165B
Machine Learning
Sequence to Sequence Learning

Lei Li (leili@cs)
UCSB

Acknowledgement: Slides borrowed from Bhiksha Raj's 11485 and Mu Li & Alex Smola's 157 courses on Deep Learning, with modification

Recap

- Vocabulary building
  - Subword Tokenization
    - No OOV.
- Language Model

$$P(y) = \prod_t P(y_{t+1} \mid y_1 \ldots y_t)$$

- Word Embedding
- CNN-Language model
- Recurrent neural network
  - memory
  - Long-short term memory

Language Modeling

- Given a sentence y, estimate the probability

$$P(y) = \prod_t P(y_{t+1} \mid y_1 \ldots y_t)$$

$$P(y_{t+1} \mid y_1 \ldots y_t) = f_\theta(y_1, \ldots, y_t)$$

The cat sits on a __

y_1 y_2 y_3 y_4 y_5 y_6

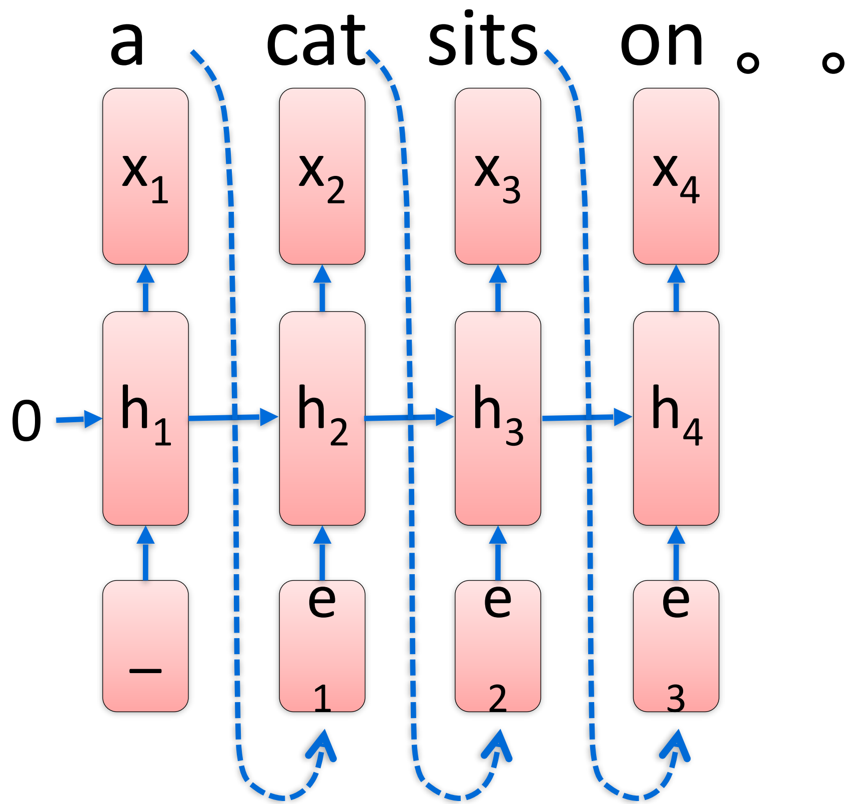
$p(y_6 \mid y_1, \ldots, y_5)$

mat	0.15
rug	0.13
chair	0.08
hat	0.05
dog	0.01

Recurrent Neural Network

$$p(x_t \mid x_1, \ldots, x_{t-1}) = \mathrm{softmax}(U \cdot h_t)$$

$$h_t = \sigma\left(W \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b \right)$$

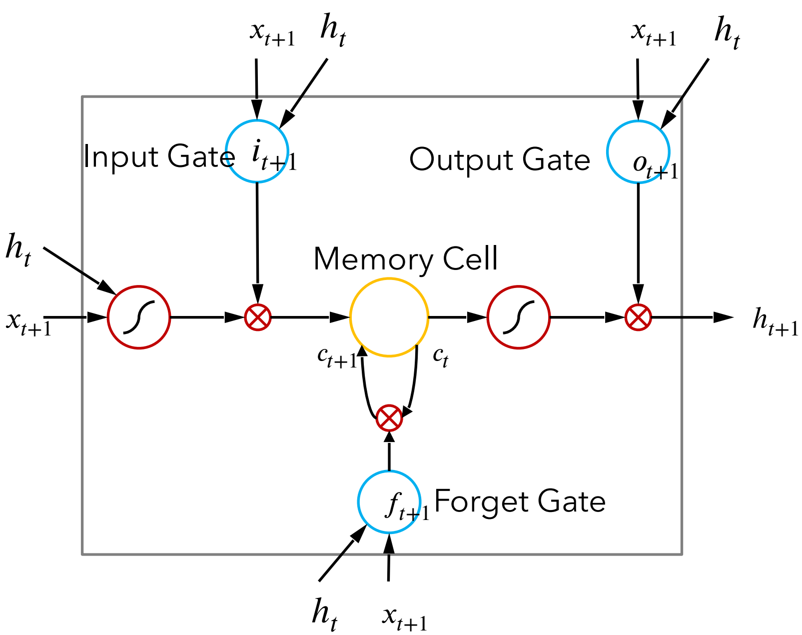


Elman, Finding Structure in Time. Cog. Sci. 1990.

Mikolov et al, Recurrent neural network based language model. Interspeech 2010.

Long-Short Term Memory (LSTM)

- Replace cell with more advanced one
- Adaptively memorize short and long term information



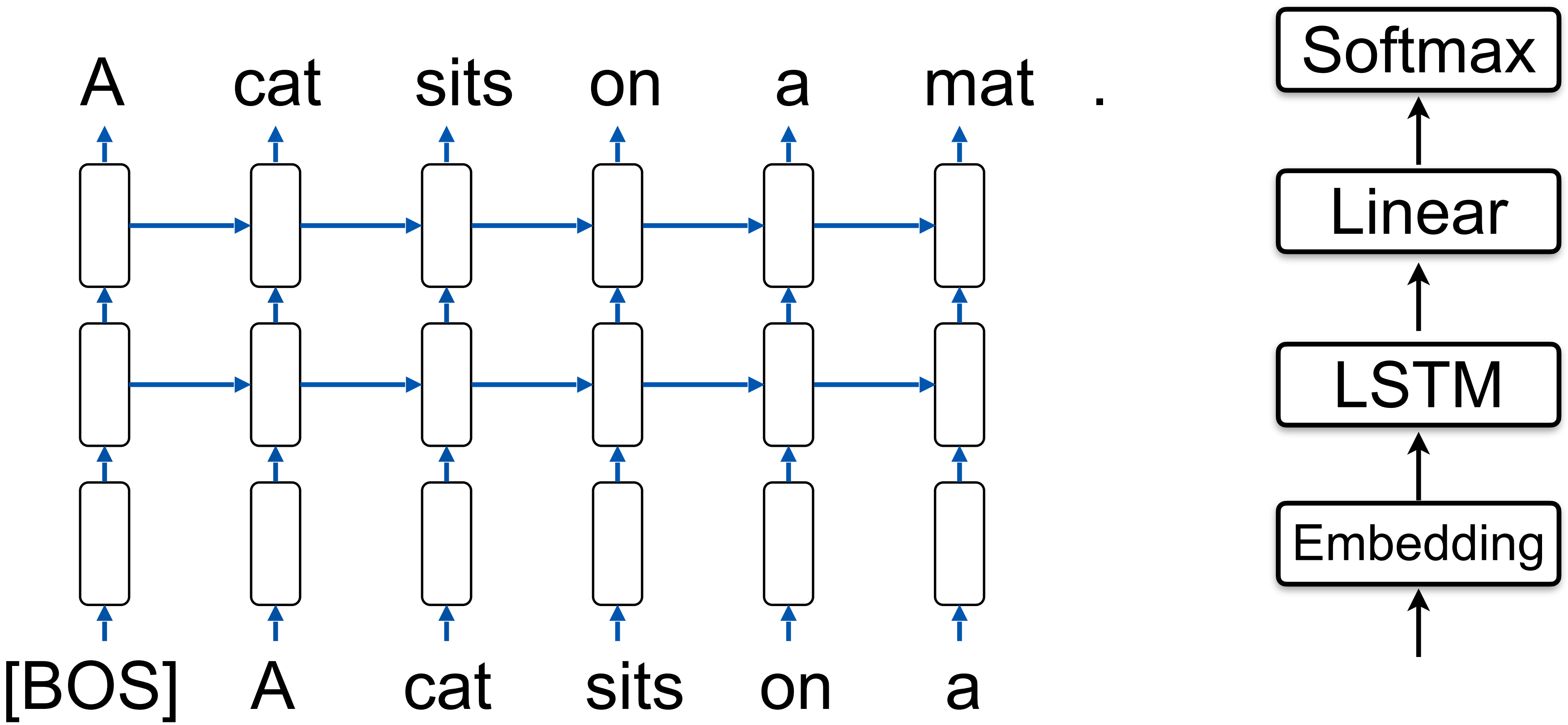
$$i_{t+1} = \sigma(M_{ix}x_{t+1} + M_{ih}h_t + b_i)$$
$$f_{t+1} = \sigma(M_{fx}x_{t+1} + M_{fh}h_t + b_f)$$
$$o_{t+1} = \sigma(M_{ox}x_{t+1} + M_{oh}h_t + b_o)$$

$$a_{t+1} = \tanh(M_{cx}x_{t+1} + M_{ch}h_t + b_a)$$

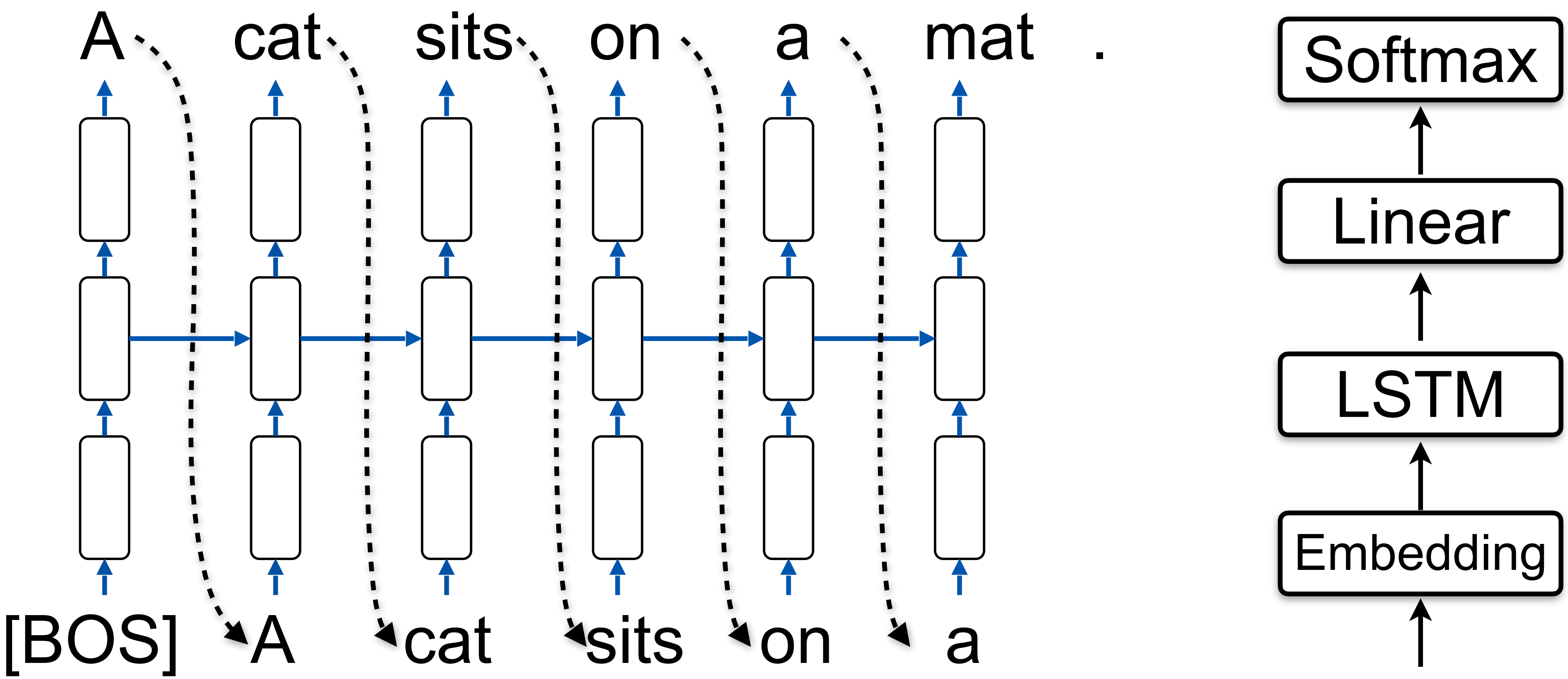
$$c_{t+1} = f_{t+1} \otimes c_t + i_{t+1} \otimes a_{t+1}$$
$$h_{t+1} = o_{t+1} \otimes \tanh(c_{t+1})$$

Hochreiter & Schmidhuber. Long Short-Term Memory, 1997
Gers et al. Learning to Forget: Continual Prediction with LSTM. 2000

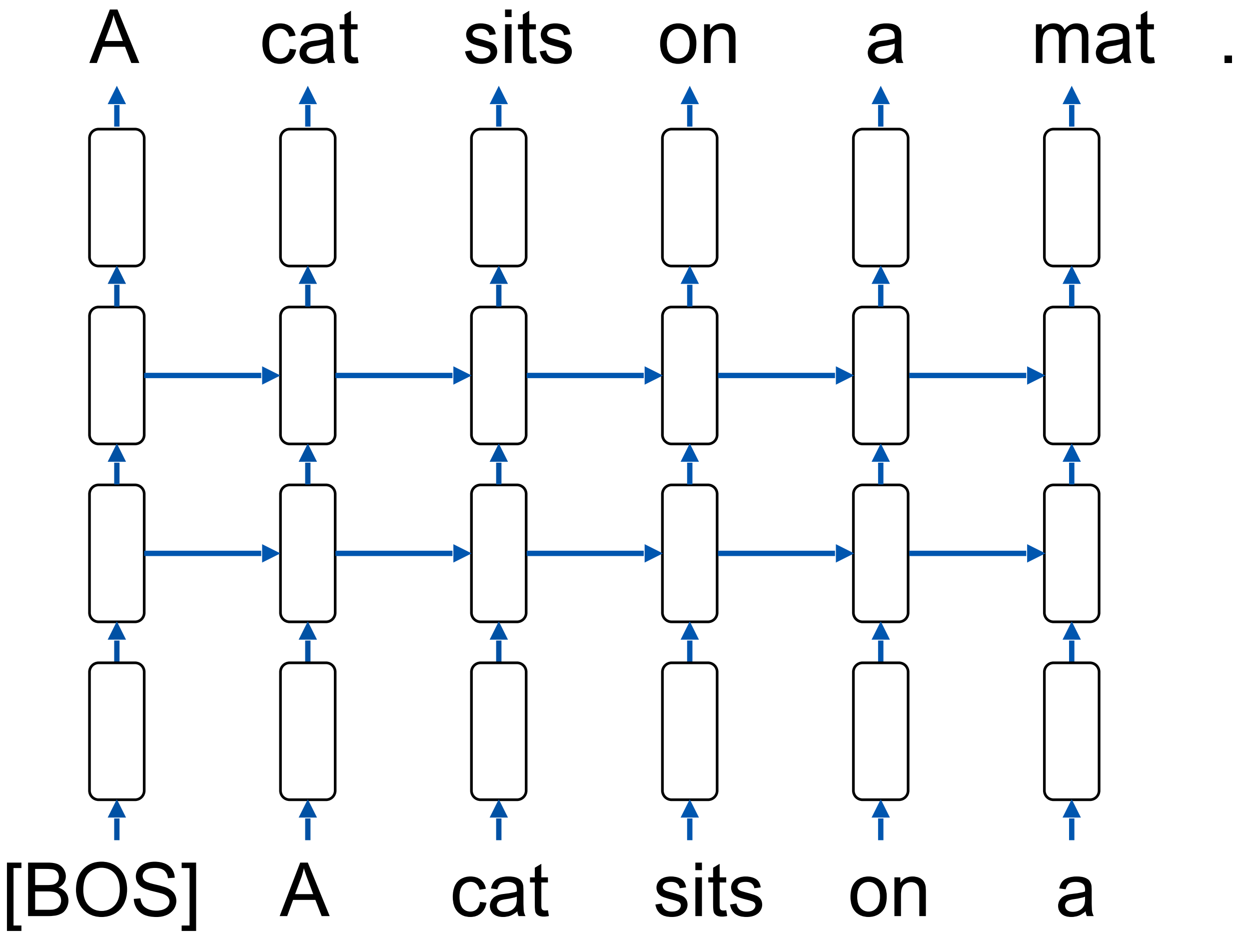
LSTM Language Modelling

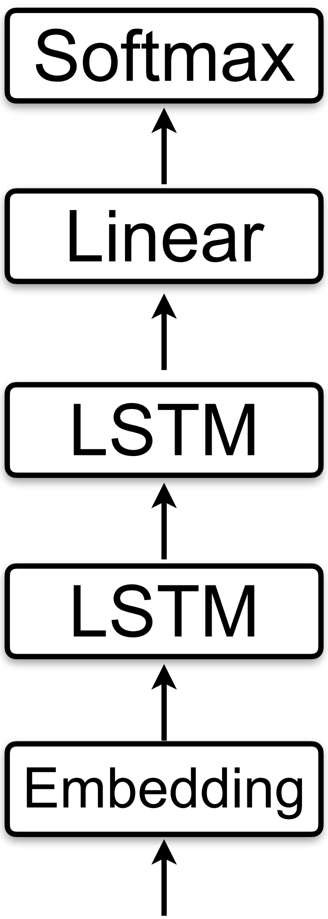


LSTM Generation



LSTM: More layers





Expressive Power of RNN-LM

Perplexity:

$$PPL = P(x_1, \ldots, x_N)^{-\frac{1}{N}} = \exp(-\frac{1}{N}\sum_{n=1}^{N} \log P(x_n \mid x_1 \ldots x_{n-1}))$$

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

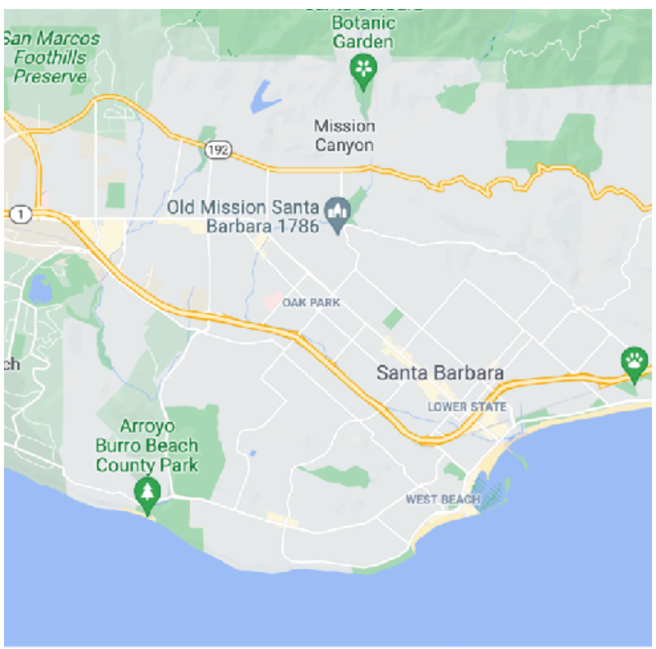
Jozefowicz et al. Exploring the limits of language modelling, 2016

Sequence Labelling

Understanding Query Intention

Noodle house near Santa Barbara
[Keyword] [Location]

How to go from Santa Barbara to Log Angeles ?
[Origin] [Destination]



Sequence Labelling

Named entity recognition

date   Location

In **April 1775** fighting broke out between **Massachusetts** militia units and **British** regulars at **Lexington** and **Concord** .

Geo-Political

Sequence Labelling

- Named entity recognition

  In **April 1775** fighting broke out between **Massachusetts** militia units and **British** regulars at **Lexington** and **Concord** .

- Semantic role labeling

The excess supply pushed gasoline prices down in that period .
subject verb object

- Question Answering: subject parsing

  Who created Harry Potter ?

Represent the Output Labels

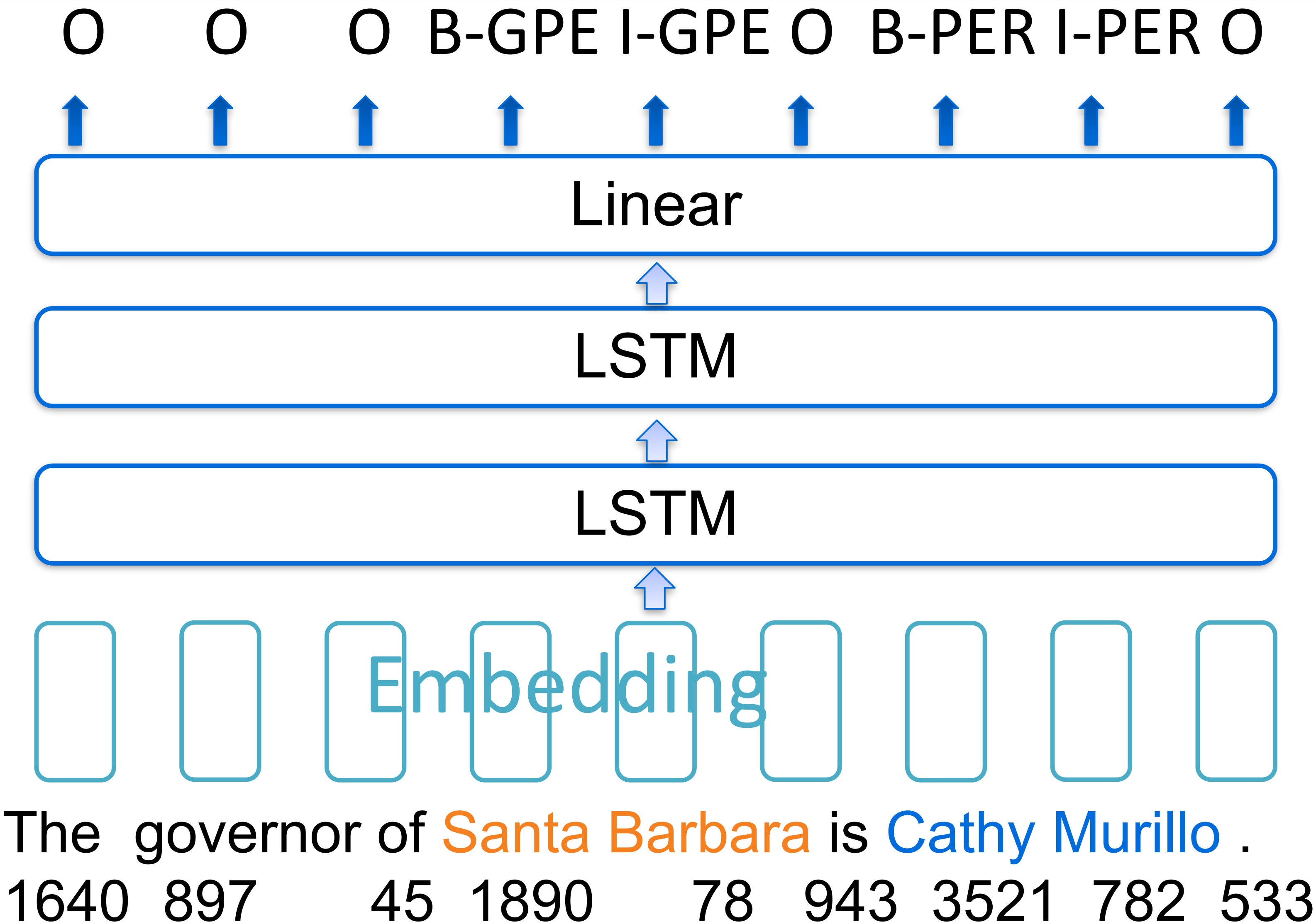
- BIO scheme

O O O B-GPE I-GPE O B-PER I-PER O

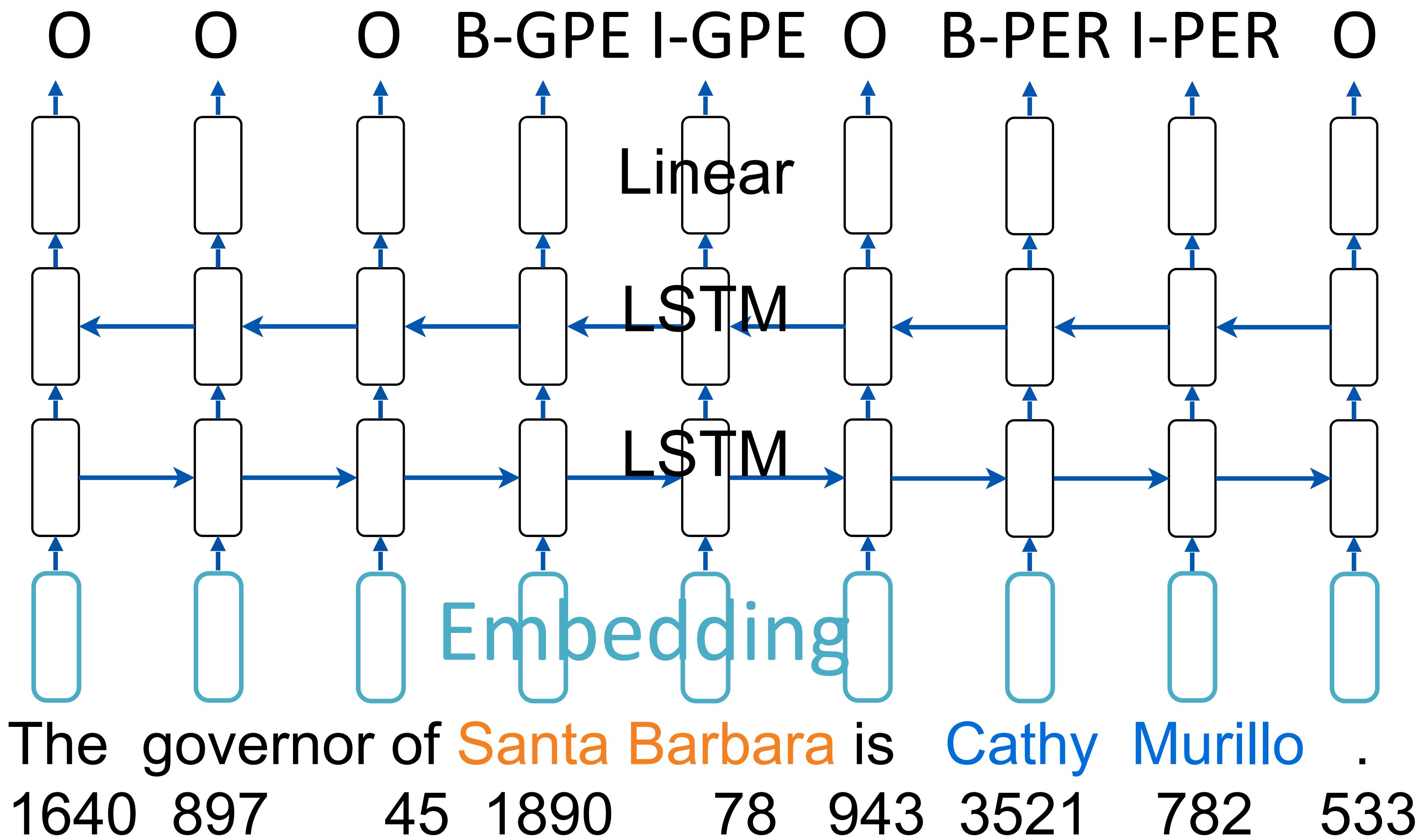
The governor of Santa Barbara is Cathy Murillo .

1640 897 45 1890 78 943 3521 782 533

RNN/LSTM for Sequence Labelling

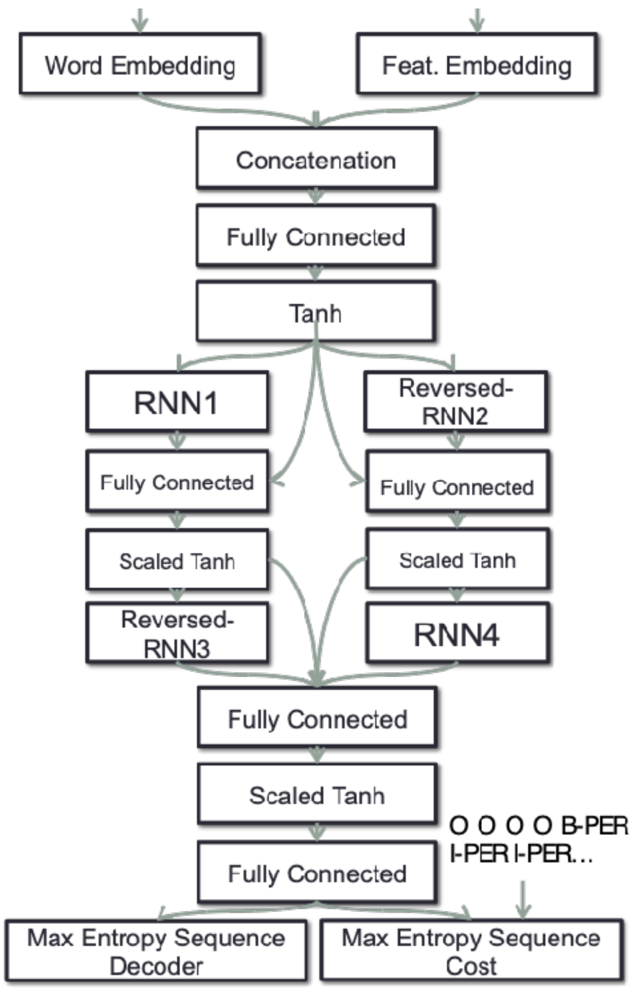


Bi-LSTM



O O O B-GPE I-GPE O B-PER I-PER O
Linear
LSTM
LSTM
Embedding
The governor of Santa Barbara is Cathy Murillo .
1640 897 45 1890 78 943 3521 782 533

Twisted NN for NER



Chinese NER

OntoNotes Data 4-class:

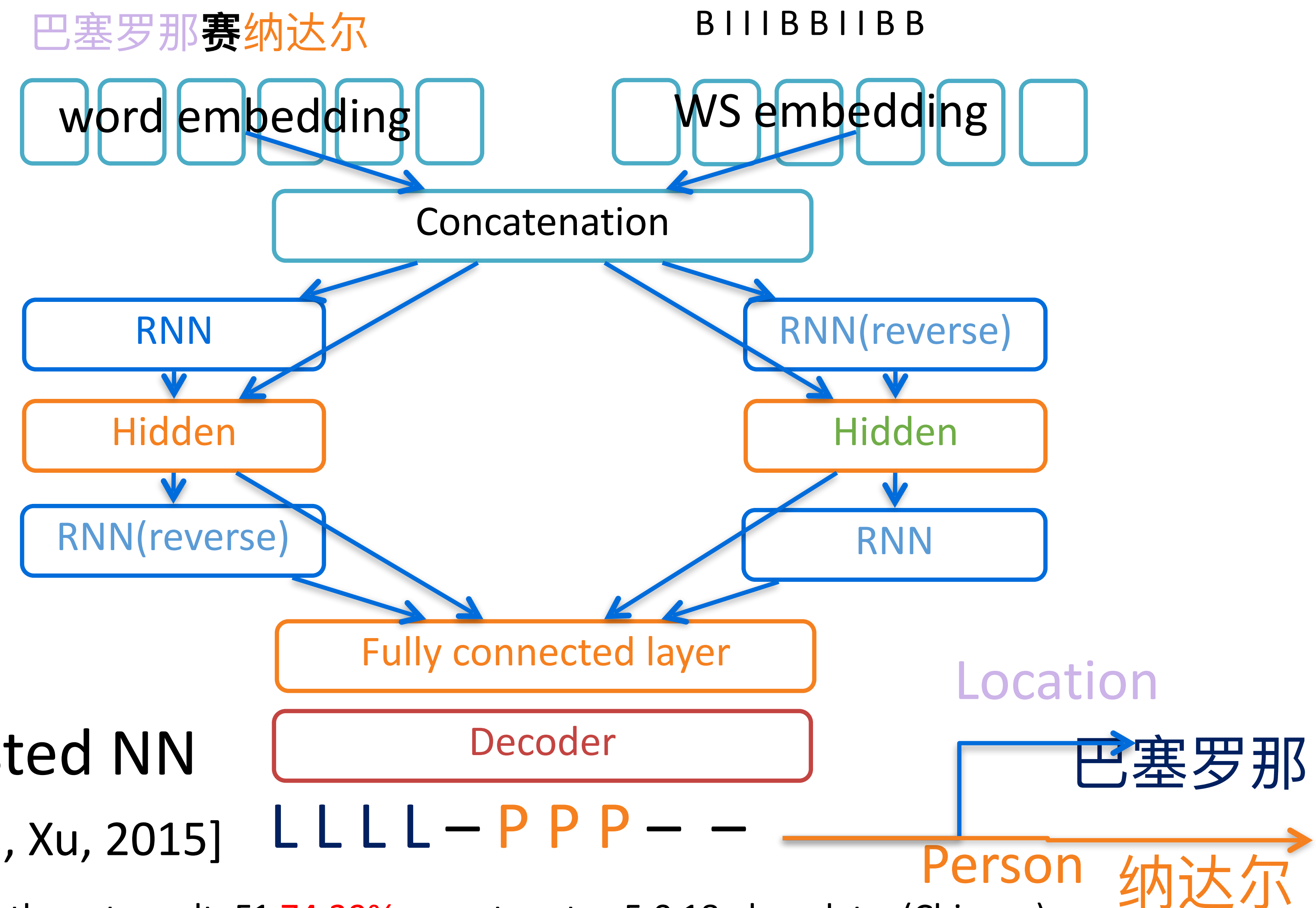
Model	P	R	F1
Bi-NER-WA* Wang et al.	84.42	76.34	80.18
RNN-2b with WS ours	84.75	77.85	81.15

* Wang et al used bilingual data

OntoNotes Data 18-class:

Model	P	R	F1
Sameer Pradhan et al.	78.20	66.45	71.85
RNN-2b with WS ours	78.69	70.54	74.39

Twisted NN [Zefu Lu, Lei Li, Wei Xu, 2015]



Twisted NN

[Lu, Li, Xu, 2015]

State-of-the-art result: F1 74.39% on ontonotes-5.0 18-class data. (Chinese)

Sequence Labelling using LSTM (Pytorch)

```
class LSTMTagger(nn.Module):

    def __init__(self, embedding_dim, hidden_dim, vocab_size, tagset_size):
        super(LSTMTagger, self).__init__()
        self.hidden_dim = hidden_dim

        self.word_embeddings = nn.Embedding(vocab_size, embedding_dim)

        # The LSTM takes word embeddings as inputs, and outputs hidden states
        # with dimensionality hidden_dim.
        self.lstm = nn.LSTM(embedding_dim, hidden_dim)

        # The linear layer that maps from hidden state space to tag space
        self.hidden2tag = nn.Linear(hidden_dim, tagset_size)

    def forward(self, sentence):
        embeds = self.word_embeddings(sentence)
        lstm_out, _ = self.lstm(embeds.view(len(sentence), 1, -1))
        tag_space = self.hidden2tag(lstm_out.view(len(sentence), -1))
        tag_scores = F.log_softmax(tag_space, dim=1)
        return tag_scores
```

Training in Pytorch

```
model = LSTMTagger(EMBEDDING_DIM, HIDDEN_DIM, len(word_to_ix), len(tag_to_ix))
loss_function = nn.NLLLoss()
optimizer = optim.SGD(model.parameters(), lr=0.1)

# See what the scores are before training
# Note that element i,j of the output is the score for tag j for word i.
# Here we don't need to train, so the code is wrapped in torch.no_grad()
with torch.no_grad():
    inputs = prepare_sequence(training_data[0][0], word_to_ix)
    tag_scores = model(inputs)
    print(tag_scores)

for epoch in range(300):  # again, normally you would NOT do 300 epochs, it is toy data
    for sentence, tags in training_data:
        # Step 1. Remember that Pytorch accumulates gradients.
        # We need to clear them out before each instance
        model.zero_grad()

        # Step 2. Get our inputs ready for the network, that is, turn them into
        # Tensors of word indices.
        sentence_in = prepare_sequence(sentence, word_to_ix)
        targets = prepare_sequence(tags, tag_to_ix)

        # Step 3. Run our forward pass.
        tag_scores = model(sentence_in)

        # Step 4. Compute the loss, gradients, and update the parameters by
        #  calling optimizer.step()
        loss = loss_function(tag_scores, targets)
        loss.backward()
        optimizer.step()
```

Testing in Pytorch

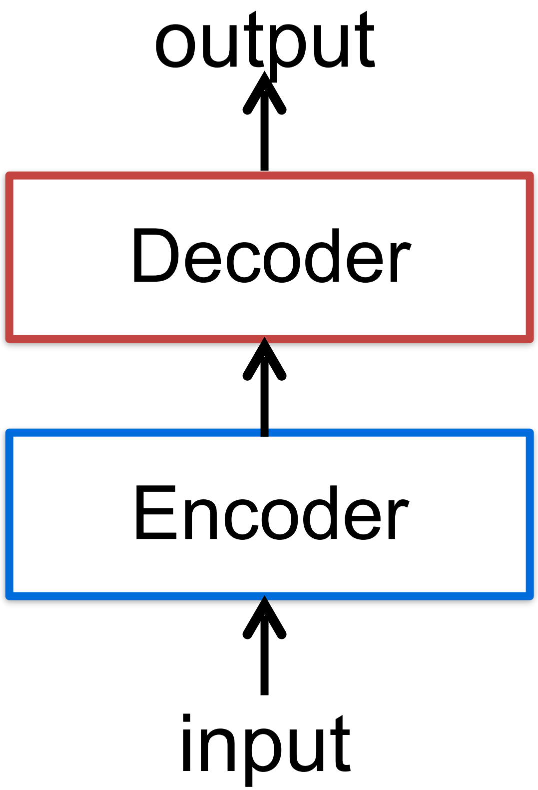
```
# See what the scores are after training
with torch.no_grad():
    inputs = prepare_sequence(training_data[0][0], word_to_ix)
    tag_scores = model(inputs)
```

Better Loss Function (advanced)

- Loss using Conditional Random Fields

$$
\begin{aligned}
-\log(P(\mathbf{y} \mid \mathbf{X})) &= -\log\left(\frac{\exp\left(\sum_{k=1}^{\ell} U(\mathbf{x}_k, y_k) + \sum_{k=1}^{\ell-1} T(y_k, y_{k+1})\right)}{Z(\mathbf{X})}\right) \\
&= \log\left(Z(\mathbf{X})\right) - \log\left(\exp\left(\sum_{k=1}^{\ell} U(\mathbf{x}_k, y_k) + \sum_{k=1}^{\ell-1} T(y_k, y_{k+1})\right)\right) \\
&= \log\left(Z(\mathbf{X})\right) - \left(\sum_{k=1}^{\ell} U(\mathbf{x}_k, y_k) + \sum_{k=1}^{\ell-1} T(y_k, y_{k+1})\right) \\
&= Z_{\log}(\mathbf{X}) - \left(\sum_{k=1}^{\ell} U(\mathbf{x}_k, y_k) + \sum_{k=1}^{\ell-1} T(y_k, y_{k+1})\right)
\end{aligned}
$$

Encoder-decoder framework



A generic formulation

ImageCaption

Text-to-Image Generation

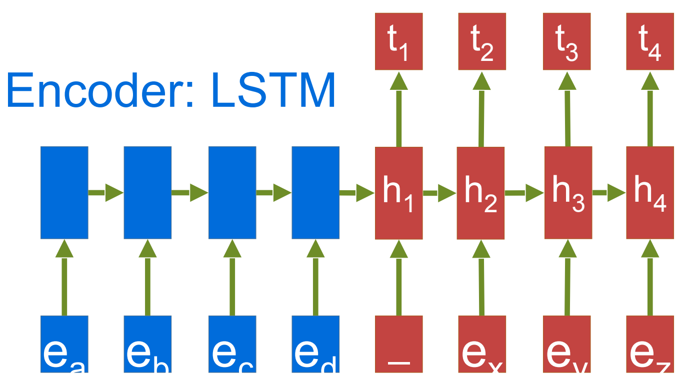
ASR (speech-to-text)

MT (text-to-text)

Sequence To Sequence (Seq2seq)

- Machine translation as directly learning a function mapping from source sequence to target sequence

target:
The weather is nice



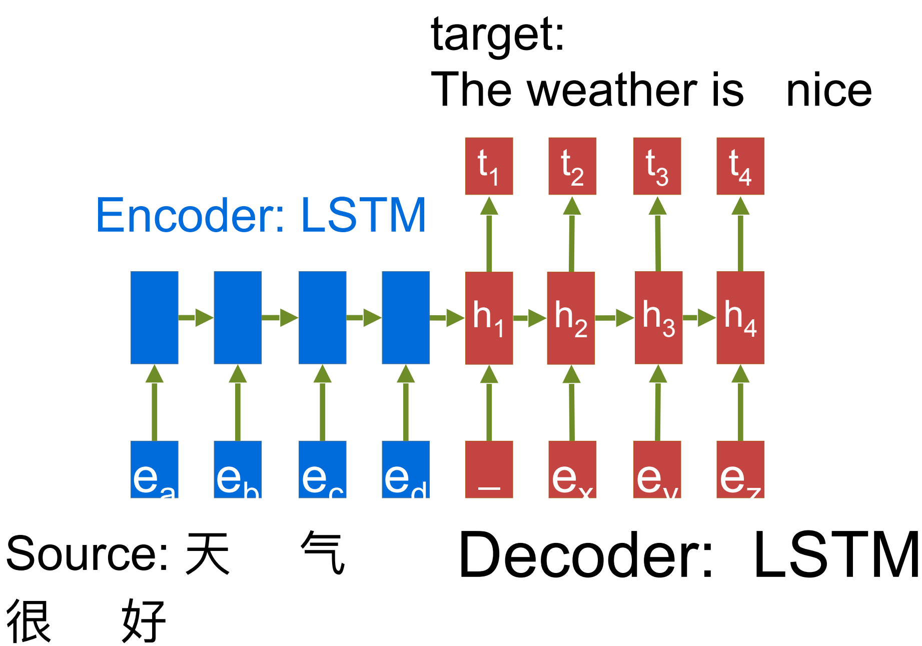
Source: 天 气 很 好

Decoder: LSTM

Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014

Sequence To Sequence (Seq2seq)

- Machine translation as directly learning a function mapping from source sequence to target sequence



$$P(Y|X) = \prod P(y_t | y_{<t}, x)$$

Training loss: Cross-Entropy

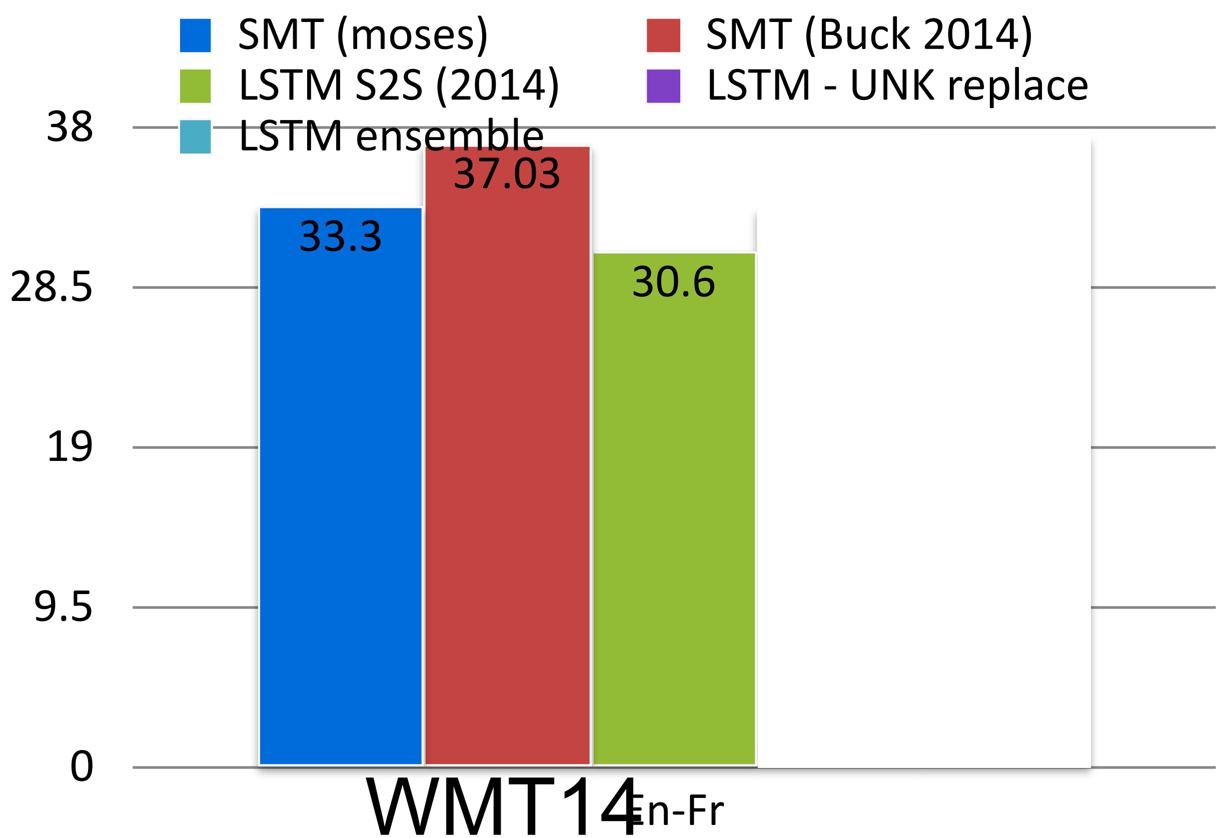
$$l = -\sum_n \sum_t \log f_\theta(x_n, y_{n,1}, \ldots, y_{n,t-1})$$

Teacher-forcing during training.

(pretend to know groundtruth for prefix)

Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014

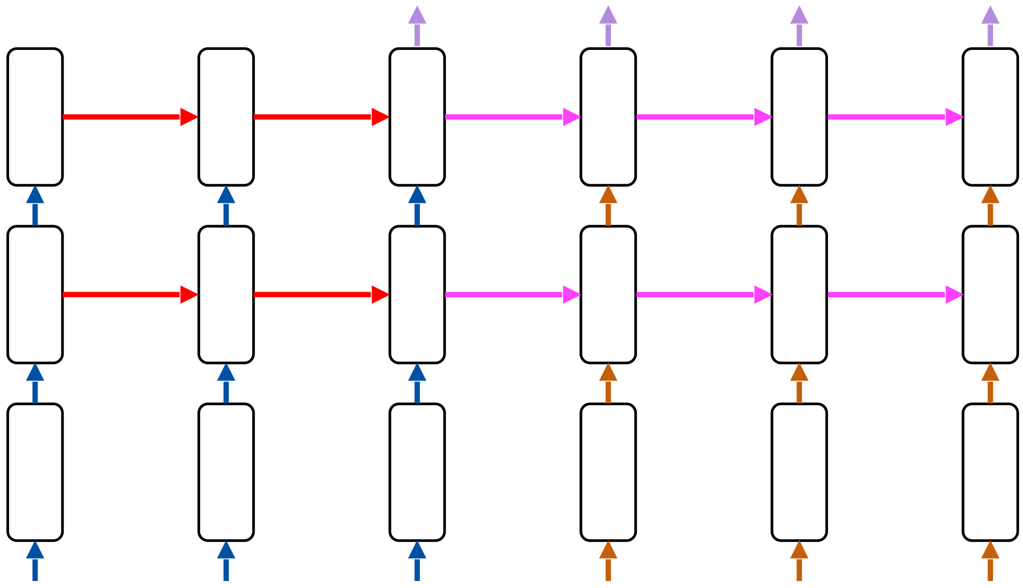
Performance (2014)



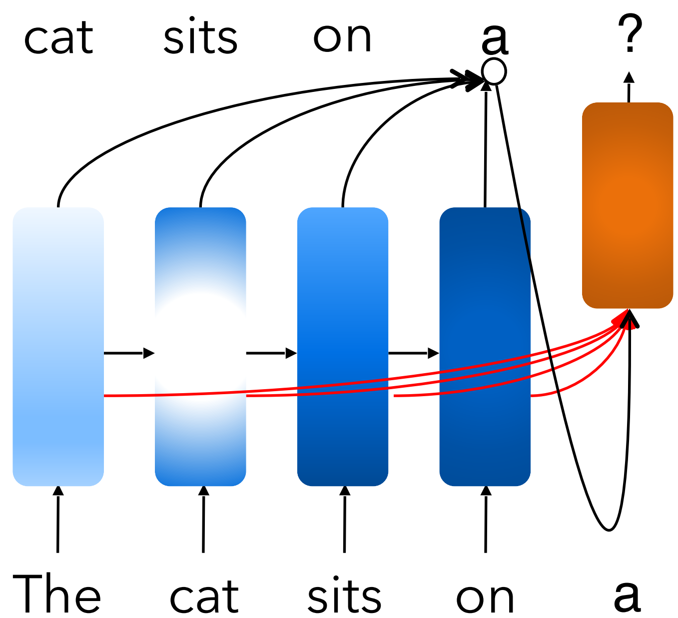
Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014
Durrani et al. Edinburgh's Phrase-based Machine Translation Systems for WMT-14. 201

Stacked LSTM for seq-2-seq

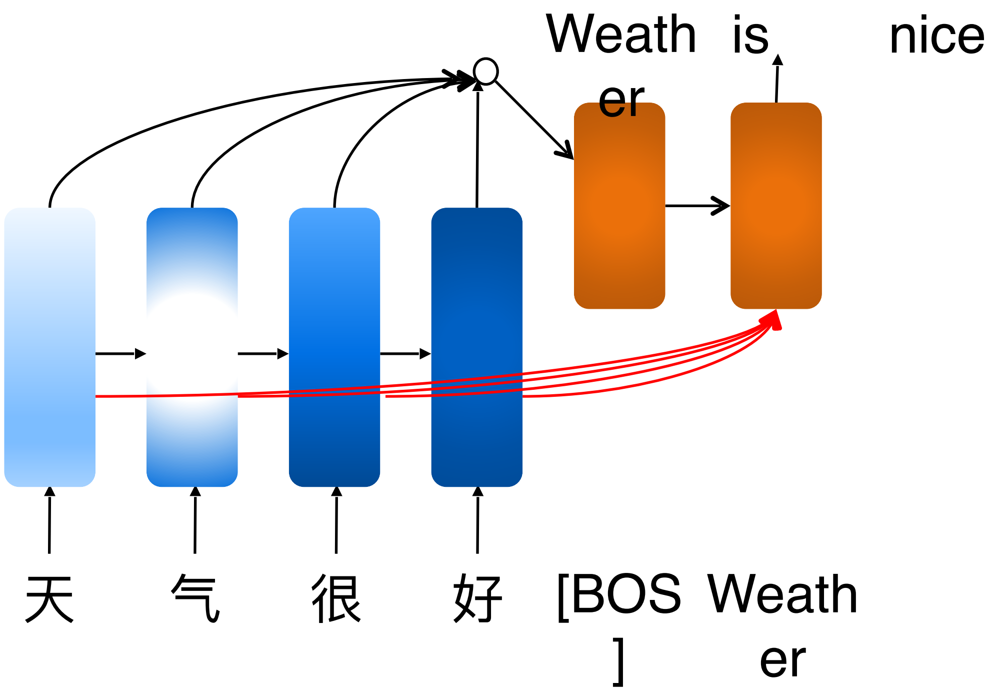
- More layers of LSTM



Attention

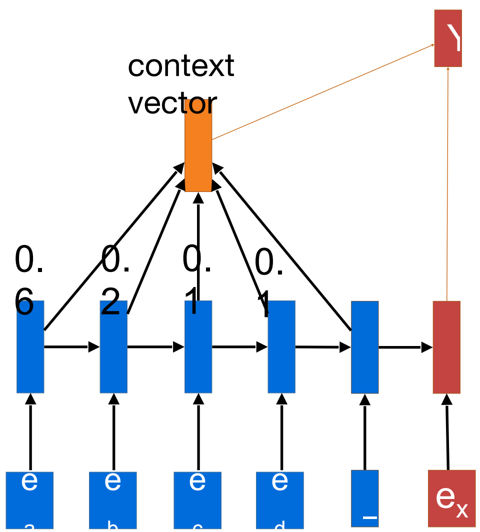


LSTM Seq2seq with Attention



Bahdanau et al., Neural Machine Translation by Jointly Learning to Align and Translate 2015

Generation by Attention



A context vector c will be predicted before, which represents the related source context for current predicted word.

$$\alpha_{nj} = \mathrm{Softmax}(D(s_n, h_{1\ldots n-1})) = \frac{\exp(D(s_n, h_j))}{\sum_k \exp(D(s_n, h_k))}$$

$$c_n = \sum_j \alpha_{nj} h_j$$

The probability of word y_i is computed as:

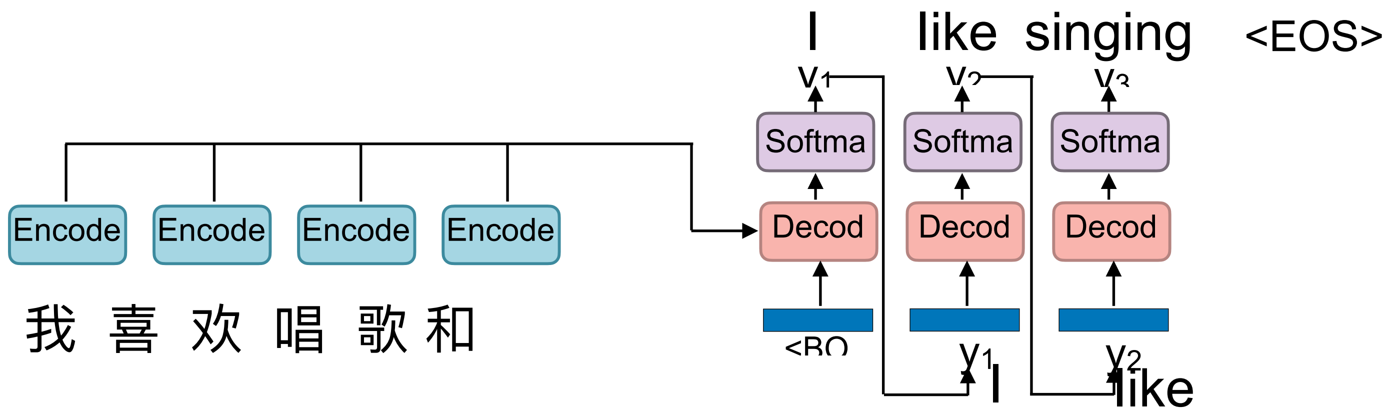
$$p(y_i) \propto \exp(W h_i) \quad \Rightarrow \quad p(y_i) \propto \exp(W h_i + V c_i)$$

Mnih et al. Recurrent Models of Visual Attention. 2014.

Decoding

Autoregressive Generation

greedy decoding: output the token with max next token prob



But, this is not necessary the best

Inference

- Now already trained a model θ
- Decoding/Generation: Given an input sentence x, to generate the target sentence y that maximize the probability $P(y \mid x; \theta)$
- $\underset{y}{\operatorname{argmax}}\, P(y \mid x) = f_\theta(x, y)$
- Two types of error
  - the most probable translation is bad → fix the model
  - search does not find the most probably translation → fix the search
- Most probable translation is not necessary the highest BLEU one!

Decoding

- $\underset{y}{\operatorname{argmax}} P(y \mid x) = f_\theta(x, y)$
- naive solution: exhaustive search
  - too expensive
- Beam search
  - (approximate) dynamic programming

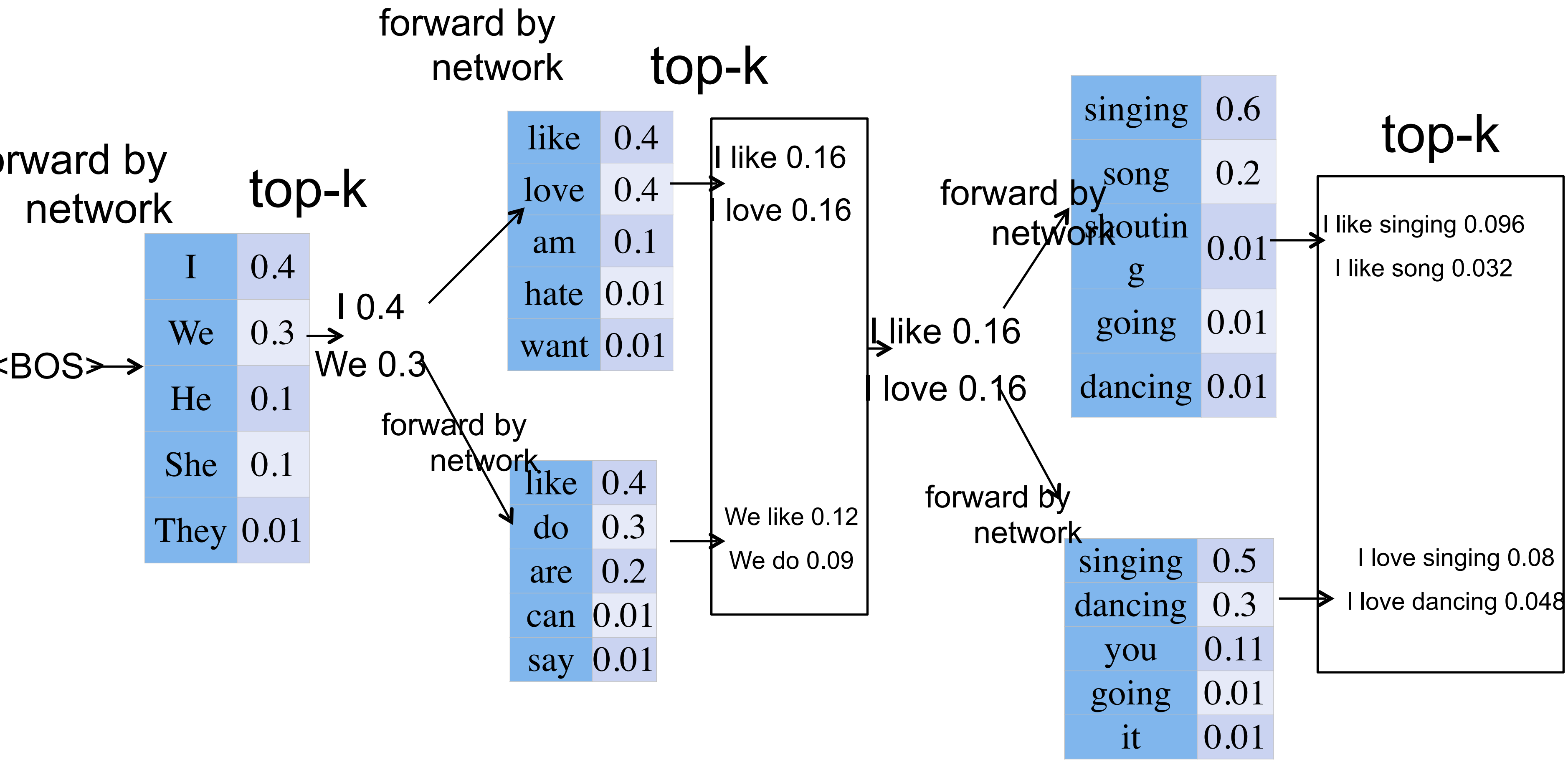
Beam Search

- start with empty S
- at each step, keep k best partial sequences
- expand them with one more forward generation
- collect new partial results and keep top-k

Beam Search (pseudocode)

```
best_scores = []
add {[0], 0.0} to best_scores # 0 is for beginning of sentence token
for i in 1 to max_length:
  new_seqs = PriorityQueue()
  for (candidate, s) in best_scores:
    if candidate[-1] is EOS:
        prob = all -inf
        prob[EOS] = 0
      else:
      prob = using model to take candidate and compute next token
probabilities (logp)
    pick top k scores from prob, and their index
    for each score, index in the top-k of prob:
      new_candidate = candidate.append(index)
      new_score = s + score
      if not new_seqs.full():
```

Beam Search



forward by network
top-k
<BOS>
I 0.4
We 0.3
He 0.1
She 0.1
They 0.01
I 0.4
We 0.3
forward by network
top-k
like 0.4
love 0.4
am 0.1
hate 0.01
want 0.01
forward by network
like 0.4
do 0.3
are 0.2
can 0.01
say 0.01
I like 0.16
I love 0.16
We like 0.12
We do 0.09
I like 0.16
I love 0.16
forward by network
singing 0.6
song 0.2
shouting 0.01
going 0.01
dancing 0.01
forward by network
singing 0.5
dancing 0.3
you 0.11
going 0.01
it 0.01
top-k
I like singing 0.096
I like song 0.032
I love singing 0.08
I love dancing 0.048

Seq2seq for Machine Translation

Many possible translation, which is better?

SpaceX周三晚间进行了一次发射任务，将四名毫无航天经验的业余人士送入太空轨道。

SpaceX launched a mission Wednesday night to put four amateurs with no space experience into orbit.

SpaceX conducted a launch mission on Wednesday night, sending four amateurs with no aerospace experience into space orbit.

SpaceX conducted a launch mission Wednesday night that sent four amateurs with no spaceflight experience into orbit.

SpaceX carried out a launch mission on Wednesday night to put four amateurs without Aerospace experience into orbit.

BLEU

- Measuring the precision of n-grams
  – Precision of n-gram: percentage of tokens in output sentences

$$\_\, p_n = \frac{num\,.\,of\,.\,correct\,.\,token\,.\,ngram}{total\,.\,output\,.\,ngram}$$

- Penalize for brevity
  – if output is too short
  – $bp = min(1, e^{1-r/c})$
- BLEU=$bp \cdot (\prod p_i)^{\frac{1}{4}}$
- Notice BLEU is computed over the whole corpus, not on one sentence

Example

Ref: A SpaceX rocket was launched into a space orbit Wednesday evening.

System A: SpaceX launched a mission Wednesday evening into a space orbit.

System B: A rocket sent SpaceX into orbit Wednesday.

Example

Ref: A SpaceX rocket was launched into a space orbit Wednesday evening.

System A: SpaceX launched a mission Wednesday evening into a space orbit.

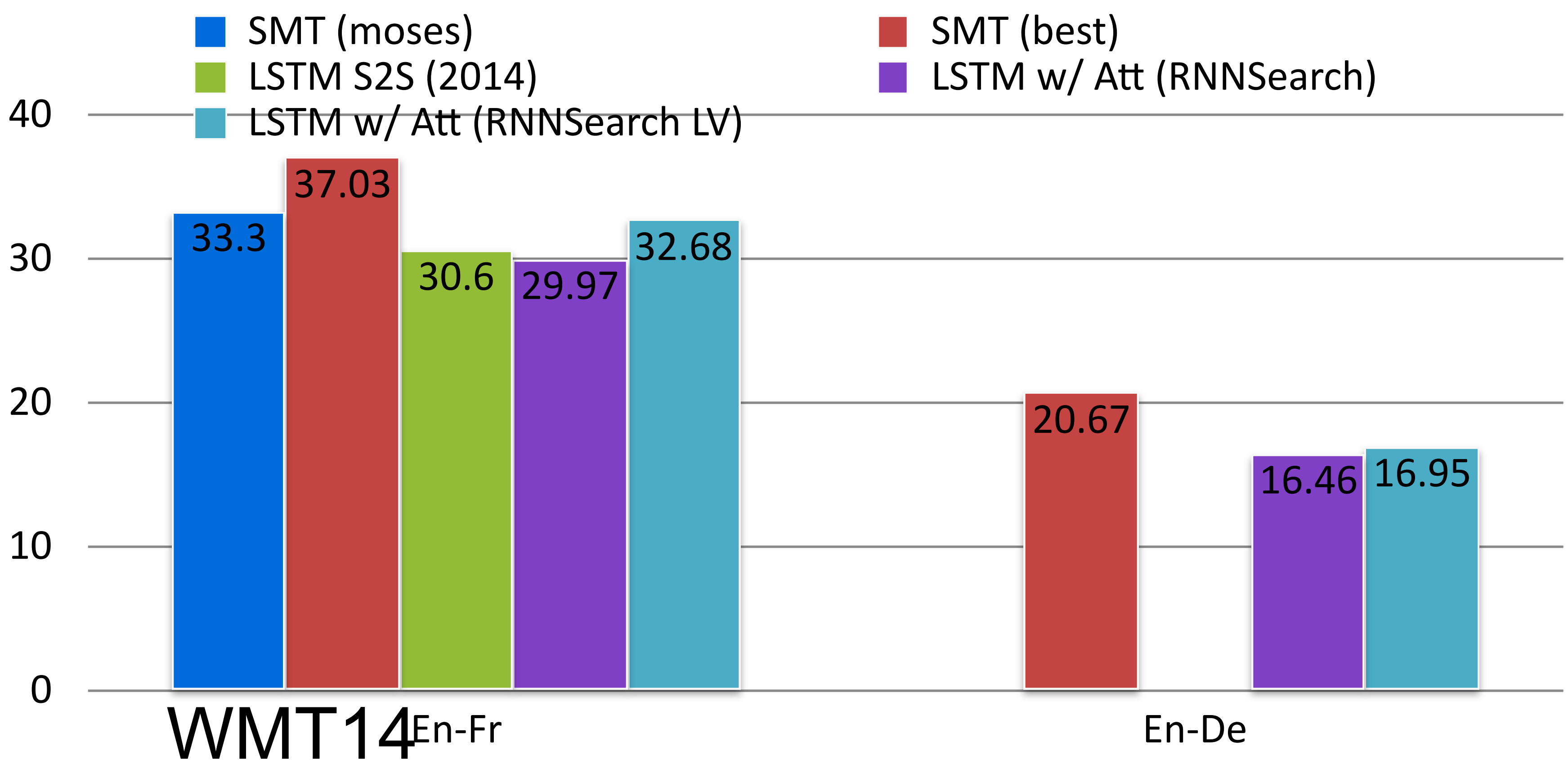
	Precision
Unigram	9/11
Bigram	4/10
Trigram	2/9
Four-gram	1/8

$$bp=e^{1-12/11}=0.91$$

$$BLEU=0.91*(9/11 * 4/10 * 2/9 * 1/8)^{1/4}$$

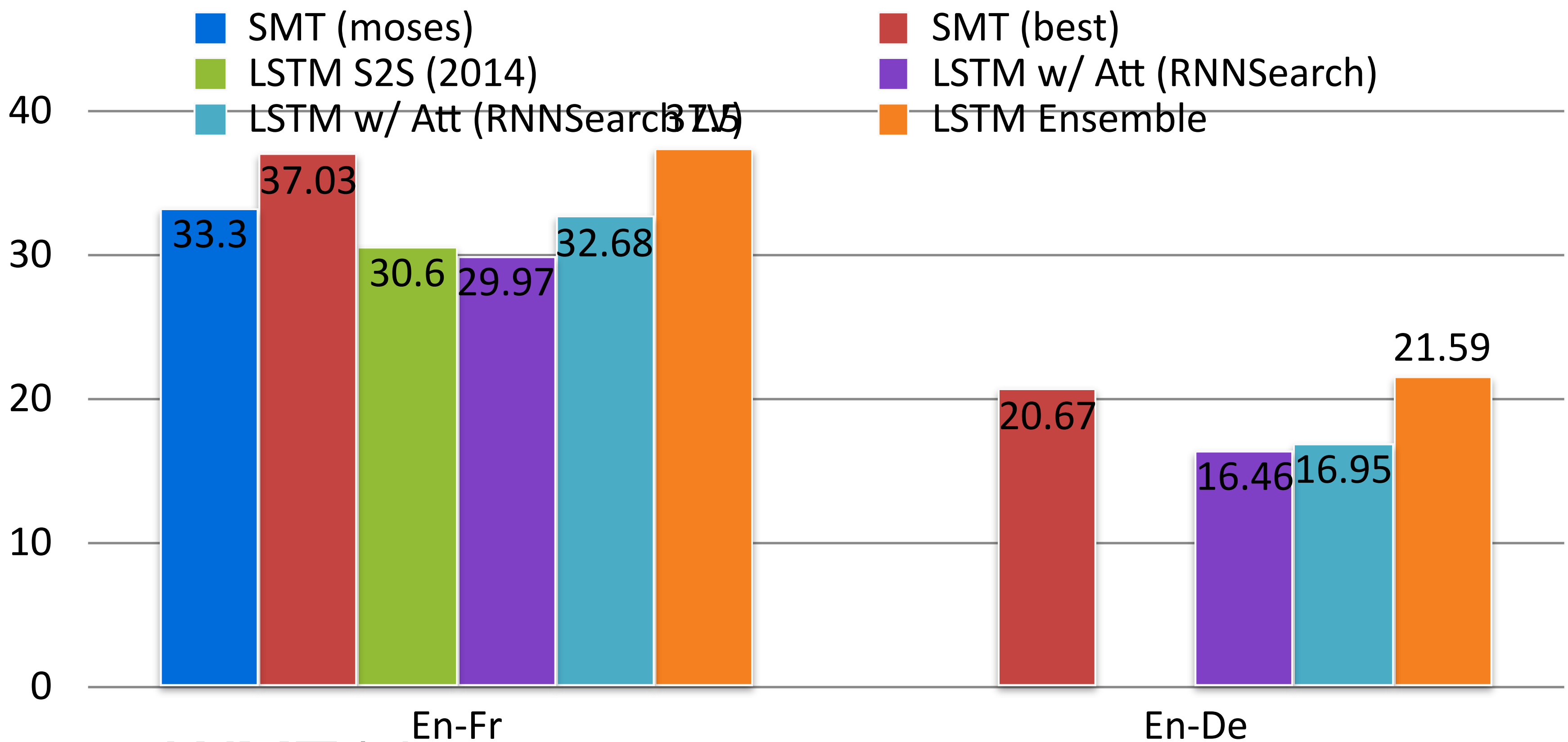
$$=28.1\%$$

LSTM Seq2Seq w/ Attention



Jean et al. On Using Very Large Target Vocabulary for Neural Machine Translation. 2015

Performance with Model Ensemble



WMT14

Luong et al. Effective Approaches to Attention-based Neural Machine Translation. 2015

Summary

- LSTM-RNN Language Modelling
- Sequence Labelling
  - named entity recognition/semantic role labelling/POS tagging
  - Bi-LSTM
- Seq2seq
  - End-to-end model for Machine Translation

Next Up

- Transformer