]$$

Statistical Language Models

- **A statistical LM**: The conditional probability of the next word given all the previous words.

Brewing a great cup of coffee

- coffee (w_t)
- Brewing a great cup of (w_1^{t-1})
- Brewing a great cup of coffee (W_1^T)



Forward Language Model

Given a sequence of N tokens, $(t_1, t_2, \ldots, t_N)$, a forward language model computes the probability of the sequence by modeling the probability of token t_k, given the history $(t_1, \ldots, t_{k-1})$:

2: Forward LM

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

t_1

$p(t_1 | \text{START})$

$p(t_2 | t_1)$

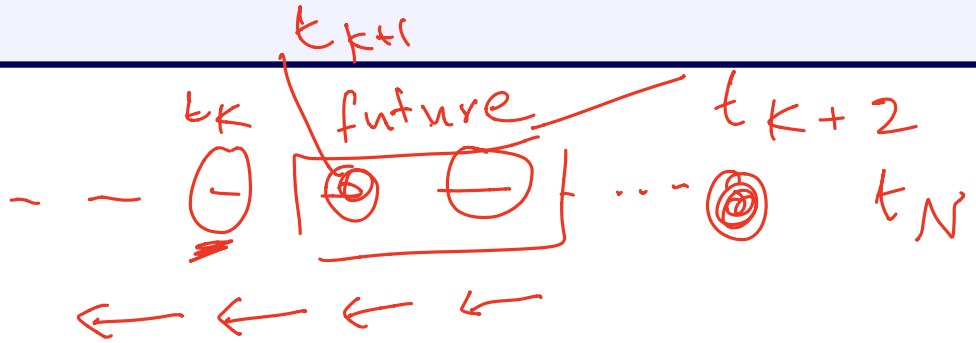
$p(t_3 | t_1, t_2)$

Backward Language Model

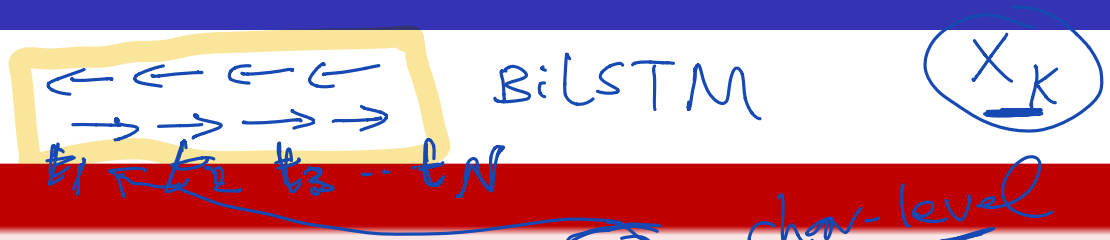
- A backward language model runs over the sequence in reverse, predicting the previous token given the future context:

3: Backward LM

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t+1, t+2, \ldots, t_N)$$



A Language Model with BiLSTM



BiLSMT LM

- Let's refer to token representation as $\mathbf{x}_k^{LM}$
- We will pass $\mathbf{x}_k^{LM}$ to a L-layered BiLSTM:
  1. Learn forward LM with a forward LSTM. In layer j, getting $\overrightarrow{\mathbf{h}}_{k,j}^{LM}$
  2. Learn backward LM with a backward LSTM. In layer j, getting $\overleftarrow{\mathbf{h}}_{k,j}^{LM}$

j=1

4: Bidirectional LM (Layer j)

$$\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}].$$

BiLSTM

Remember: Each LSTM takes token representation $\mathbf{x}_k^{LM}$ as input.

BiLSTM LM Architecture

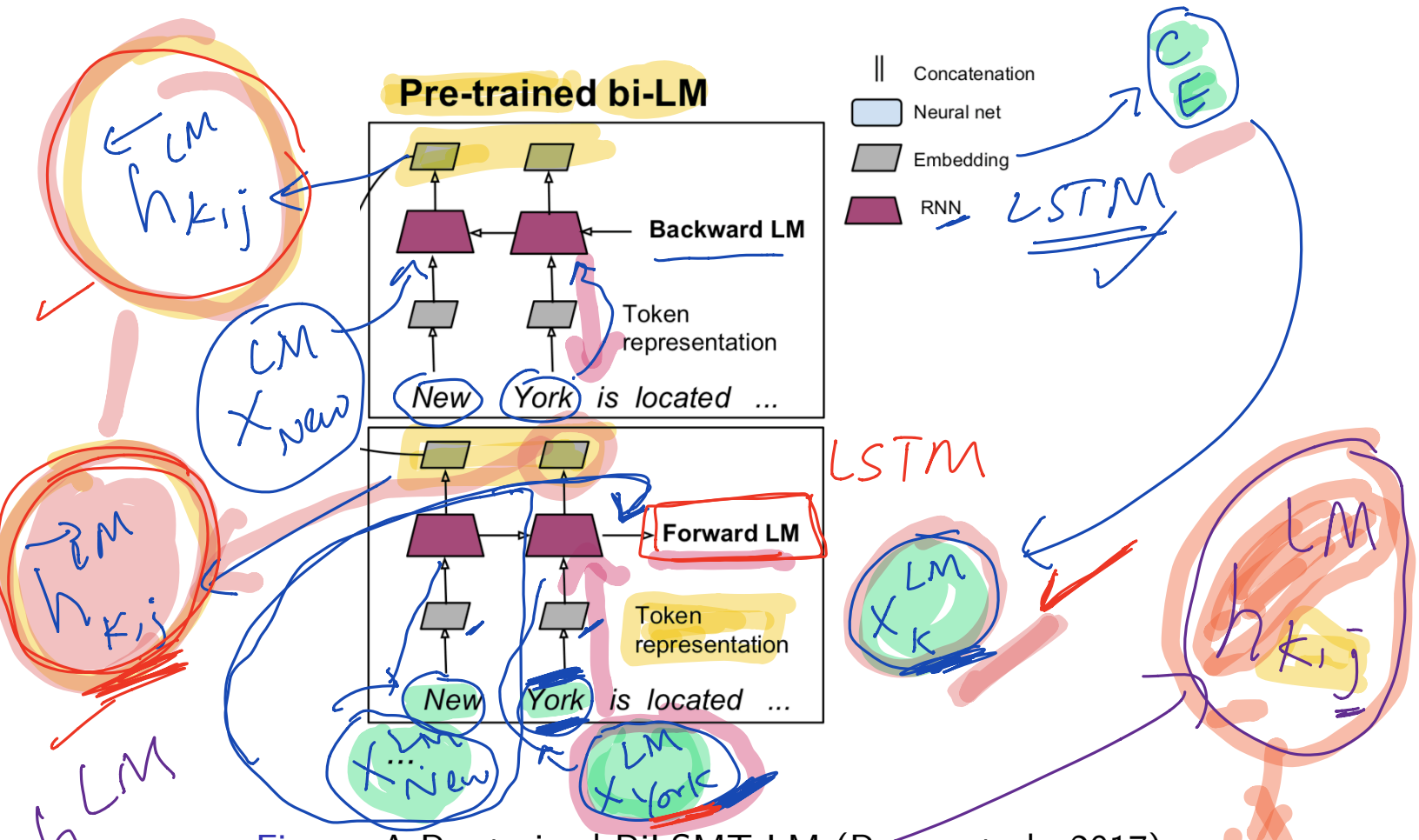


Figure: A Pre-trained BiLSMT LM (Peters et al., 2017)

Downstream Task Use

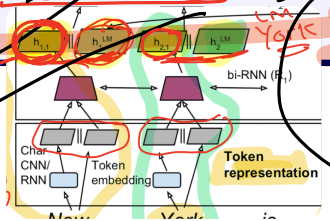
BiLSMT LM

- Pre-trained BiLSTM LM can be fine-tuned on a downstream task such as sequence tagging.
- Peters et al. (2017) **concatenate** the LM embeddings h^{LM} with the output of the first bidirectional RNN layer for the **sequence tagging model** ($\overrightarrow{\mathbf{h}}_{k,1}$ and $\overleftarrow{\mathbf{h}}_{k,1}$).

5: Bidirectional LM (Layer j)

$$\mathbf{h}_{k,1}^{LM} = [\overrightarrow{\mathbf{h}}_{k,1}, \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_{k}^{LM}]$$

$$= \mathbf{h}_{k,L}^{LM}$$



Exploiting Pre-Trained LM in Sequence Tagging

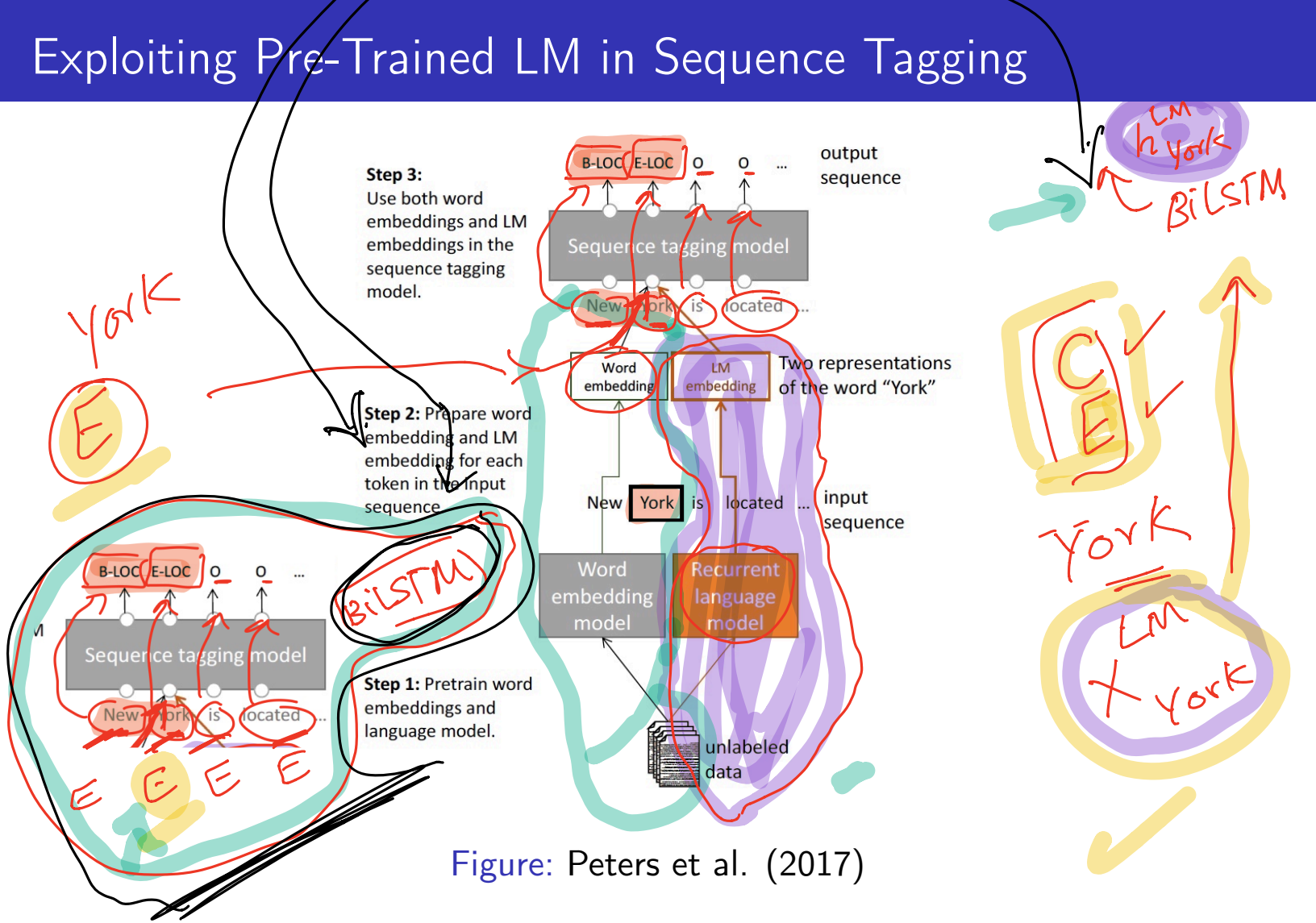


Figure: Peters et al. (2017)

Exploiting Pre-Trained LM in Sequence Tagging *Contd.*

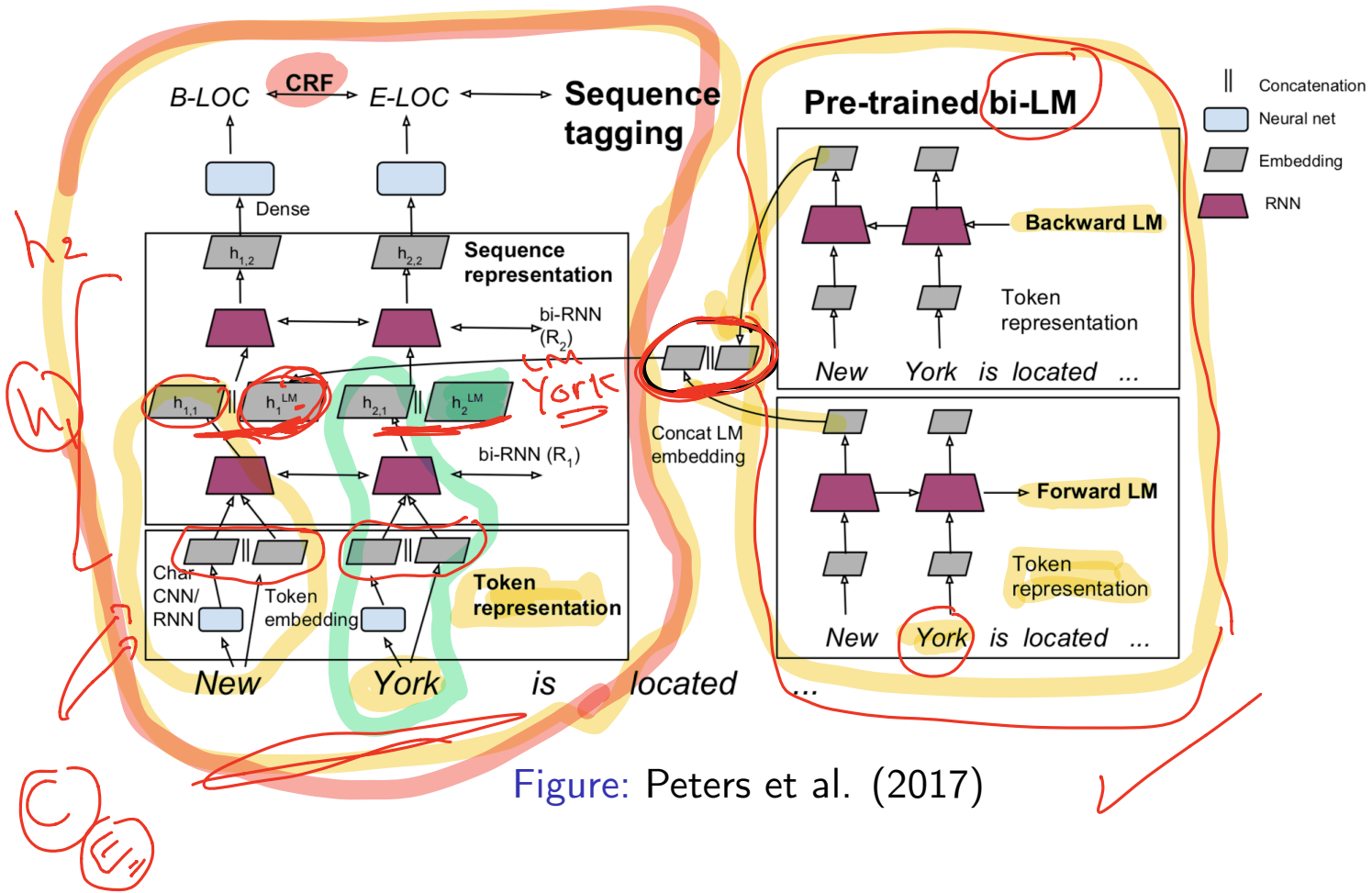


Figure: Peters et al. (2017)

ELMo

Deep contextualized word representations

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

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

Abstract

We introduce a new type of *deep contextualized* word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

guage model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

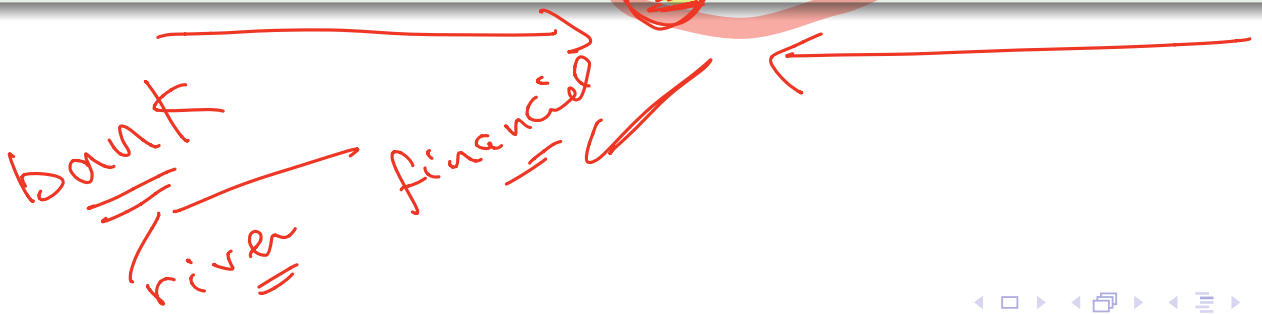
Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised

Word Meaning

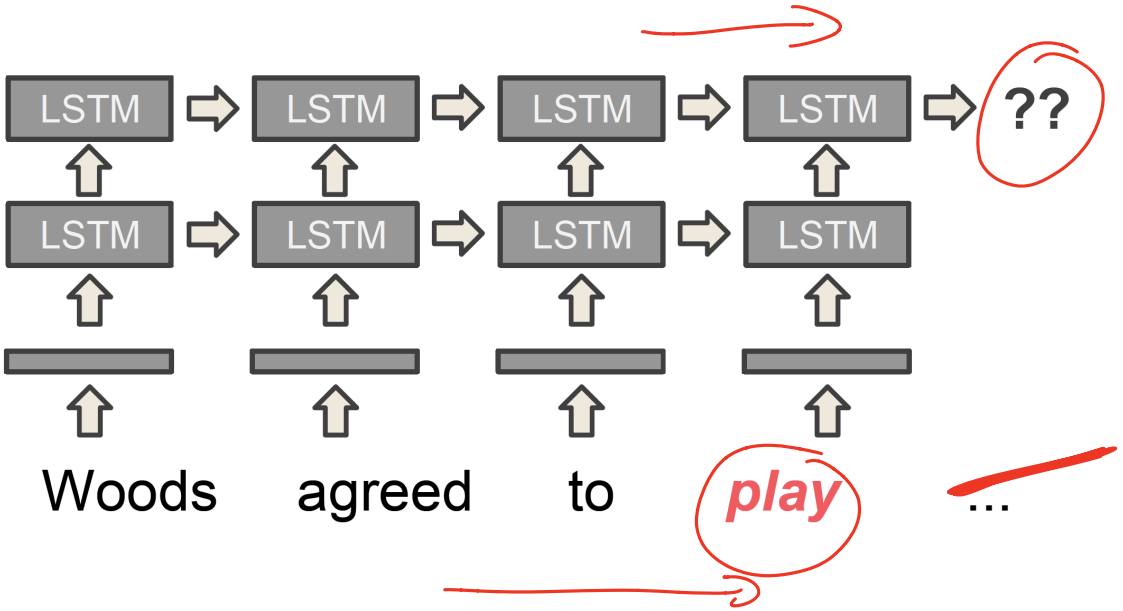
Distributional hypothesis: a word is characterized by the company it keeps

Consider the word "play"

1. The new-look play area is due to be completed by early spring 2010 .
2. Gerrymandered congressional districts favor representatives who play to the party base .
3. The children wouldn't stop play-fighting in the coffee shop .

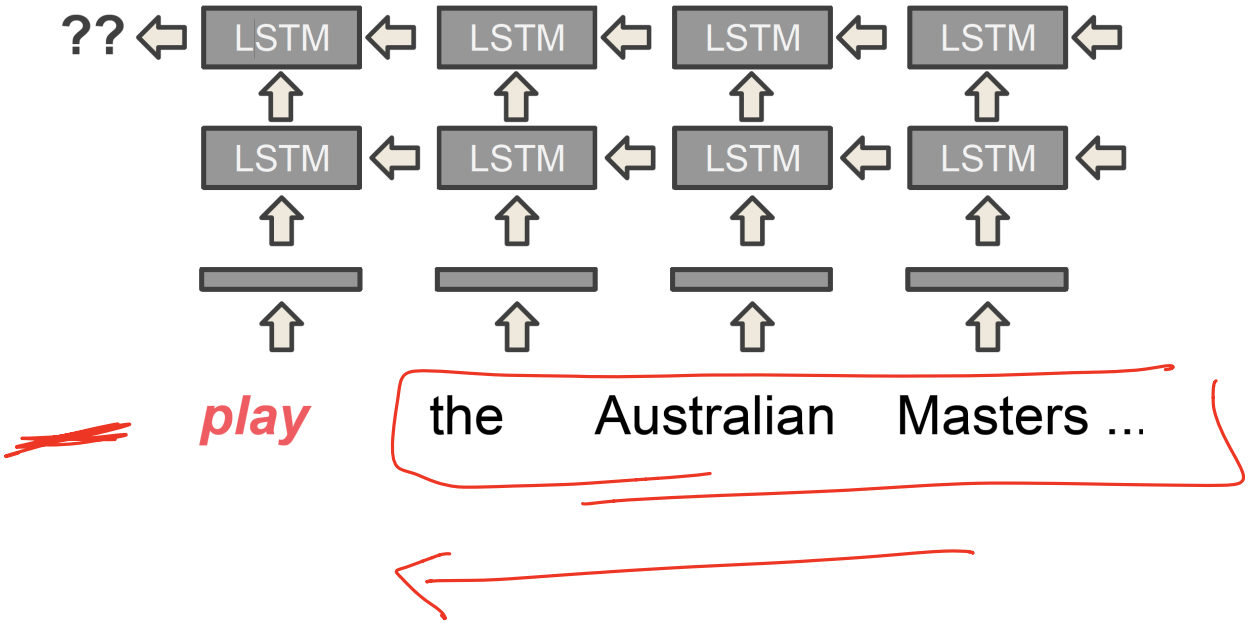


Forward LSTM

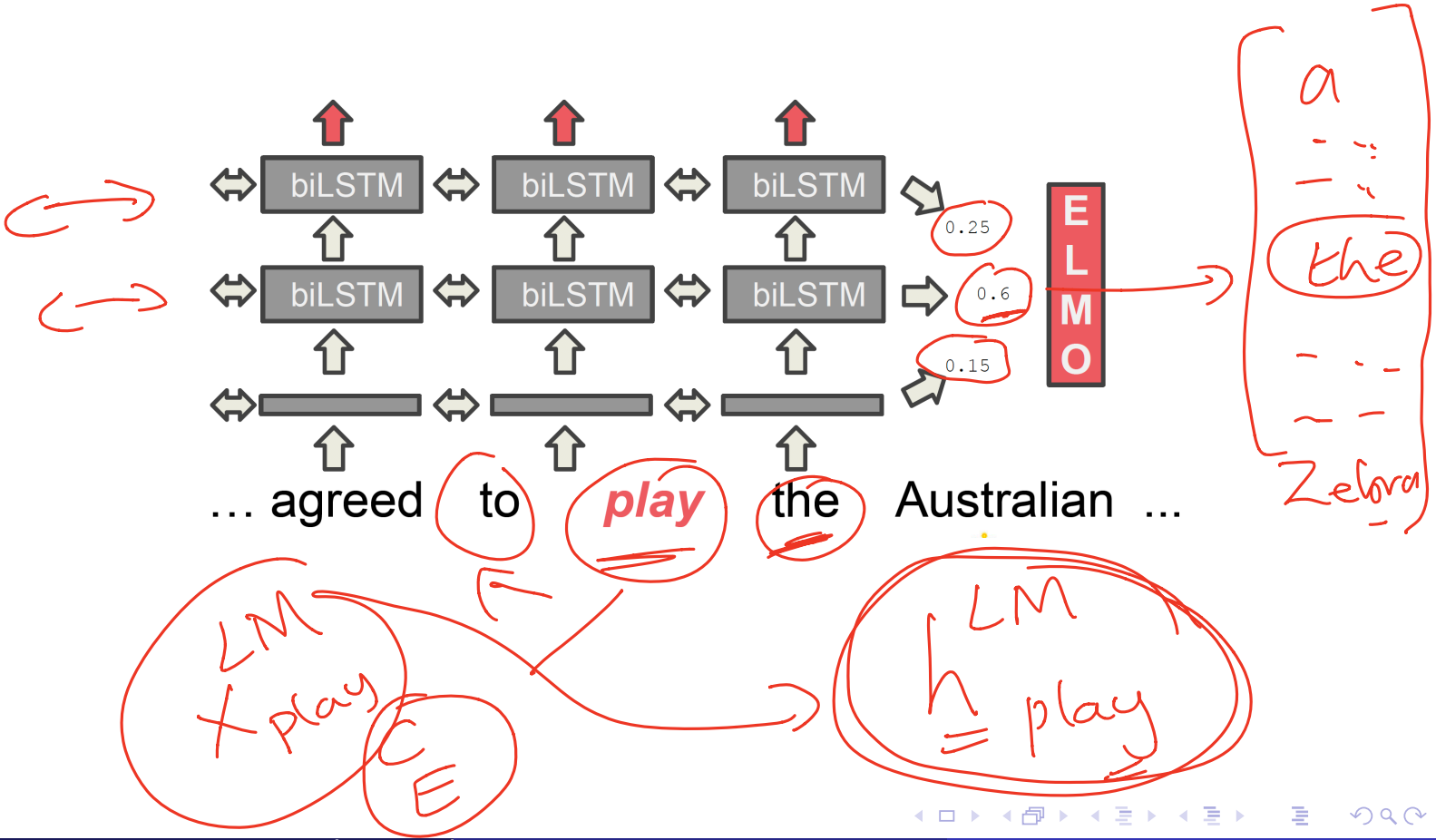


Mark Neumann

Backward LSTM



BiLSTM


biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
biLSTM
0.25
0.6
0.15
ELMO
… agreed to play the Australian ...
LM
play
LM
play

ELMo (Peters et al., 2018)

ELMo Representation R

- For each token t_k, an ELMo L-layer biLM computes a set of $2L + 1$ representations:

6: ELMo Representation

$$R_k = \{x_{k,0}^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} | j = 1, \ldots, L\}$$
$$= \{\mathbf{h}_{k,j}^{LM} | j = 0, \ldots, L, \}$$

where $x_{k,0}^{LM}$ = token embedding,
and $\mathbf{h}_{k,j}^{LM}$ = bidirectional language model =

$$[\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}].$$

Fine-Tuning ELMo

Using ELMo on a Supervised Task

- For inclusion in a **downstream model**, ELMo collapses all layers in R into a single vector, $\mathbf{ELMo}_k$
- In simplest case, ELMO selects top layer, $E(R_k) = \mathbf{h}_{k,L}^{LM}$
- More generally, **compute task specific weighting of all biLM layers**:

7: ELMo vector $\mathbf{ELMo}_k$

$$\mathbf{ELMo}_k = E(R_k; \theta^{task}) = \lambda^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

where: – s^{task}: softmax-normalized weights
– The scalar λ: allows the task model to scale the entire ELMO vector.

Fine-Tuning ELMo

Using ELMo on a Supervised Task

- For a given task, simply concatenate the ELMo vector $\mathbf{ELMo}_k^{task}$ with token representation $\mathbf{x}_k$
- Pass the ELMo-enhanced representation $[\mathbf{x}_k; \mathbf{ELMo}_k^{task}]$ to the task network (e.g. RNN)

ELMo Results

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 $\pm$ 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 $\pm$ 0.19	90.15	92.22 $\pm$ 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 $\pm$ 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

Figure: Peters et al. (2018)

For Your Reference: Modeling Agency and Sociality With ELMo

Model	Agency			Sociality		
	Accuracy	AUC	F1 Score	Accuracy	AUC	F1 Score
Baseline	0.7382	0.5	0.8494	0.5327	0.5	0.6951
LinSVM(BOW+TF-IDF)	0.8381	0.7713	0.8926	0.8963	0.8951	0.9037
CoVe (LSTM-A)	0.8726	0.8081	0.7345	0.9181	0.9180	0.9127
ELMo (LSTM-A)	**0.8797**	**0.8444**	**0.9185**	**0.9313**	**0.9309**	**0.9342**
ULMFiT (LSTM)	0.8660	0.8277	0.9095	0.9237	0.9235	0.9285

Figure: Rajendran, Zhang, & Abdul-Mageed (2019)

What's in Agency I?

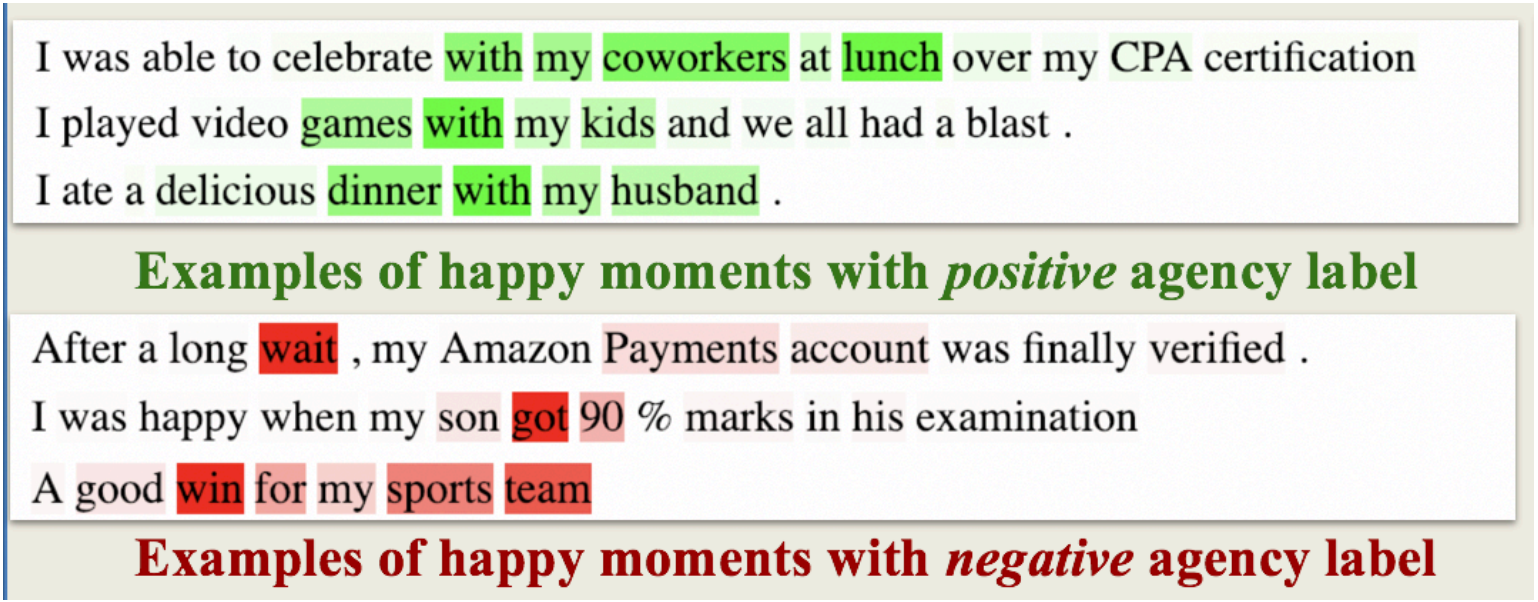


Figure: Attention heatmap for hand-picked **agency** examples

What's in Agency II?

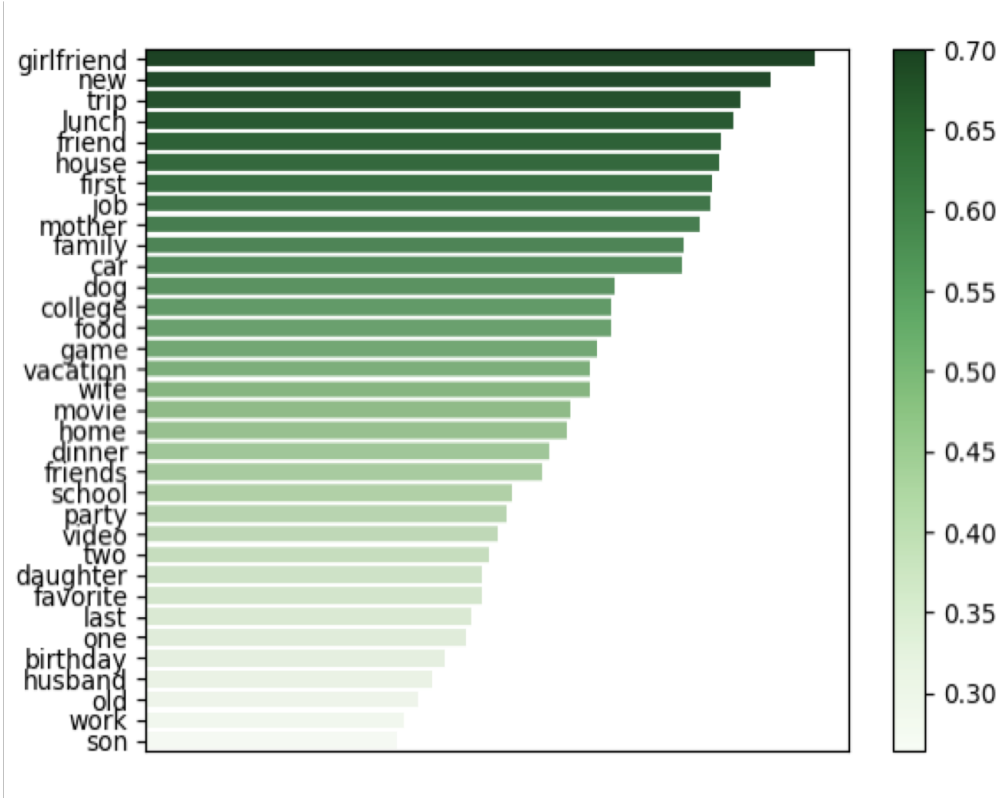


Figure: Attention-based feature ranking for **positive agency**

What's in Sociality I?

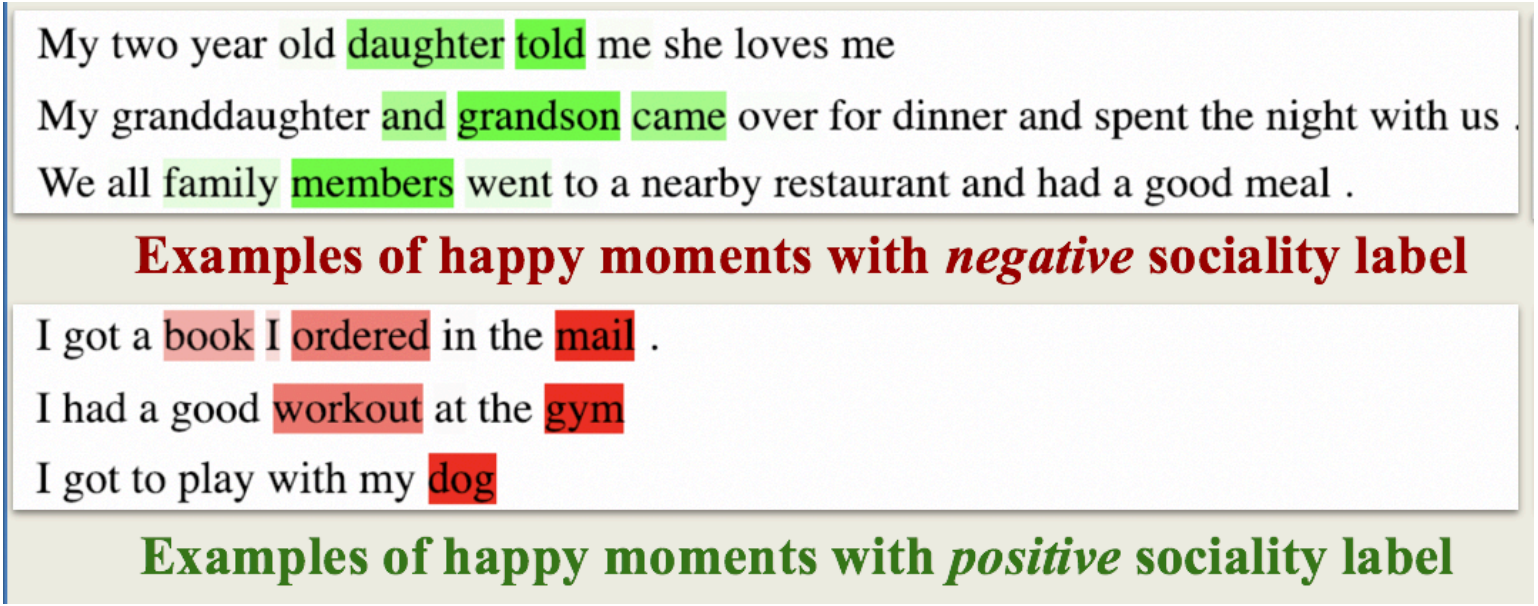


Figure: Attention heatmap for hand-picked **sociality** examples

What's in Sociality II?

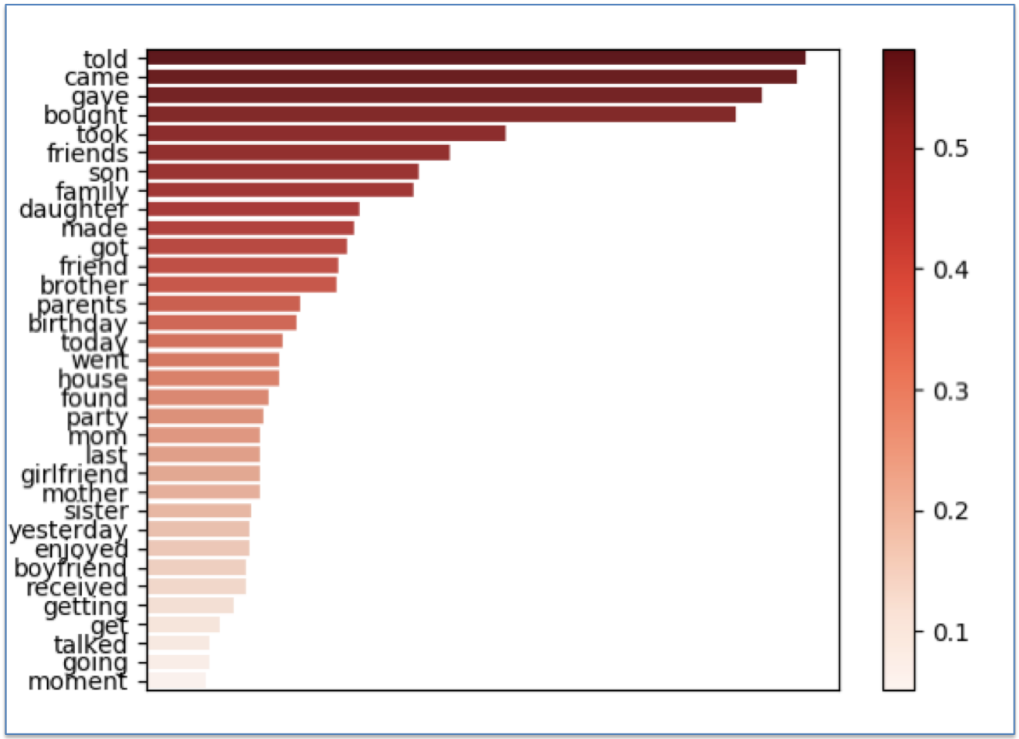


Figure: Attention-based feature ranking for **positive sociality**

For Your Reference: Other Works on Transfer Learning

Beyond Word Embedding Intializations

- **Word embeddings**, e.g., Word2vec (Mikilov et al., 2013), GloVe (Pennington et al., 2014) commonly used to initialize word vectors
- **Ramachandran et al. (2016)** initialize seq2seq models with pretrained LMs, and fine-tuning for a specific task
- **Kiros et al. (2015)** train an encoder on unlabeled data to output sentence vectors predictive of surrounding sentences
- **Hill et al. (2017)** show that fixed-length representations obtained from NMT encoders outperform those obtained from monolingual (e.g. LM) encoders on semantic similarity tasks
- This is similar to vision where an CNN pre-trained on e.g., ImageNet is de facto for initializing other (deeper) models **(transfer learning)**

CoVe (McCann et al., 2017)

CoVe

- **McCann et al., (2017)** Train an encoder for a large NLP task (like MT, supervised method), and transfer it to other tasks.
- **They transfer representations for each token in the input sequence**, rather than transferring from fixed length representations produced by NMT encoders.
- They **append the outputs of an MT-LSTM encoder** (trained on several MT tasks) **to the word vectors** typically used as input to classification tasks
- **CoVe(w) = MT-LSTM(GloVe(w))** where w is a sequence of words and GloVe(w) the corresponding sequence of word vectors produced by the GloVe model.
- **w = [GloVe(w); CoVe(w)]**

CoVe (McCann et al., 2017) *Contd.*

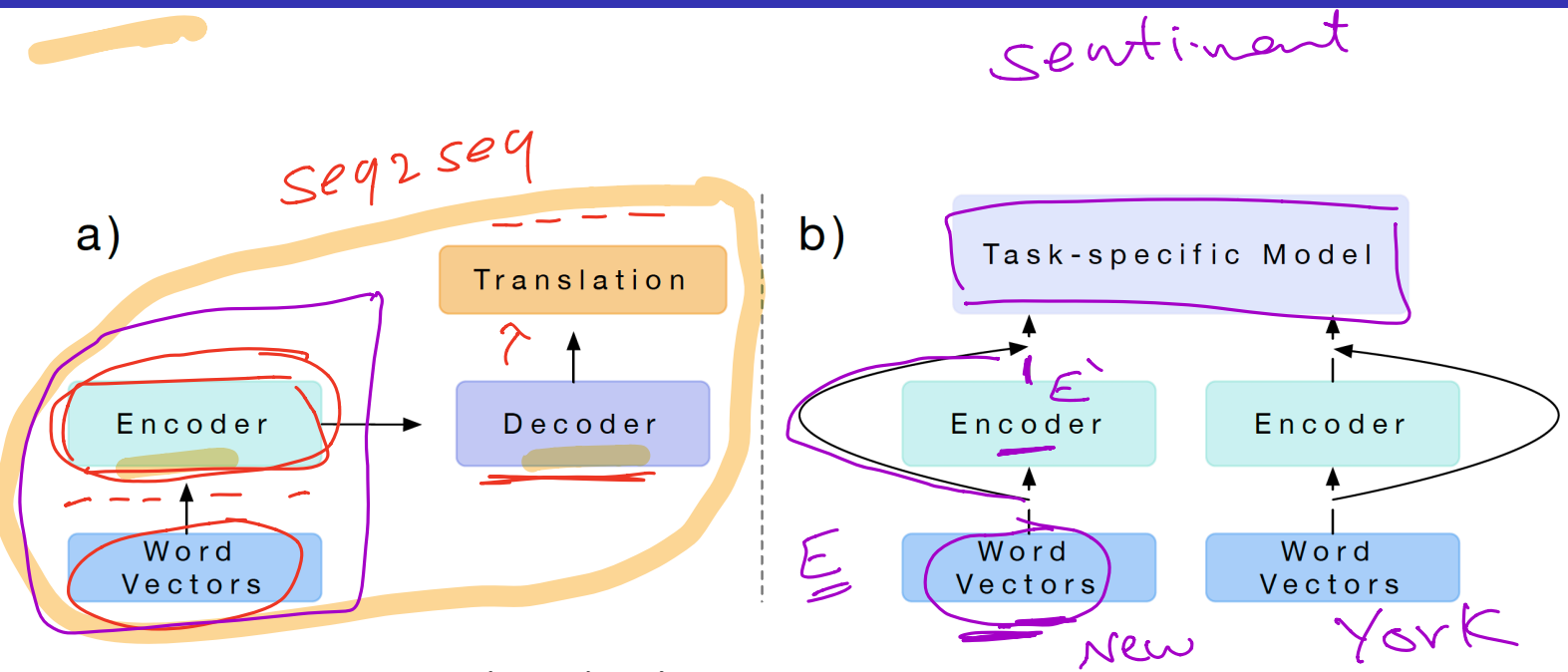


Figure: McCann et al. (2017): **a)** train a two-layer, bidirectional LSTM as the encoder of an attentional sequence-to-sequence model for MT and **b)** use it to provide context for other NLP models.