WestlakeNLP

Natural Language Processing

Yue Zhang
Westlake University

西湖大學
WESTLAKE UNIVERSITY

WestlakeNLP

Chapter 16

Working With Two Texts

Contents

WestlakeNLP

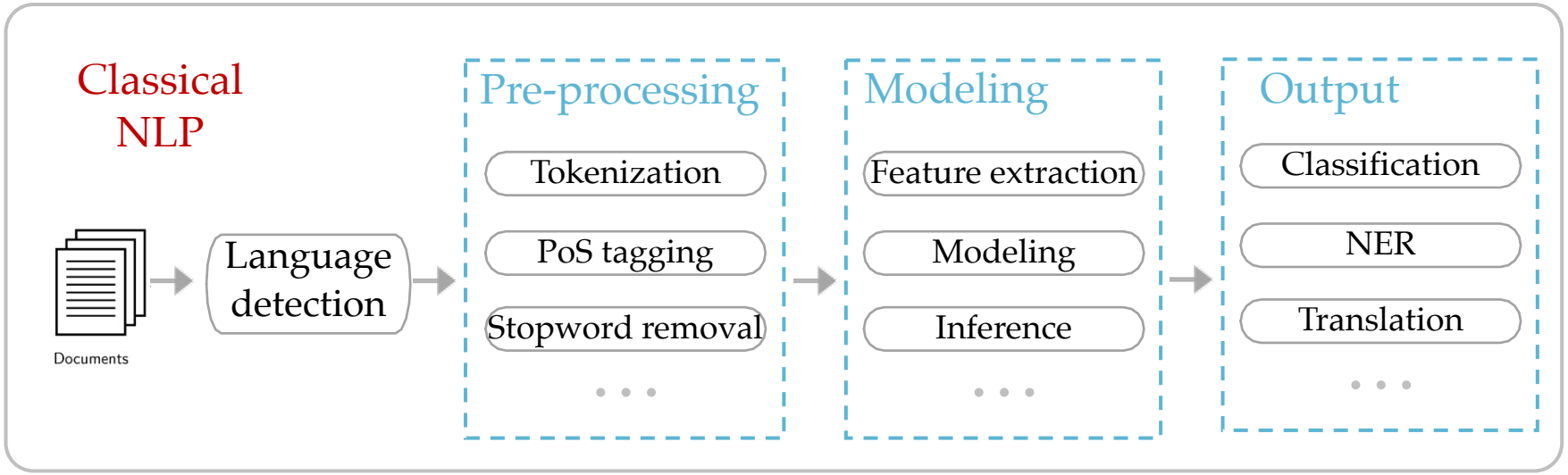
Contents

WestlakeNLP

End-to-End Architectures

WestlakeNLP

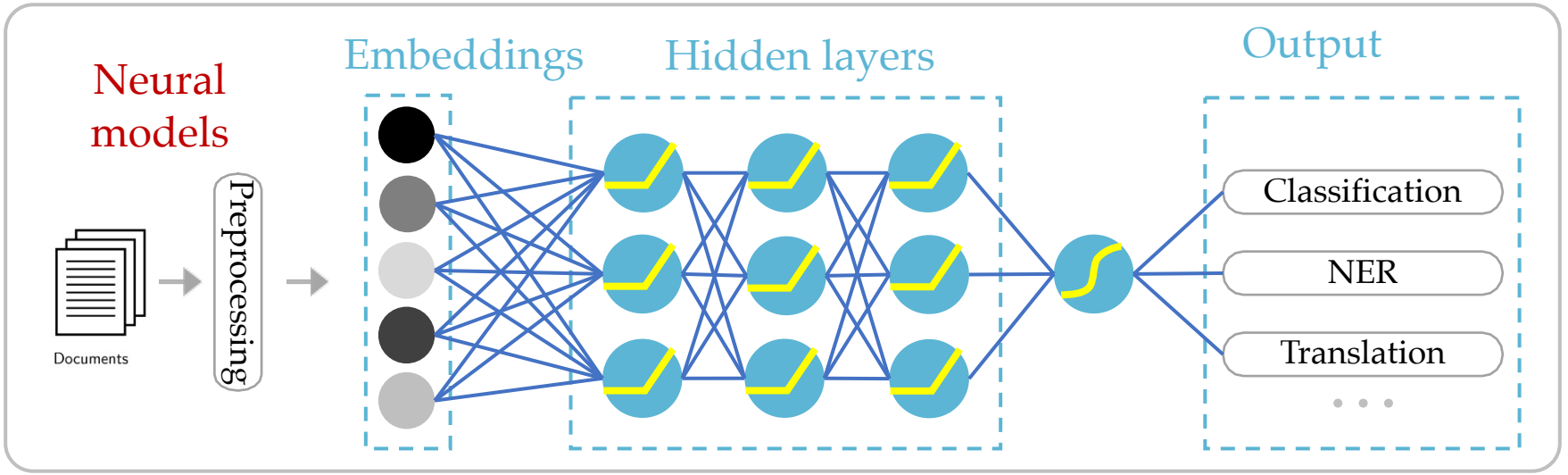
- Using neural models, hand-crafted features can be replaced by Dense Representation.
- A pipelined approach with sub-systems can be replaced by a simple end-to-end system.



End-to-End Architectures

WestlakeNLP

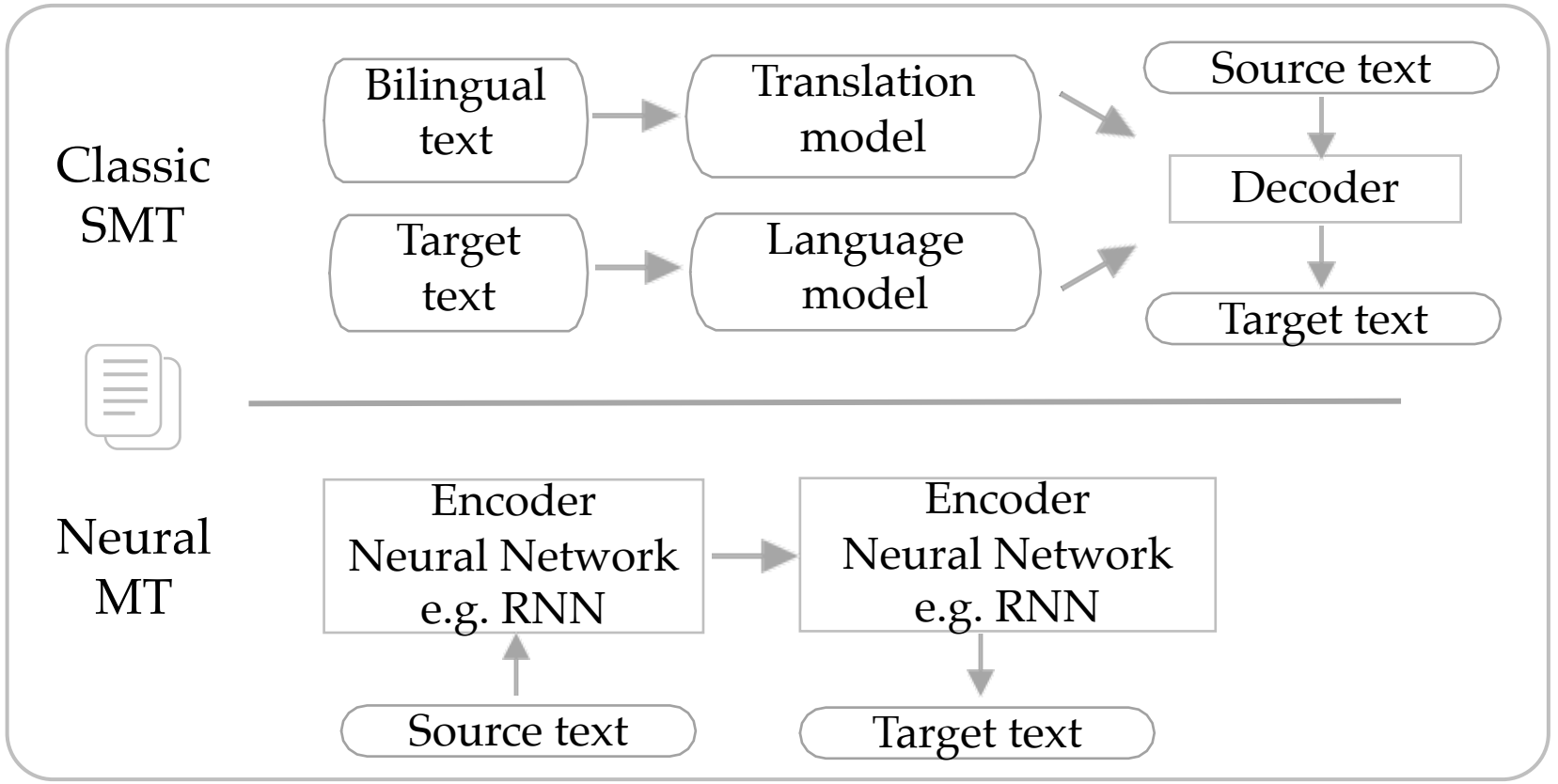
- Using neural models, hand-crafted features can be replaced by Dense Representation.
- A pipelined approach with sub-systems can be replaced by a simple end-to-end system.



End-to-End Architectures

WestlakeNLP

- Take an example of Machine Translation:



End-to-End Architectures

WestlakeNLP

- Sequence-to-sequence is useful for *more than just MT*

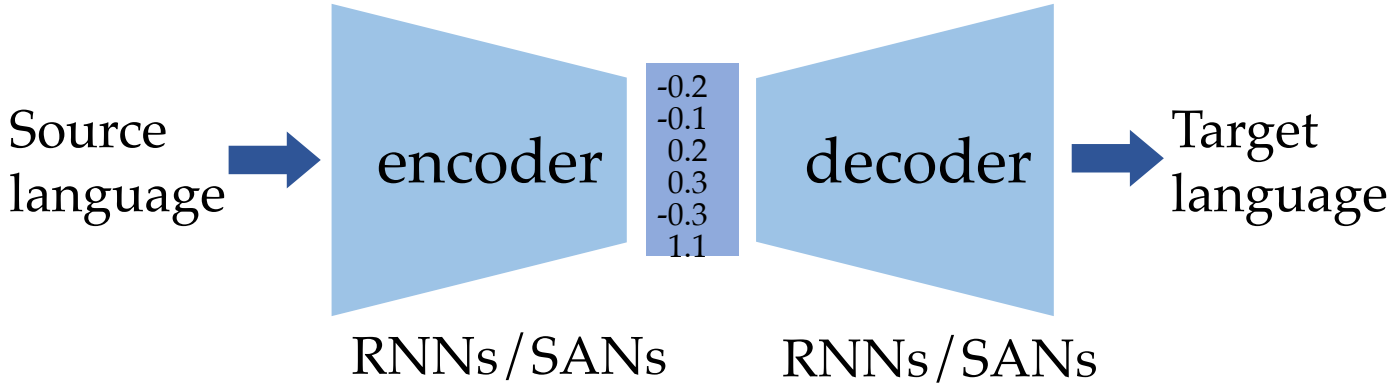
- Many NLP tasks can be phrased as sequence-to-sequence:
  - Machine translation(source text → target text)
  - Summarization (long text → short text)
  - Dialogue (previous utterances → next utterance)

Encoder-Decoder Structure

WestlakeNLP

- Encoder for learning a neural input representation.
- Decoder for constructing the output sequence.

For Machine Translation:

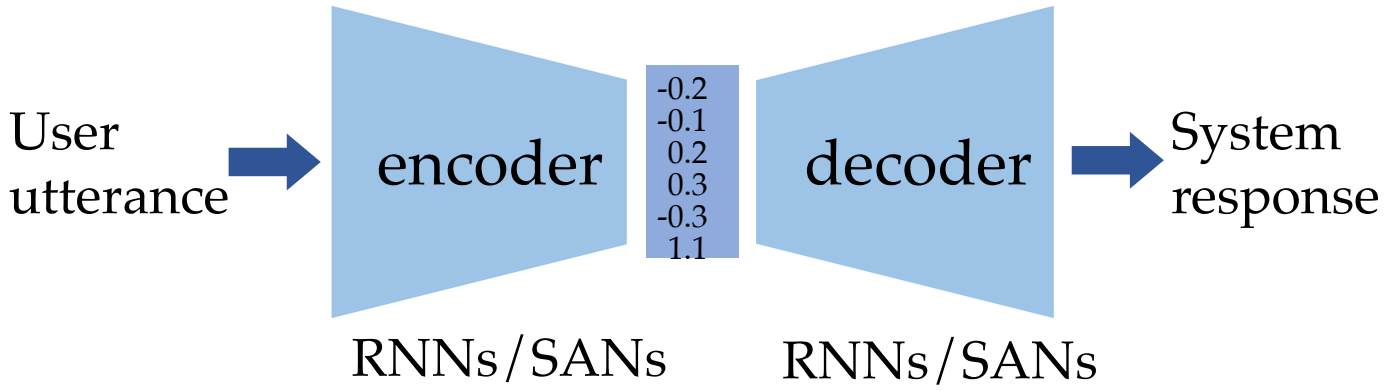


Encoder-Decoder Structure

WestlakeNLP

- Encoder for learning a neural input representation.
- Decoder for constructing the output sequence.

For Dialogue System:



Contents

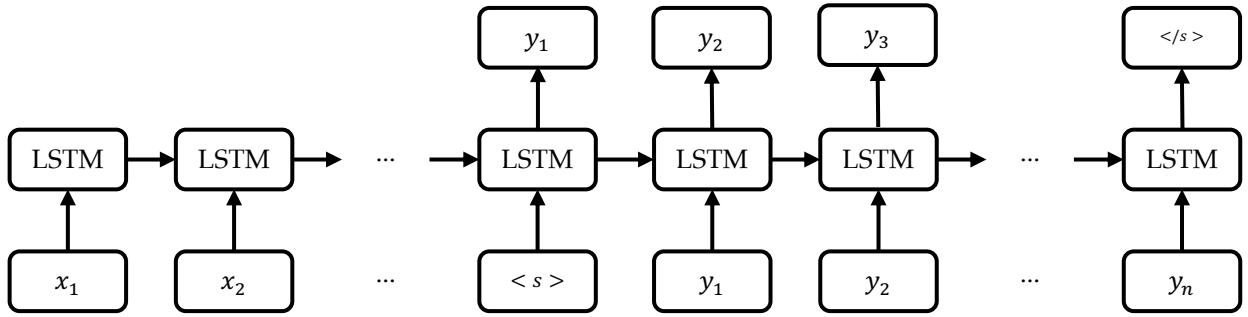
WestlakeNLP

Model 1: Seq2seq using LSTM

WestlakeNLP

- Encoding and Decoding with LSTM

Given an input sentence: $X_{1:n} = x_1, x_2, \ldots, x_n$

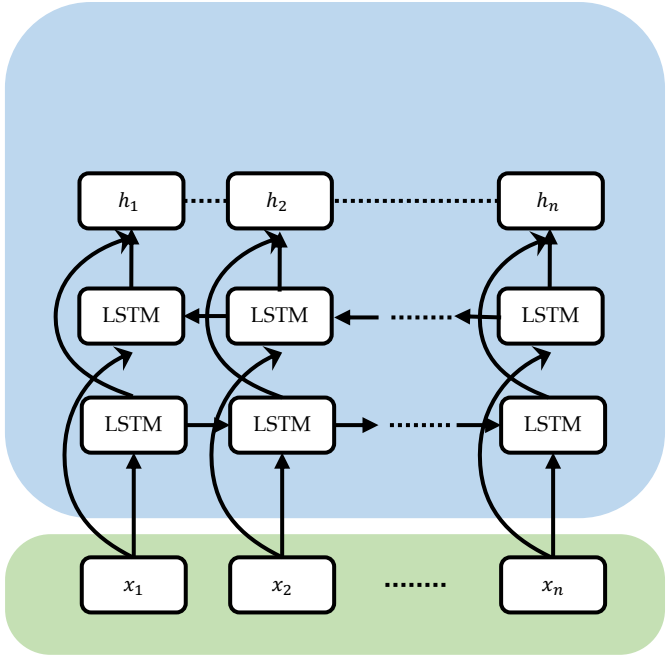


Generating an output sentence: $Y_{1:m} = y_1, y_2, \ldots, y_m$

Model 1: Seq2seq using LSTM

WestlakeNLP

- Encoder: Bi-directional LSTM



$$\overrightarrow{\mathbf{h}}_i^{enc} = LSTM\left(\overrightarrow{\mathbf{h}}_{i-1}^{enc}, emb(x_i)\right)$$

$$\overleftarrow{\mathbf{h}}_i^{enc} = LSTM\left(\overleftarrow{\mathbf{h}}_{i+1}^{enc}, emb(x_i)\right)$$

$$\mathbf{h}^{enc} = [\overrightarrow{\mathbf{h}}_n^{enc}; \overleftarrow{\mathbf{h}}_1^{enc}]$$

To decoder

Model 1: Seq2seq using LSTM

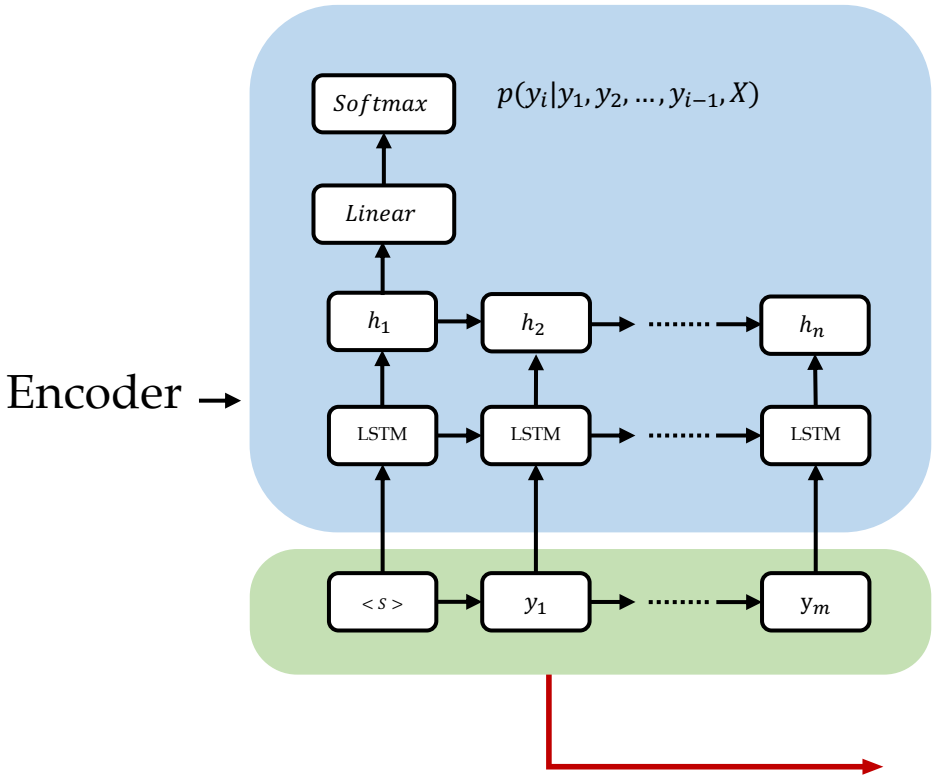
WestlakeNLP

- Encoder: some tips
  - A useful practice to add pseudo start $< s >$ and end tokens $</s >$ to the beginning and end of the input.
  - It is useful to use $\sum_{i=1}^{n} emb(x_i)$ as the start hidden states $\vec{\mathbf{h}}_0^{enc}$ and $\overleftarrow{\mathbf{h}}_{n+1}^{enc}$ in both directions instead of zeros.

Model 1: Seq2seq using LSTM

WestlakeNLP

- Decoder: output of encoder is used as an initial state h_0^{dec}



$$\mathbf{h}_1^{dec} = LSTM\left(\mathbf{h}_0^{dec}, emb'(\langle s \rangle)\right)$$

Predicting the first word y_i:

$$\mathbf{o}_1 = \mathbf{W}\mathbf{h}_1^{dec}$$

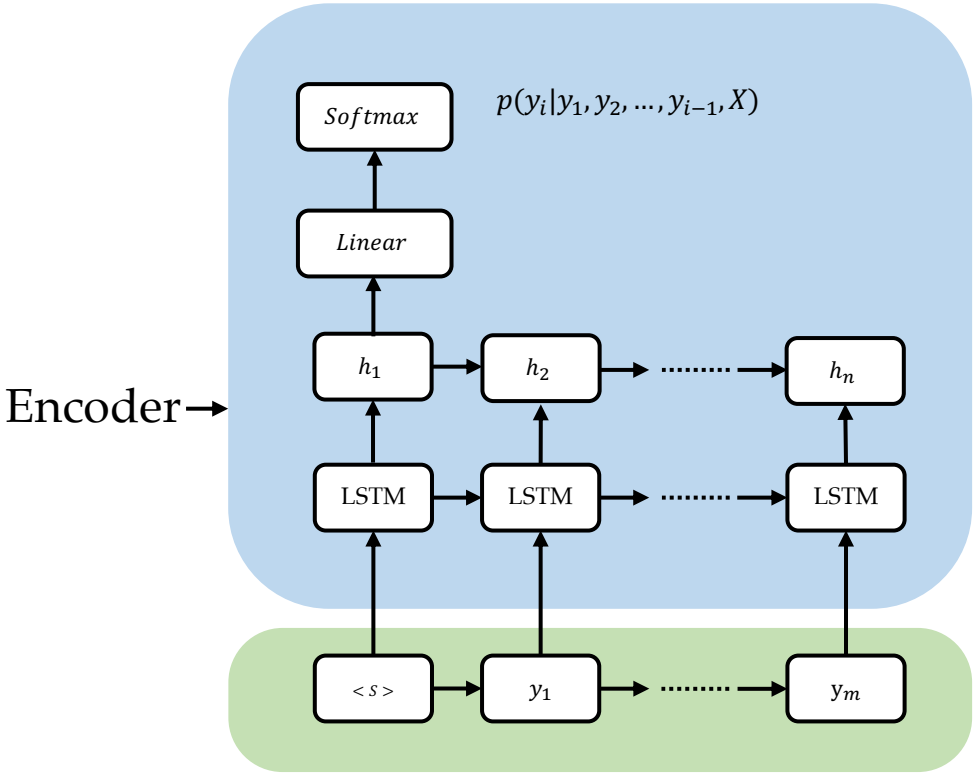
$$\mathbf{p}_1 = softmax(\mathbf{o}_1) \quad \mathbf{p}_1 \in \mathbb{R}^{|V|}$$

$$P(y_1|X_{1:n}) = \mathbf{p}_1[y_1]$$

Model 1: Seq2seq using LSTM

WestlakeNLP

- Decoder: each subsequent step i



$$\mathbf{h}_i^{dec} = LSTM\left(\mathbf{h}_{i-1}^{dec}, emb'(y_{i-1})\right)$$

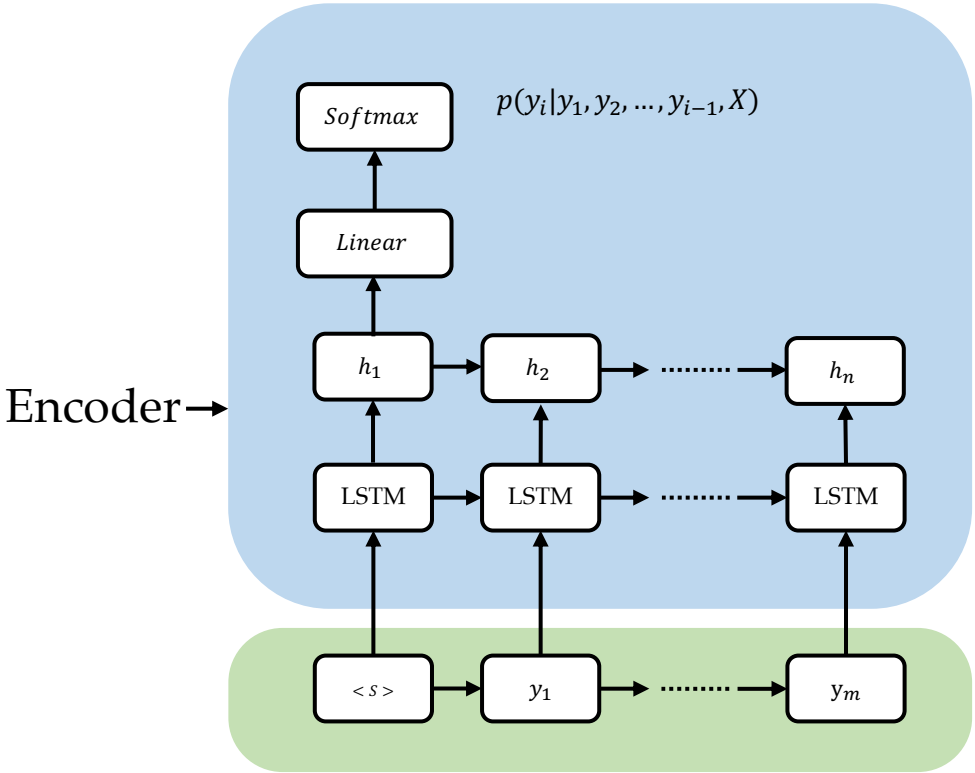
$$\mathbf{o}_i = \mathbf{W}\mathbf{h}_i^{dec}$$

$$\mathbf{p}_i = softmax(\mathbf{o}_i)$$

Model 1: Seq2seq using LSTM

WestlakeNLP

- Decoder: each subsequent step i



$$\mathbf{h}_i^{dec} = LSTM\left(\mathbf{h}_{i-1}^{dec}, emb'(y_{i-1})\right)$$

$$\mathbf{o}_i = \mathbf{W}\mathbf{h}_i^{dec}$$

$$\mathbf{p}_i = softmax(\mathbf{o}_i)$$

A greedy local Greedy search:

$$y_i = argmax_{y_{i'}} P(y_{i'}|X_{1:n}, Y_{1:i-1})$$

Model 1: Seq2seq using LSTM

WestlakeNLP

- Training:

Given an input sentence: $X_{1:n} = x_1, x_2, \dots, x_n$

$$L = -\sum_{i=1}^{N} \sum_{j=1}^{m_i} \log\left(P(y_j^i | X_i, Y_{i:j-1}^i)\right)$$

Generating an output sentence: $Y_{1:m} = y_1, y_2, \dots, y_m$

Model 1: Seq2seq using LSTM

WestlakeNLP

- Target vocabulary
  - OOV: Vocabulary, in the scale of 10^4 is very large. So highly infrequent words are not considered to make the model smaller.
  - <UNK>: adding a special token <UNK> to address OOV problem, so that gradients can be computed.
  - Disallowing the <UNK> from being generated may lead to inconsistency between training and testing. So some post-processing can be conducted.

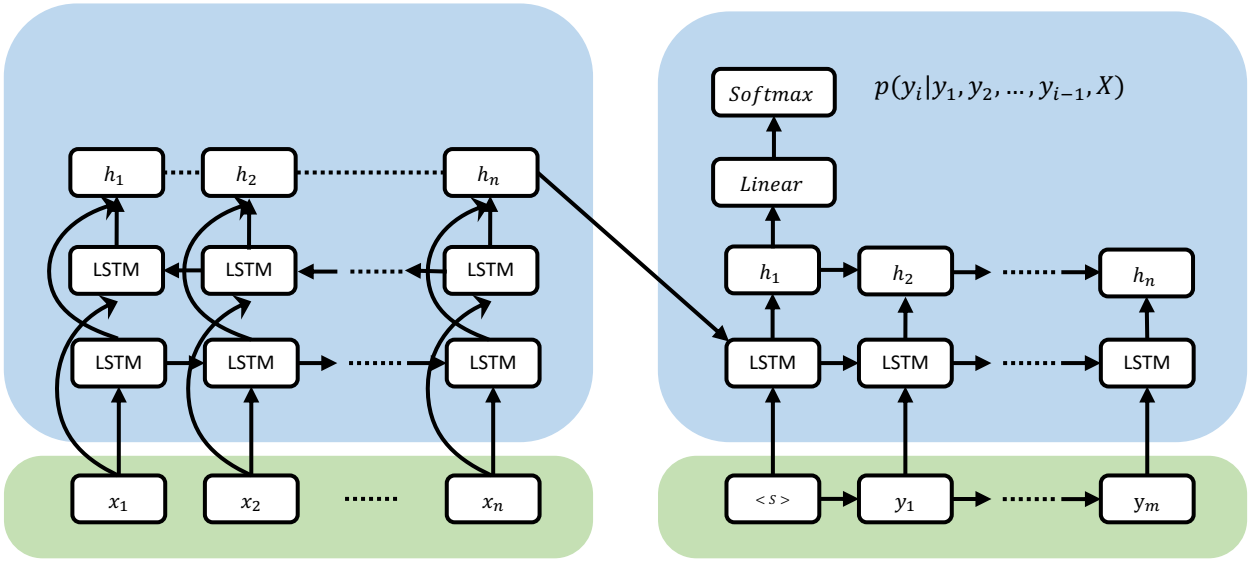
Contents

WestlakeNLP

Model 2: LSTM using attention

WestlakeNLP

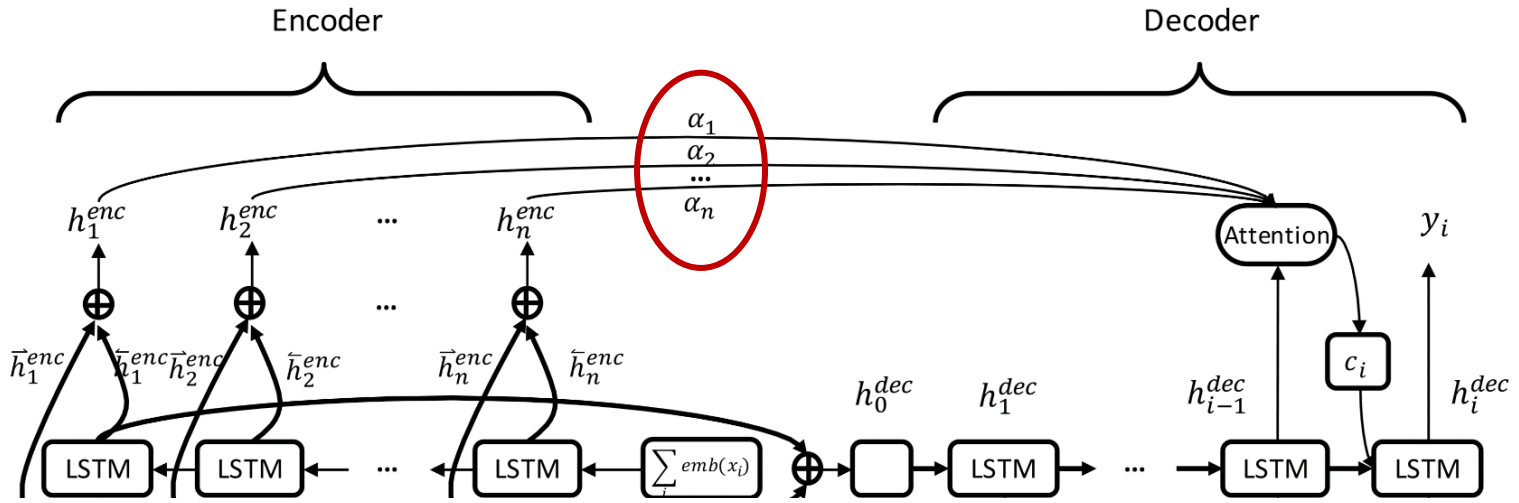
- Lack of model 1
  - Model 1 gives competitive results for MT and summarization, however, source representation vector $\mathbf{h}^{enc}$ is the only connection point.
  - RNNs face challenges in maintaining long-range dependencies between target and source words, when sentences are too long.



Model 2: LSTM using attention

WestlakeNLP

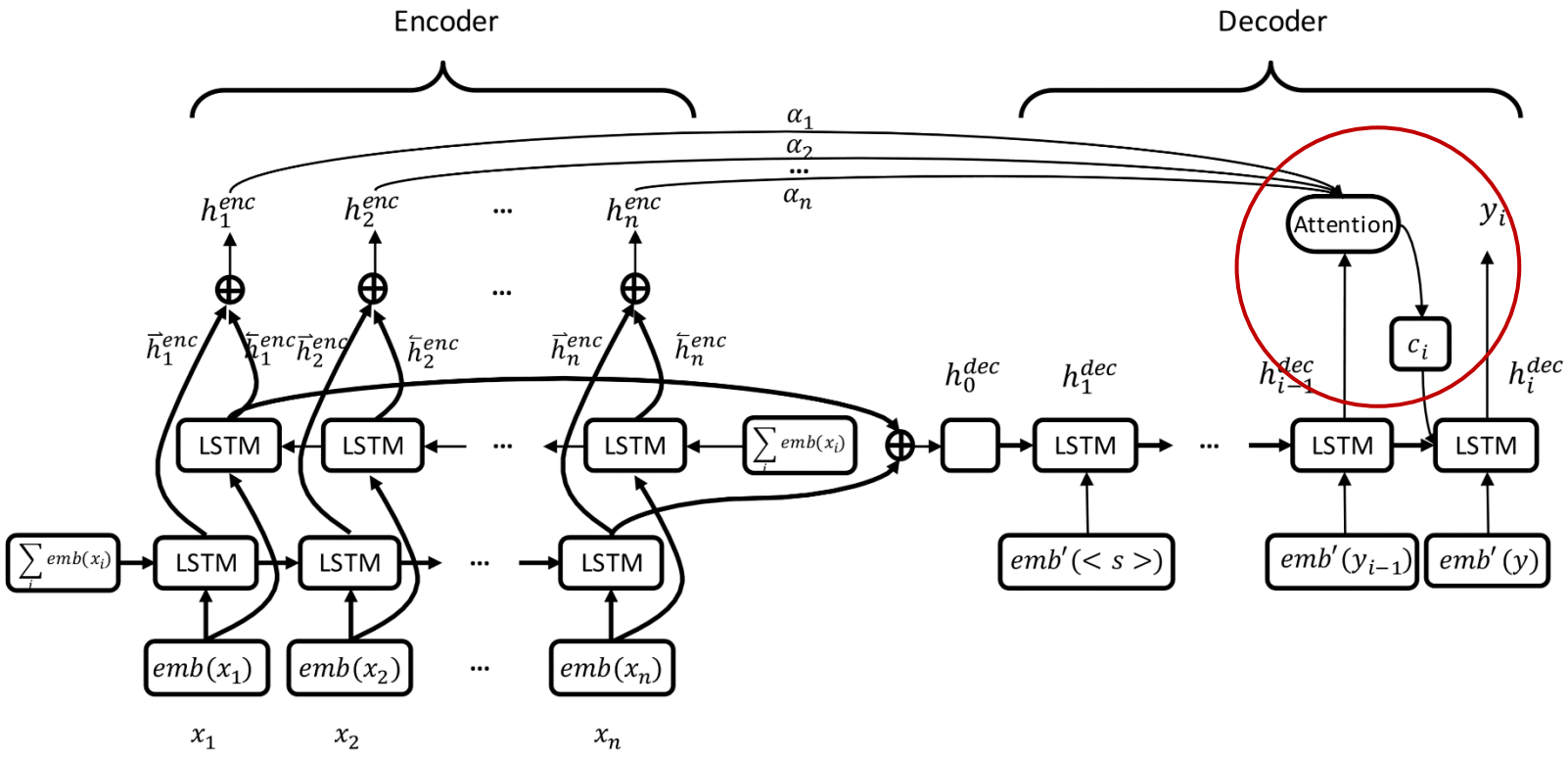
- Intuition beyond model 1
  - Keep the source at hand in each generating step.
  - Consider the hidden states for each source word simultaneously as the representation of whole source sentence rather than using only the last hidden states.



Model 2: LSTM using attention

WestlakeNLP

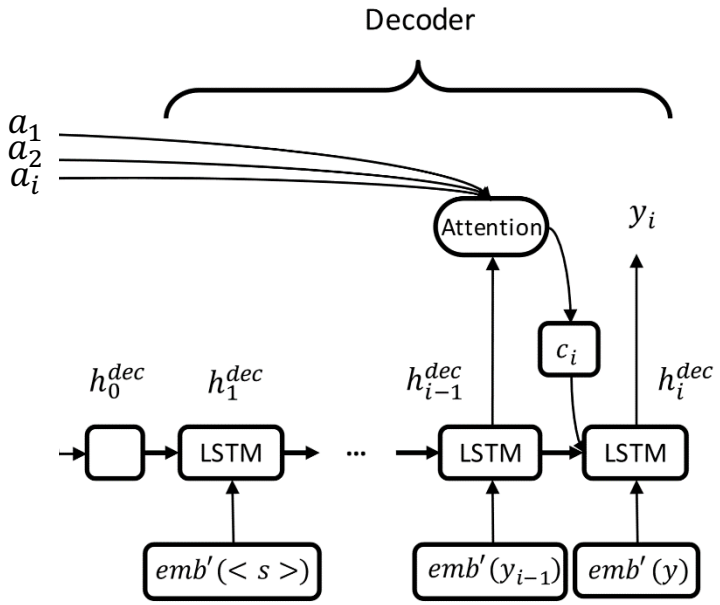
- Construct a context vector using attention mechanism



Model 2: LSTM using attention

WestlakeNLP

- Construct a context vector using attention mechanism



$$\mathbf{c}_i = \sum_{j=1}^{n} \alpha_{ij} \mathbf{h}_j^{enc}$$

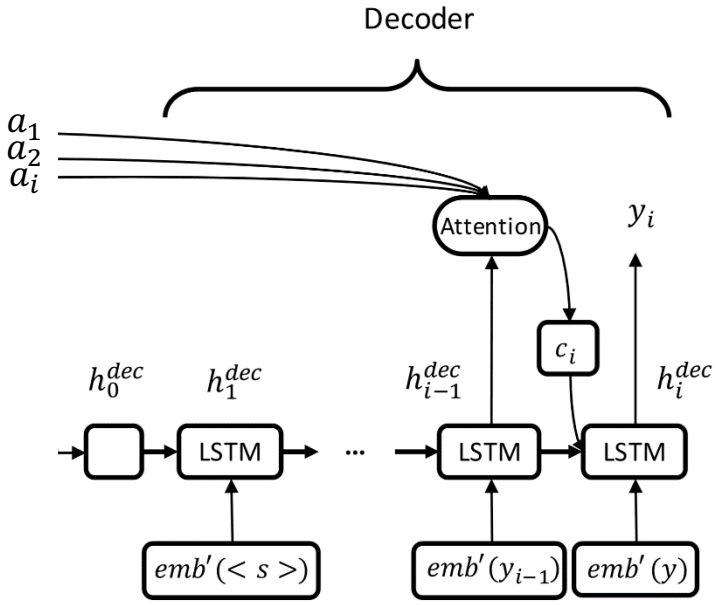
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

$$e_{ij} = \mathbf{v}_a^T \tanh(\mathbf{W}_a \mathbf{h}_{i-1}^{dec} + \mathbf{U}_a \mathbf{h}_j^{enc})$$

Model 2: LSTM using attention

WestlakeNLP

- Construct a context vector using attention mechanism



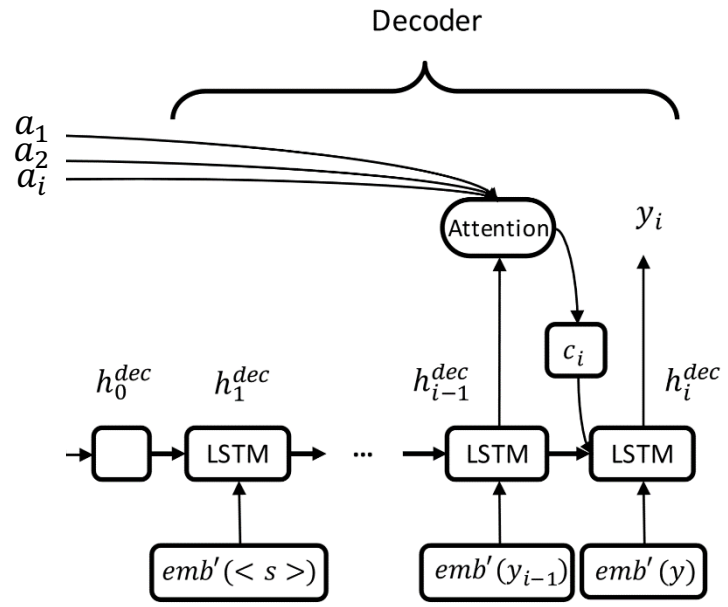
Introducing c_i into LSTM gates respectively:

$$\mathbf{u}_i = \sigma(\mathbf{W}_u emb'(y_{i-1}) + \mathbf{U}_u \mathbf{h}_{i-1}^{dec} + \mathbf{C}_u \mathbf{c}_i)$$
$$\mathbf{f}_i = \sigma(\mathbf{W}_f emb'(y_{i-1}) + \mathbf{U}_f \mathbf{h}_{i-1}^{dec} + \mathbf{C}_f \mathbf{c}_i)$$
$$\mathbf{o}_i = \sigma(\mathbf{W}_o emb'(y_{i-1}) + \mathbf{U}_o \mathbf{h}_{i-1}^{dec} + \mathbf{C}_o \mathbf{c}_i)$$
$$\tilde{\mathbf{c}}_i^{dec} = \mathbf{u}_i \circ \tanh(\mathbf{W}_h emb'(y_{i-1}) + \mathbf{U}_h \mathbf{h}_{i-1}^{dec} + \mathbf{C}_h \mathbf{c}_i) + \mathbf{f}_i \circ \tilde{\mathbf{c}}_{i-1}^{dec}$$
$$\mathbf{h}_i^{dec} = \mathbf{o}_i \circ \tilde{\mathbf{c}}_i^{dec}$$

Model 2: LSTM using attention

WestlakeNLP

- Generation using attention mechanism as model 1



$$\mathbf{h}_i^{dec} = LSTM\left(\mathbf{h}_{i-1}^{dec}, emb'(y_{i-1})\right)$$

$$\mathbf{o}_i' = \mathbf{W}\mathbf{h}_i^{dec}$$

$$\mathbf{p}_i = softmax(\mathbf{o}_i')$$

Model 2: LSTM using attention

WestlakeNLP

- Beam search:
  - Denote the probability of an output sequence as model 1:

$$P(Y_{1:i}|X_{1:n}) = P(y_1|X_{1:n})P(y_2|y_1, X_{1:n}) \dots P(y_i|y_{1:i-1}, X_{1:n})$$

  - Search for high probability, tracking top K on each step.
  - Beam search is not guaranteed to find optimal solution.
  - But much more efficient than exhaustive search.

Model 2: LSTM using attention

WestlakeNLP

- Beam search : example (beam size: K= 2)

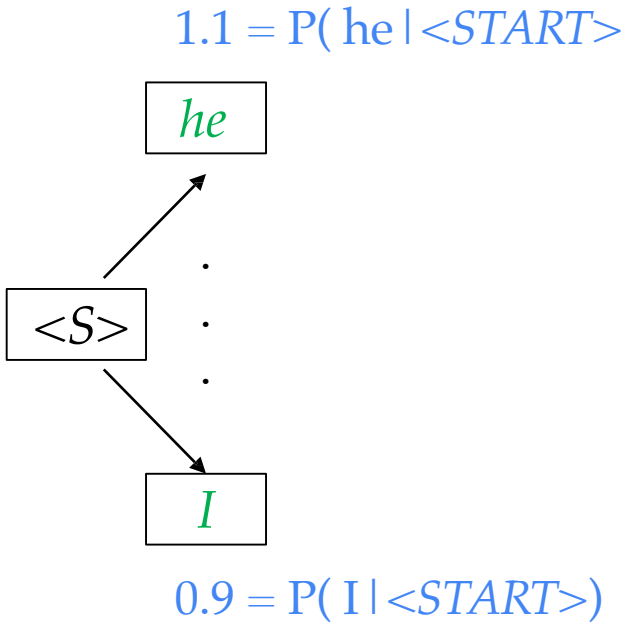
<S>

Calculate probability of next word

Model 2: LSTM using attention

WestlakeNLP

- Beam search : example (beam size: K= 2)

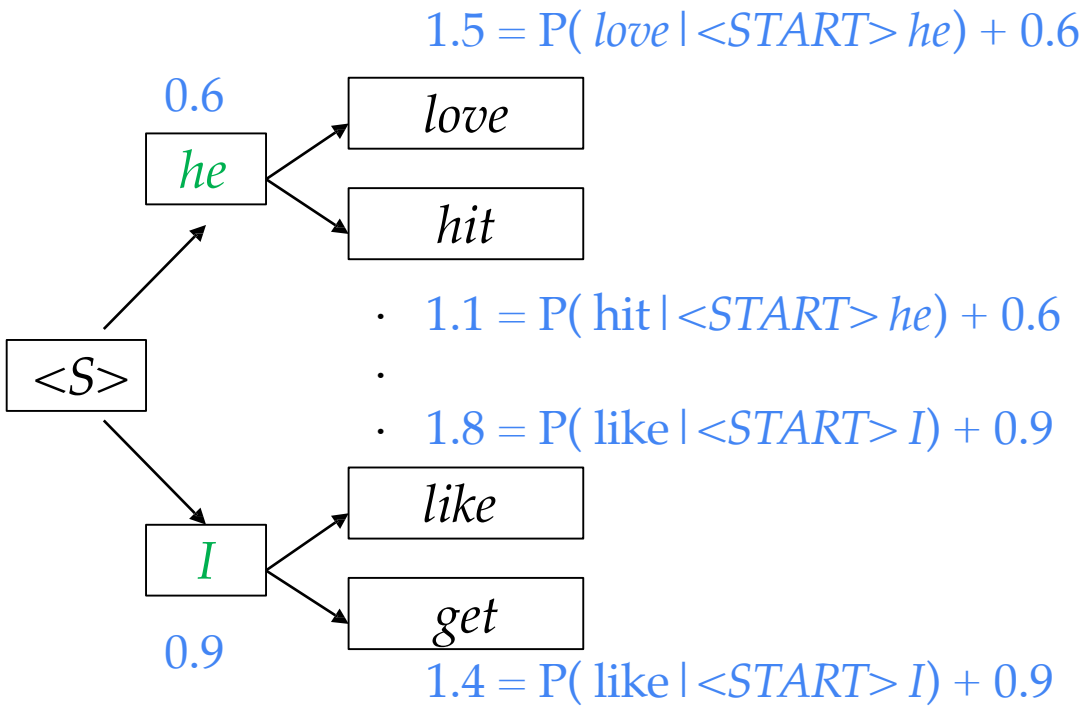


Take top K words and compute scores

Model 2: LSTM using attention

WestlakeNLP

- Beam search : example (beam size: K= 2)

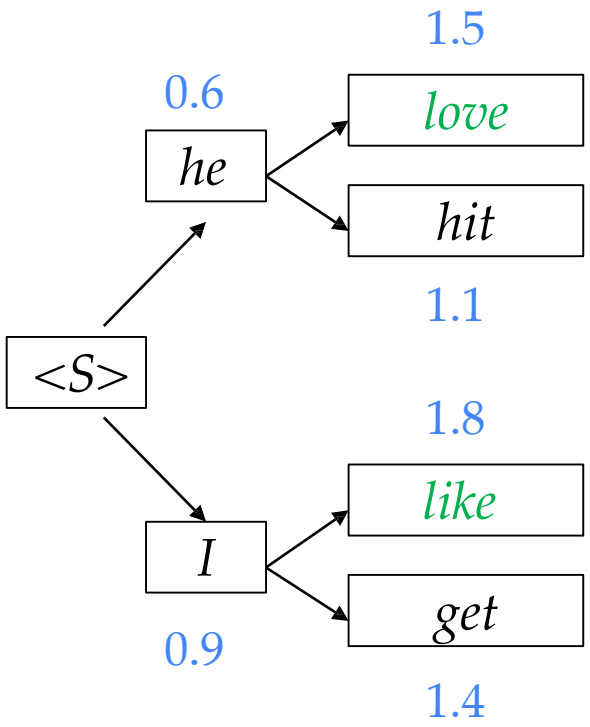


For each of the K hypotheses, find top K next words and calculate scores

Model 2: LSTM using attention

WestlakeNLP

- Beam search : example (beam size: K= 2)

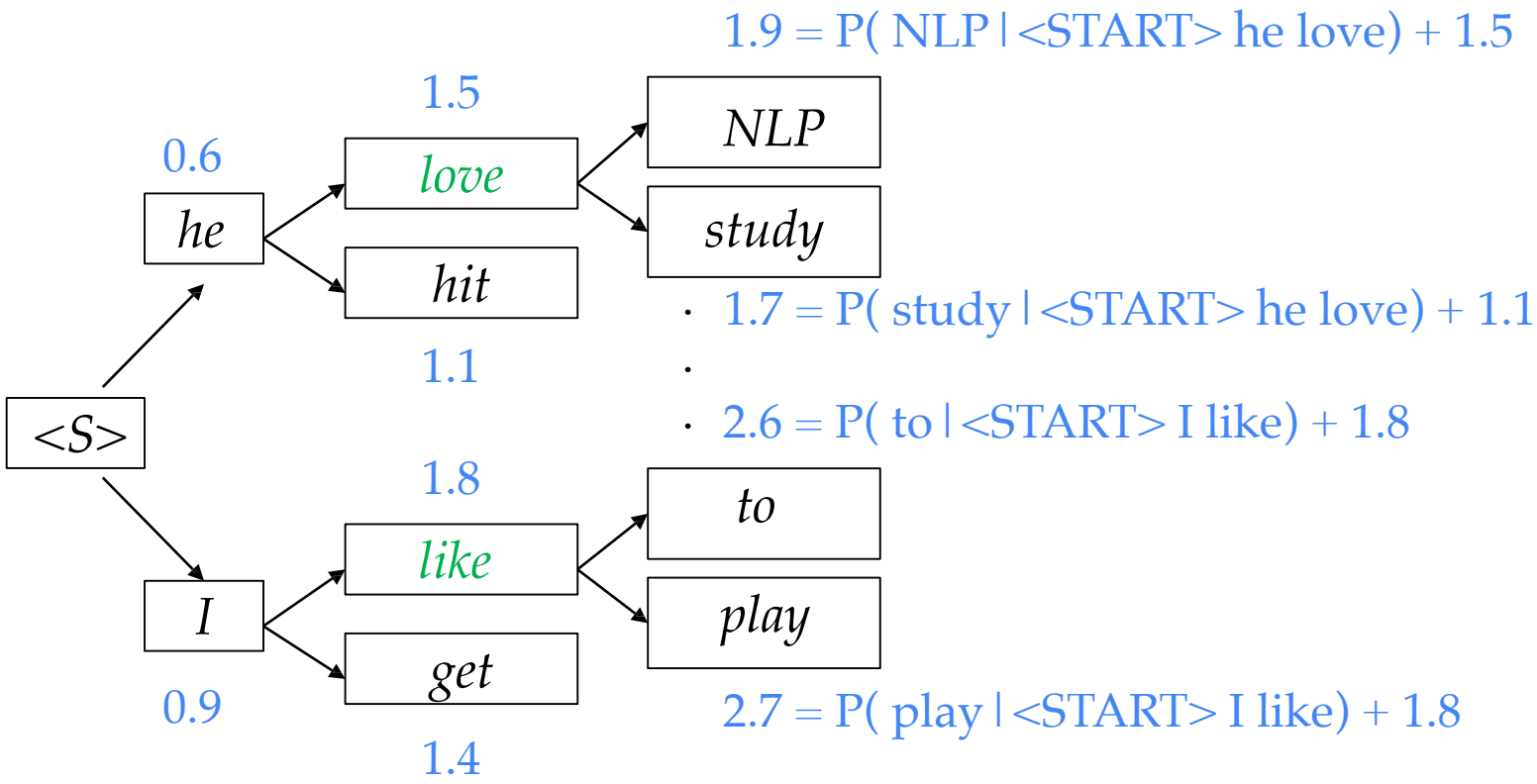


Just keep K words with highest scores

Model 2: LSTM using attention

WestlakeNLP

- Beam search : example (beam size: K= 2)

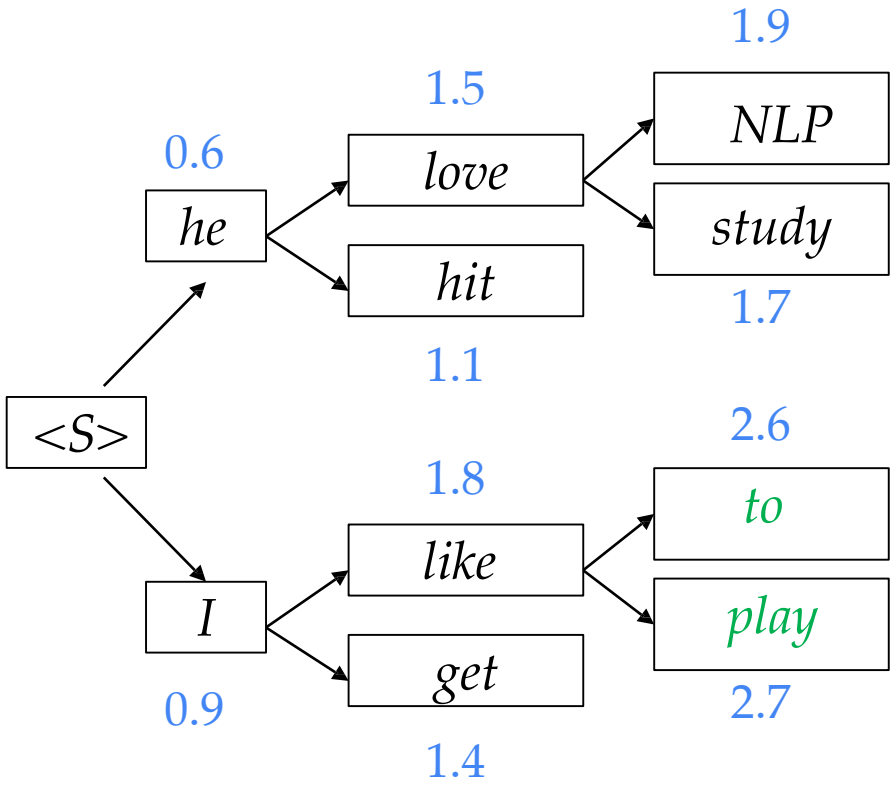


For each of the K hypotheses, find top K next words and calculate scores

Model 2: LSTM using attention

WestlakeNLP

- Beam search : example (beam size: K= 2)

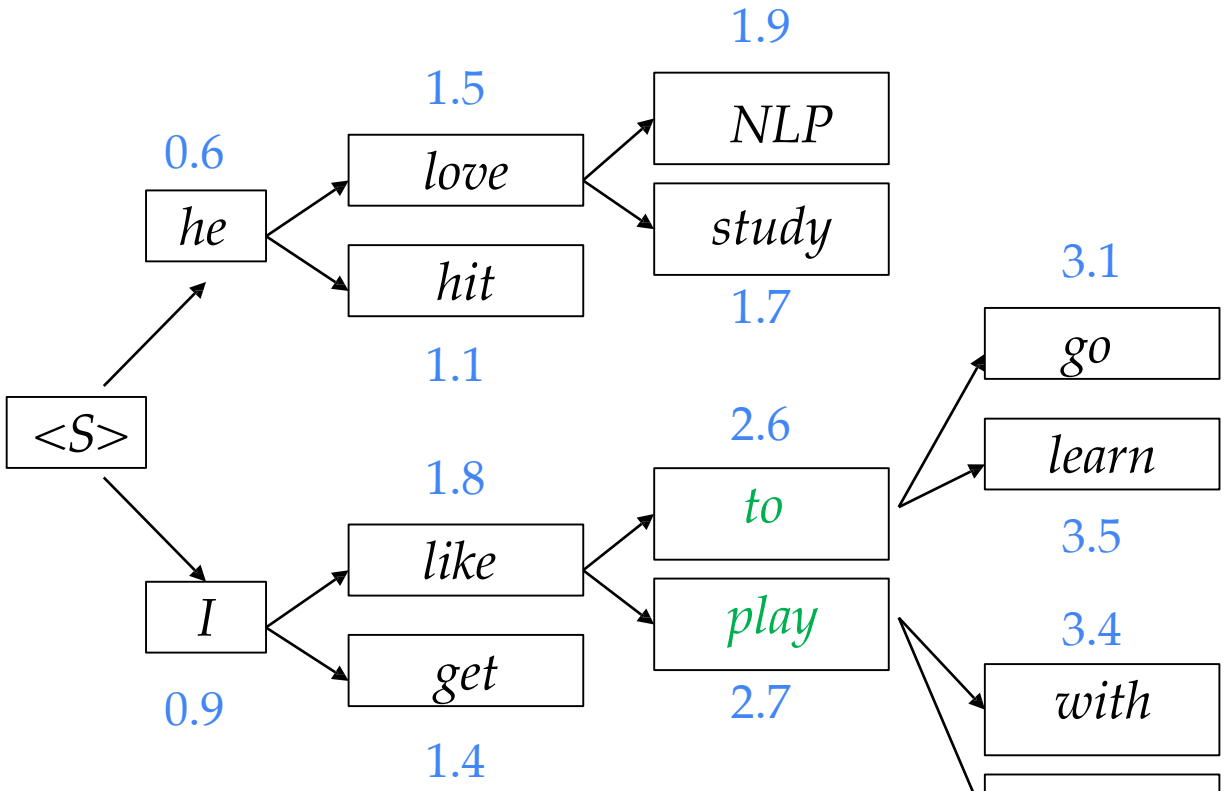


Just keep K words with highest scores

Model 2: LSTM using attention

WestlakeNLP

- Beam search : example (beam size: K= 2)

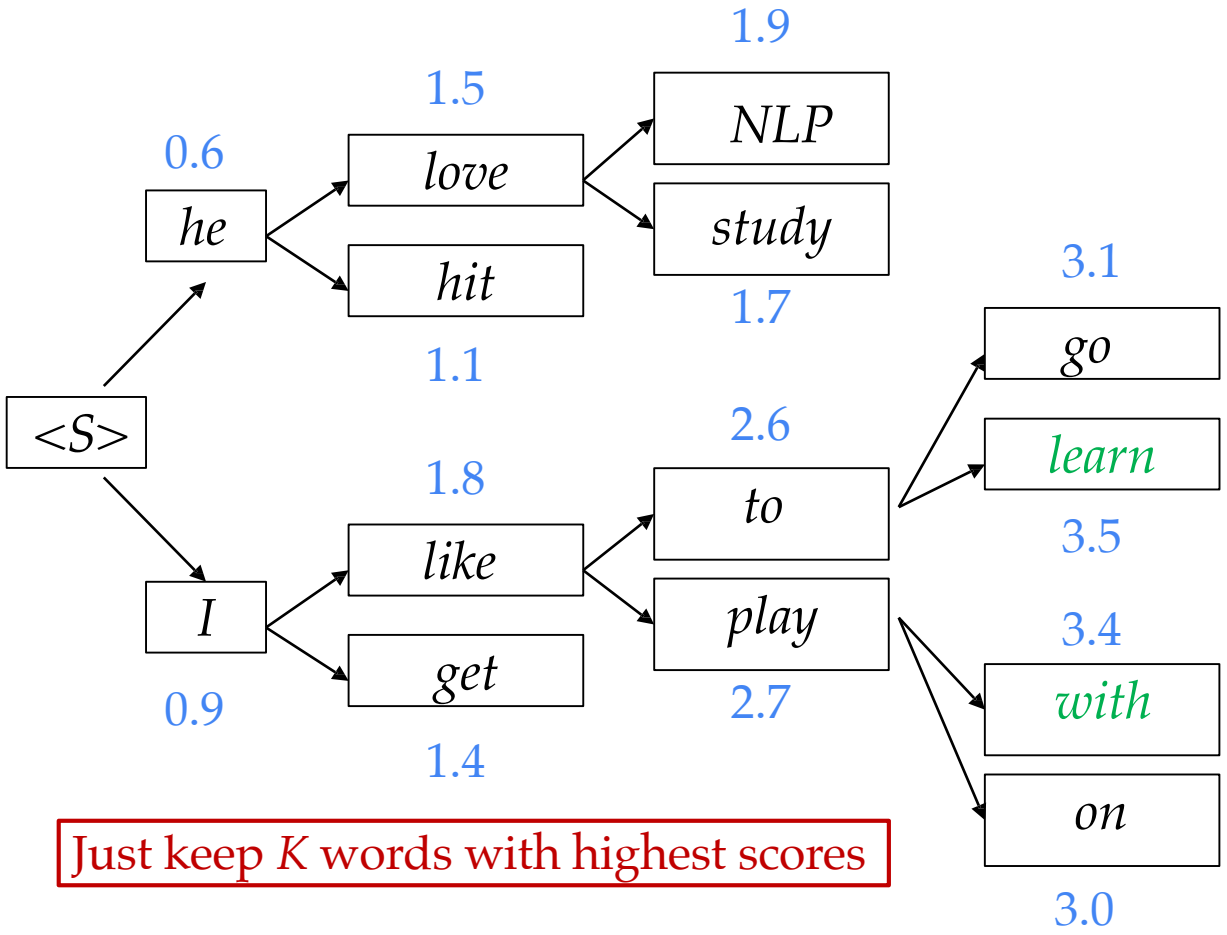


For each of the K hypotheses, find top K next words and calculate scores

Model 2: LSTM using attention

WestlakeNLP

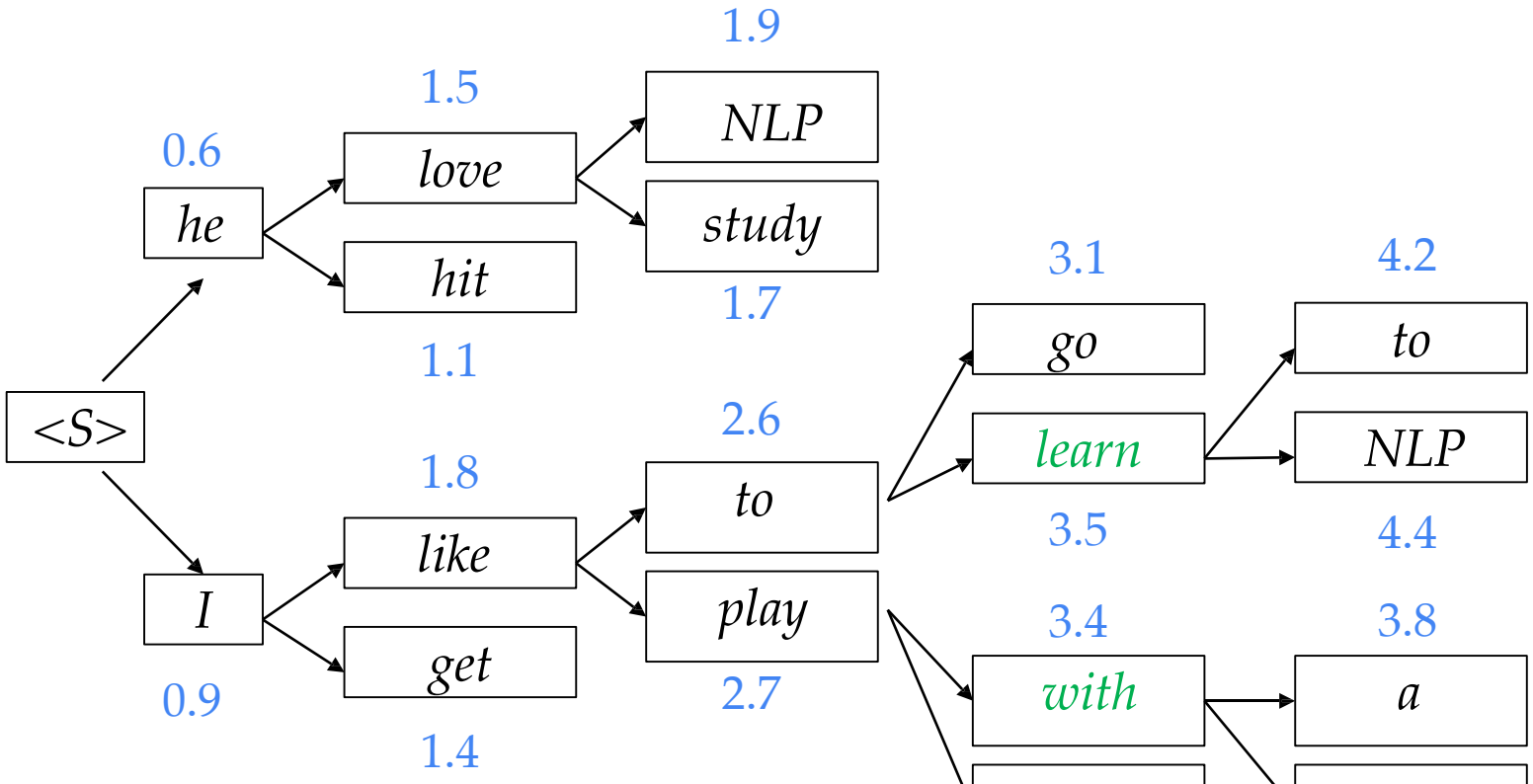
- Beam search : example (beam size: K= 2)



Model 2: LSTM using attention

WestlakeNLP

- Beam search : example (beam size: K= 2)

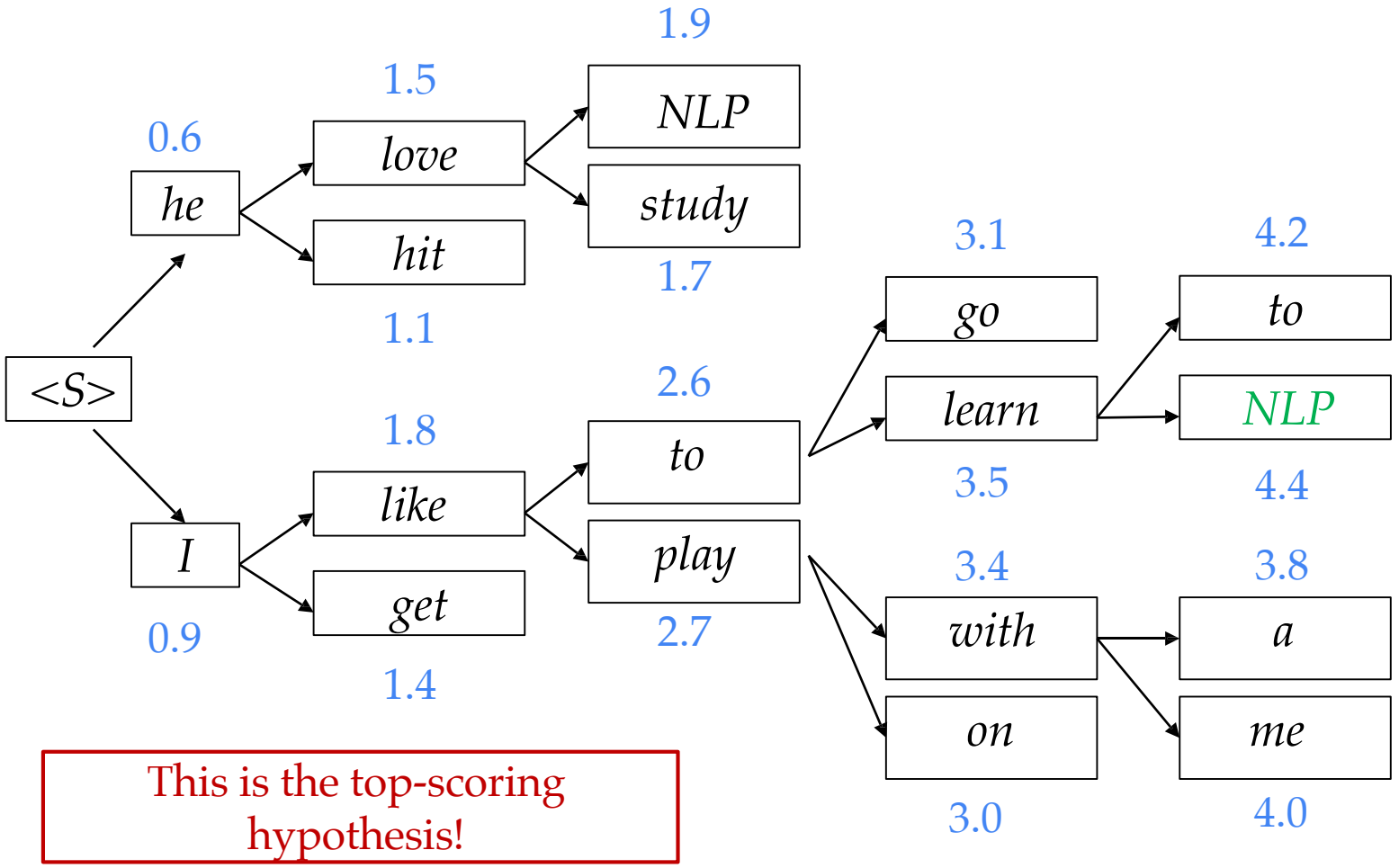


For each of the K hypotheses, find top K next words and calculate scores

Model 2: LSTM using attention

WestlakeNLP

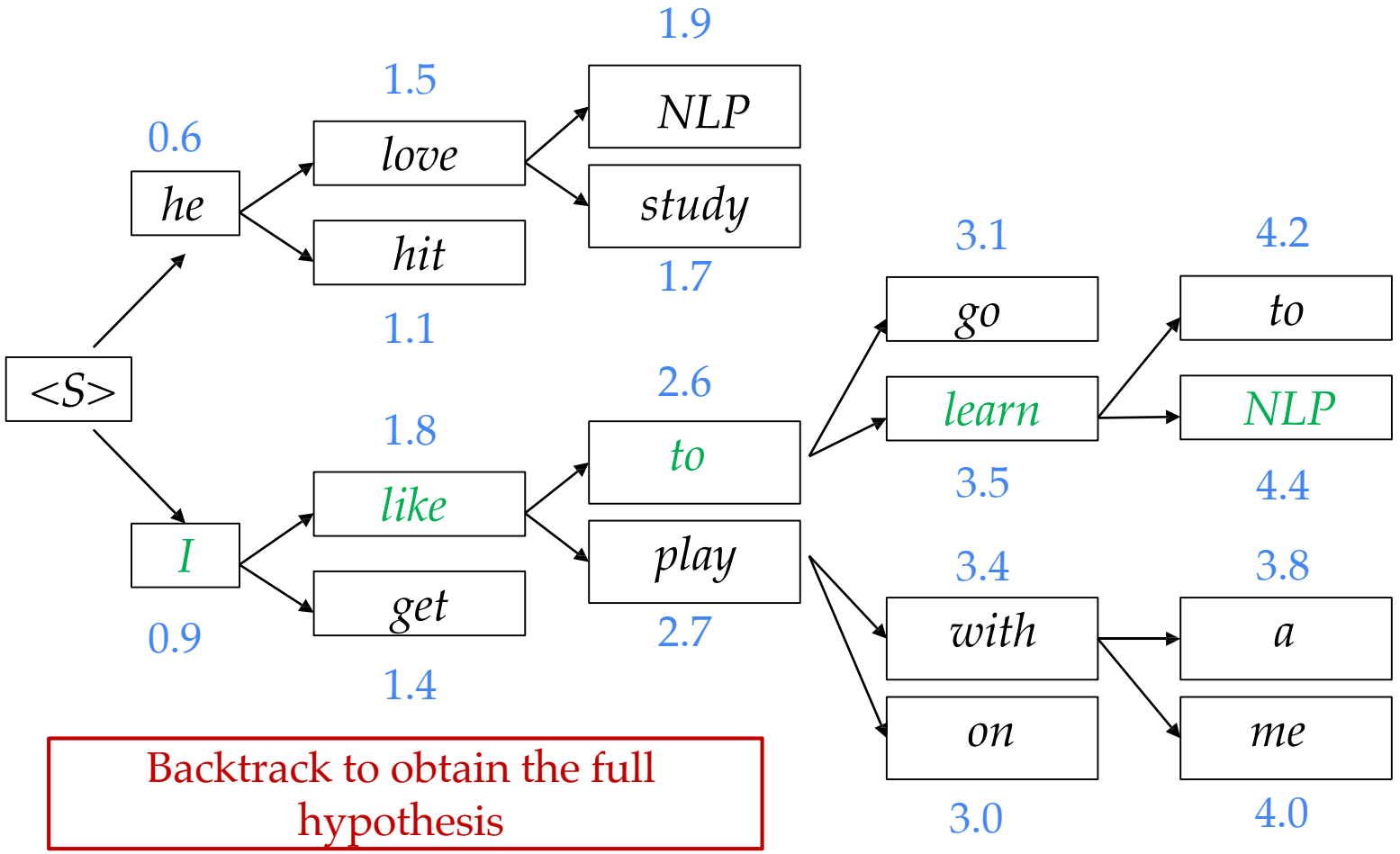
- Beam search : example (beam size: K= 2)



Model 2: LSTM using attention

WestlakeNLP

- Beam search : example (beam size: K= 2)



Backtrack to obtain the full hypothesis

Model 2: LSTM using attention

WestlakeNLP

- Beam search: tips
  - Usually we continue beam search until:
    - We reach time step T (where T is some pre-defined cutoff), or
    - We have at least n completed hypotheses (where n is pre-defined cutoff)
  - Applying pruning to each time step:
    - Enumeration of all possible candidates results in a $mK|V|$ time complexity.
    - It is efficient to conduct pruning, keeping only the top 2K candidates.
    - A type value of K in practice is 6.

Contents

Model 3: Copying from the source

WestlakeNLP

- Copying is useful
  - For machine translation, some technological terminologies can be better left.

    - I like to learn NLP. → 我喜欢学习NLP。
  - For dialogue, part of user utterance can be directly included in a system response.
    - Input: *Hello , my name is XXX.*
    - Response: *Nice to meet you, XXX.*

Model 3: Copying from the source

WestlakeNLP

- Copying: intuition
  - Basic idea: allow a copying probability to be interpolated with a vocabulary generation probability when generating each word.
    - Source vocabulary U (set of words in source sentence) and target vocabulary V → $U \cup V$)
    - The integration of copy probabilities into generation probabilities
    - Generation probability:

$$score_g(y_i = \mathbf{v}_j) = \mathbf{p}_i[\mathbf{v}_j], \mathbf{v}_j \in V$$

    - How to integrate?

Model 3: Copying from the source

WestlakeNLP

- Copying: How to integrate copy probabilities ?
  - Pointer network:
    - Reuse the attention weights as a distribution over source vocabulary U.
  - Copying network:
    - Take a dedicated neural network layer for calculating the copy score.

Model 3: Copying from the source

WestlakeNLP

- Pointer network
  - Take attention score α_{ij} as the distribution of a pointer to a source word:

$$score_c(y_i = x_k) = \alpha_{ik}, k \in [1, \dots, n]$$

  - The pointers to the same word in the source vocabulary U are then aggregated:

$$score_c(y_i = u_j) = \sum_{k:x_k = u_j} \alpha_{ik}\,, u_j \in U, j \in [1, \dots, |U|]$$

  - Finally, for the generation of y_i, the generation probability and the copy probability are linearly interpolated:

$$score(y_i) = \lambda score_g(y_i) + (1 - \lambda) score_c(y_i),$$

Model 3: Copying from the source

WestlakeNLP

- Copying network

  - A copying network calculates $score_c$ directly, by taking the current decoding state $\boldsymbol{h}_{i-1}^{dec}$ and the encoder hidden state of a source word $\boldsymbol{h}_j^{enc}$ as input:

$$score_c(y_i = u_k) = \sum_{j:x_j=u_k} \tanh\left(\mathbf{h}_j^{enc}\mathbf{W}^c\right)\mathbf{h}_{i-1}^{dec}$$

  - The probability of generating a target word y_i is calculated by the sum of a generate probability and a copy probability:

$$P(y_i|X_{1:n}, Y_{1:i-1}) = P_g(y_i|X_{1:n}, Y_{1:i-1}) + P_c(y_i|X_{1:n}, Y_{1:i-1})$$

Model 3: Copying from the source

WestlakeNLP

- Copying network
  - P_g and P_c represents normalized versions of $score_g$ and $score_c$, which are a part of a distribution P of the generation and copy:

$$P_g(y_i = v_j|X_{1:n}, Y_{1:i-1}) = \begin{cases} \frac{1}{Z}\exp(score_g(y_i = v_j)), y_i \in V \\ 0, y_i \notin V \end{cases}$$

$$P_c(y_i = u_k|X_{1:n}, Y_{1:i-1}) = \begin{cases} \frac{1}{Z}\exp(score_c(y_i = u_k)), y_i \in U \\ 0, y_i \notin U \end{cases}$$

where

$$Z = \sum_{v \in V}\exp(score_g(v)) + \sum_{u \in U}\exp(score_c(u))$$

Contents

WestlakeNLP

Subword Embedding

WestlakeNLP

- Dealing with OOV words
  - The prefix and the suffix can give us hints about the meaning of a word. Inspired of this, we can make use of subwords to enrich vocabulary instead of a <UNK> tokens in model 1, 2 and 3.
  - First consideration is morphological segmentation, however, it can be ambiguous and not able to cover all words.
  - Byte-pair encoding (BPE) is a useful algorithm:

    Most frequent byte pair → a new byte.

Subword Embedding

WestlakeNLP

- Original BPE
  - Suppose to compress the following text using BPE:

$$\boldsymbol{aabaadaab}$$

  - The byte-pair $\boldsymbol{aa}$ is the most frequent pair, which can be replaced by an unused code $\boldsymbol{Z}$:

$$\boldsymbol{ZbZdZb}$$

  - Then repeat the process with replace $\boldsymbol{Zb}$ by $\boldsymbol{Y}$:

$$\boldsymbol{YZdY}$$

  - $\boldsymbol{YZdY}$ is the final representation, since there are no pairs that occur more than once.

Subword Embedding

WestlakeNLP

- Expanding the vocabulary using BPE
  - Start with a vocabulary of characters
  - Most frequent n-gram pairs → a new n-gram

Corpus

2 b e d
2 b e t t e r
3 e a s t
4 w e s t

Vocabulary

b, e, d, t, r, a, s, w

Start with all characters in vocabulary

Subword Embedding

WestlakeNLP

- Expanding the vocabulary using BPE
  - Start with a vocabulary of characters
  - Most frequent n-gram pairs → a new n-gram

Corpus

2 b e d
2 b e t t e r
3 e a **st**
4 w e **st**

Vocabulary

b, e, d, t, r, a, s, w, **st**

Add a pair (s, t) with frequency 7

Subword Embedding

WestlakeNLP

- Expanding the vocabulary using BPE
  - Start with a vocabulary of characters
  - Most frequent n-gram pairs → a new n-gram

Corpus

2 **be** d
2 **be** t t e r
3 e a st
4 w e st

Vocabulary

b, e, d, t, r, a, s, w, st, **be**

Add a pair (b, e) with frequency 4

Subword Embedding

WestlakeNLP

- Expanding the vocabulary using BPE
  - Start with a vocabulary of characters
  - Most frequent n-gram pairs → a new n-gram

Corpus

2 be d
2 be t t e r
3 e a st
4 w e st

Vocabulary

b, e, d, t, r, a, s, w, st, be

Final vocablary

Subword Embedding

WestlakeNLP

- Expanding the vocabulary using BPE
  - Start with a vocabulary of characters
  - Most frequent n-gram pairs → a new n-gram

Corpus

2 be d
2 be t t e r
3 e a st
4 w e st

Vocabulary

b, e, d, t, r, a, s, w, st, be

A new OOV word: best
→ be@@ + st

Subword Embedding

WestlakeNLP

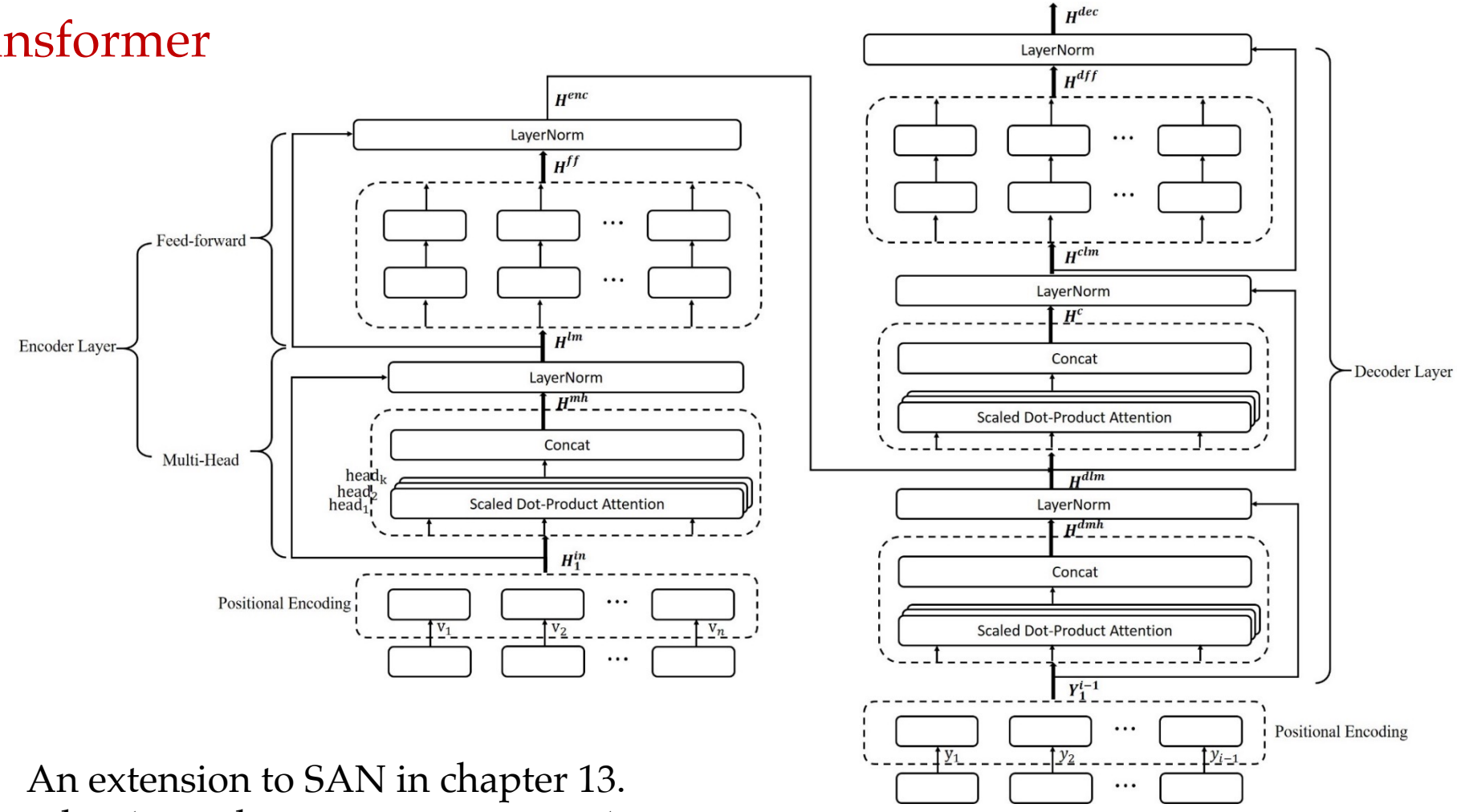
- Byte Pair Encoding: tips
  - Have a target vocabulary size and stop when you reach it.
  - Segmentation is only within words identified by some prior tokenizer.
  - Automatically decide vocabulary for system.

Contents

Seq2seq using Attention Network

WestlakeNLP

- Transformer

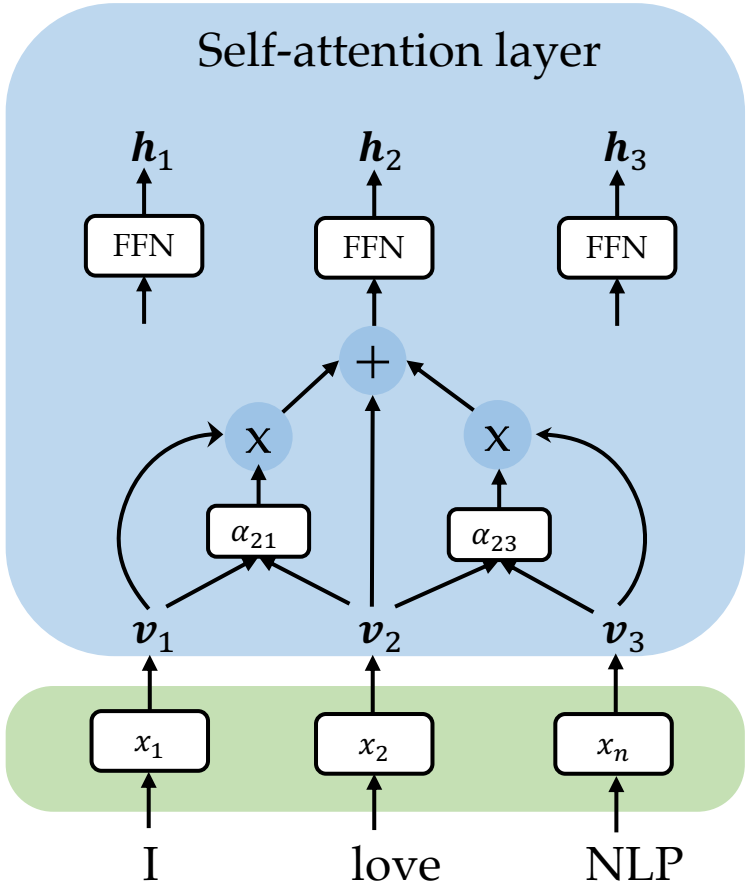


An extension to SAN in chapter 13.
It has been shown a competitive alternative to RNNs

Seq2seq using Attention Network

WestlakeNLP

- Encoder: Self-attention



Input of a self-attention layer:

$$\mathbf{V}_1^n = \mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n \qquad \mathbf{V}_1^n \in \mathbb{R}^{n \times d_h}$$

Output of a self-attention layer:

$$\mathbf{H}_1^n = \mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n \qquad \mathbf{H}_1^n \in \mathbb{R}^{n \times d_h}$$

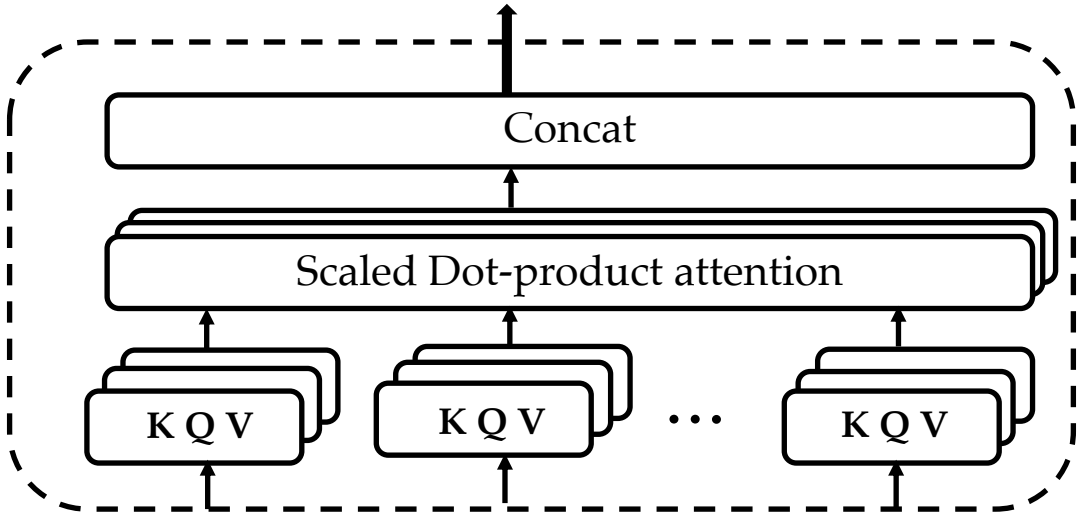
$$\mathbf{H}_1^n = attention(\mathbf{V}_1^n, \mathbf{V}_1^n, \mathbf{V}_1^n)$$

$Q = K = V$

Seq2seq using Attention Network

WestlakeNLP

- Encoder: Multi-head self-attention



$$head_j = attention(\mathbf{V}_1^n \mathbf{W}_i^{query}, \mathbf{V}_1^n \mathbf{W}_j^{key}, \mathbf{V}_1^n \mathbf{W}_j^{value})$$

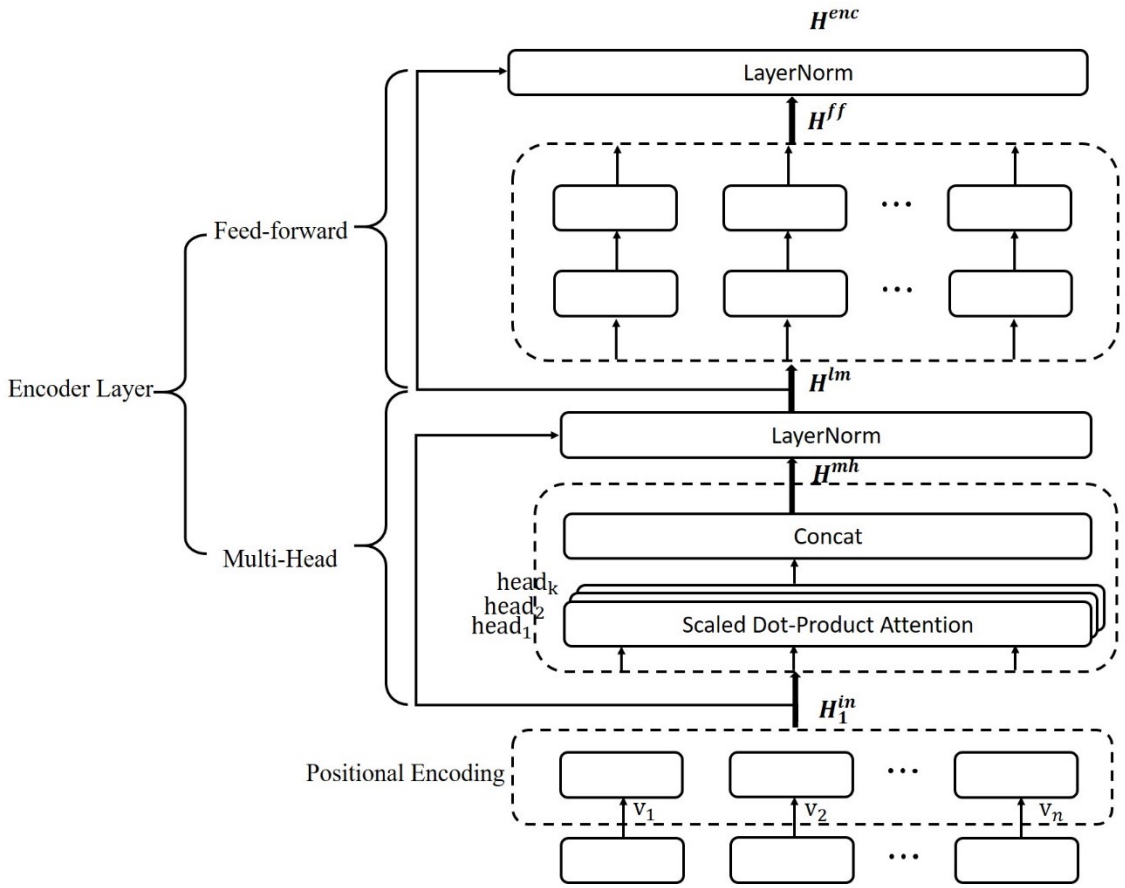
$$\mathbf{h}^{mh} = MultiHead(\mathbf{h}^{in}, \mathbf{h}^{in}, \mathbf{h}^{in})$$

$$MultiHead(\mathbf{V}_1^n, \mathbf{V}_1^n, \mathbf{V}_1^n) = concat(head_1, ..., head_k)\mathbf{W}^O$$

Seq2seq using Attention Network

WestlakeNLP

- Encoder: Layer normalization

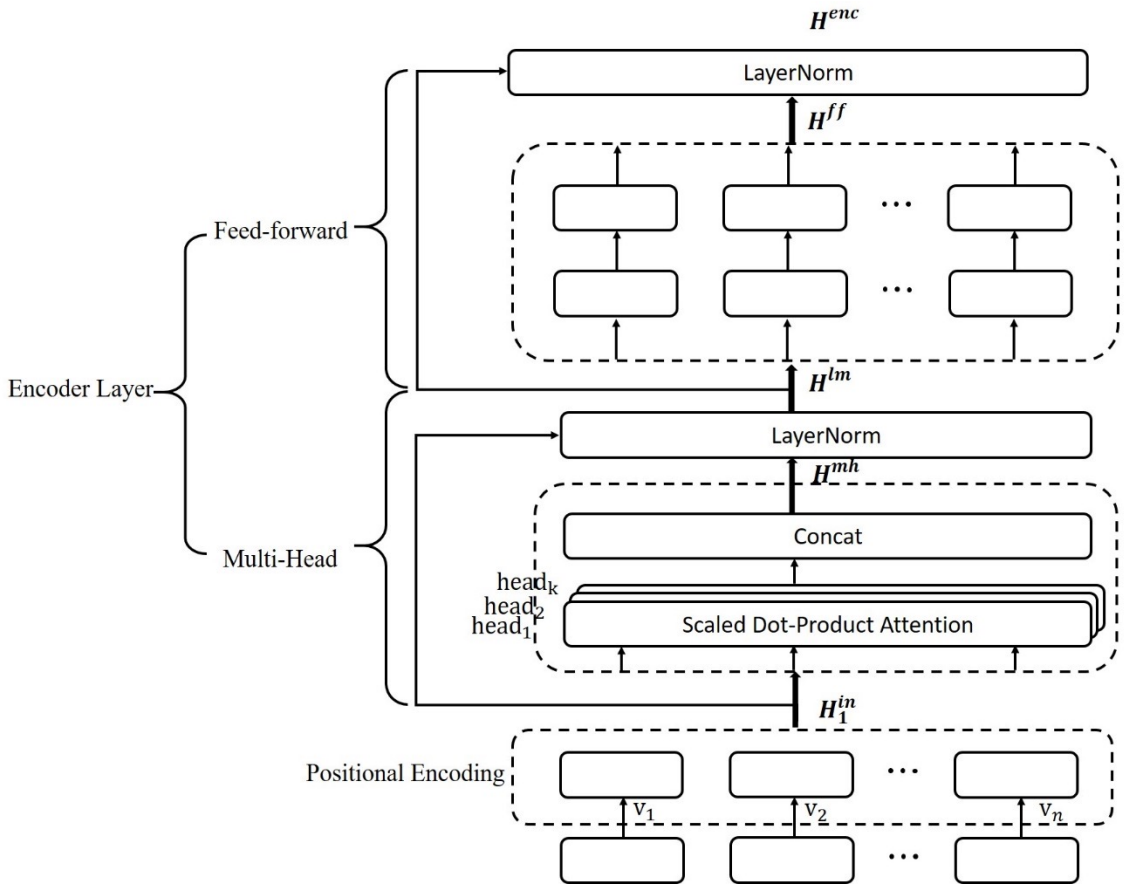


$$\mathbf{h}^{lm} = LayerNorm(\mathbf{V}_1^n + \mathbf{h}^{mh})$$

Seq2seq using Attention Network

WestlakeNLP

- Encoder: Feed-forward sub-layer



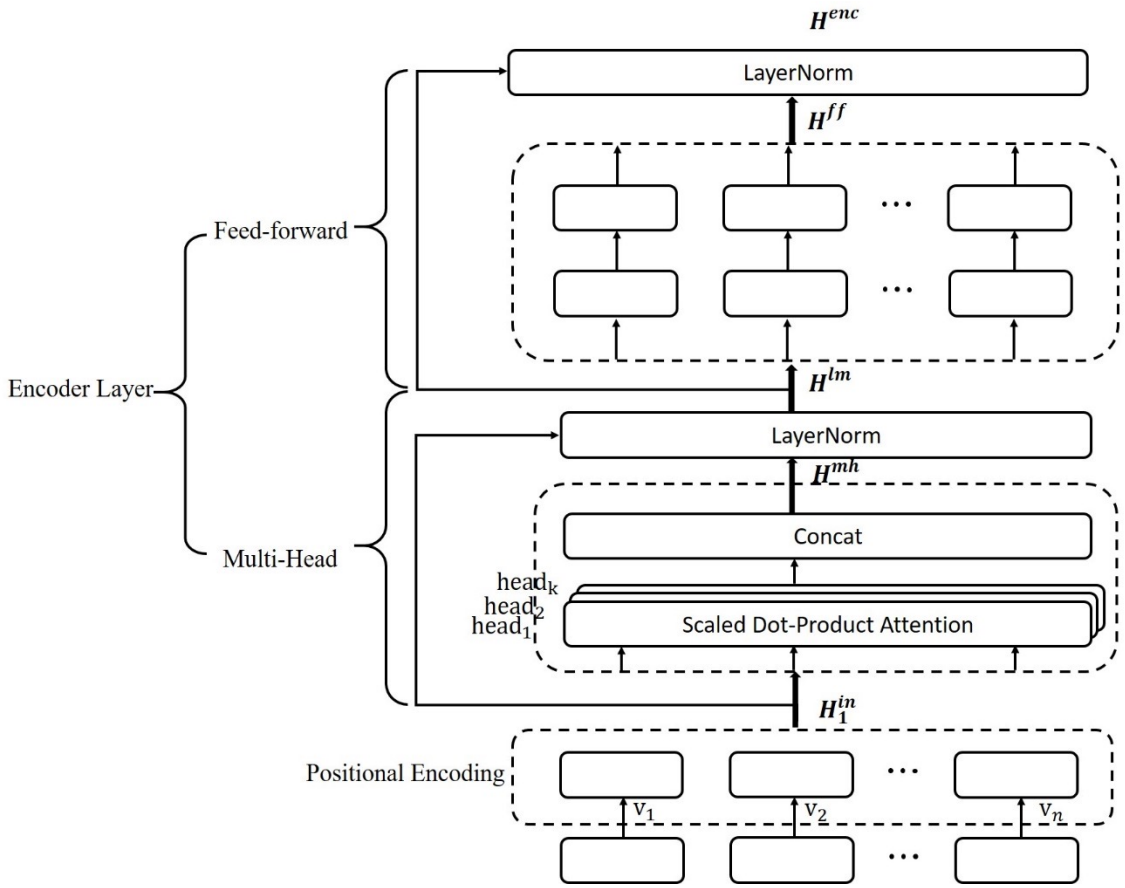
$$\mathbf{H}^{ff'} = ReLU(\mathbf{W}^{ff'}\mathbf{H}^{lm} + \mathbf{b}^{ff'})$$

$$\mathbf{H}^{ff} = \mathbf{W}^{ff}\mathbf{H}^{ff'} + \mathbf{b}^{ff}$$

Seq2seq using Attention Network

WestlakeNLP

- Encoder: output

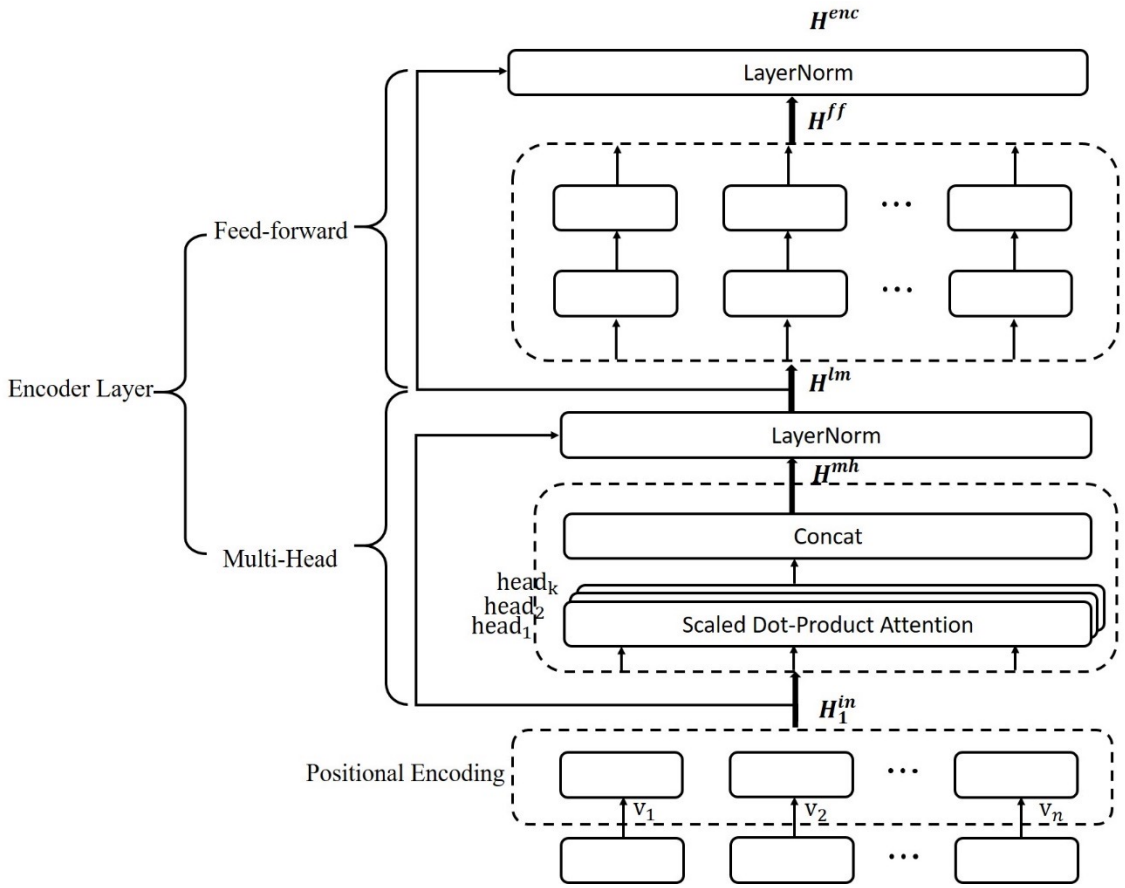


$$\mathbf{H}^{out} = LayerNorm(\mathbf{H}^{ff} + \mathbf{H}^{lm})$$

Seq2seq using Attention Network

WestlakeNLP

- Encoder: input encoding and position embedding



$$\mathbf{x}_i = emb(x_i) + \mathbf{V}_i^p$$

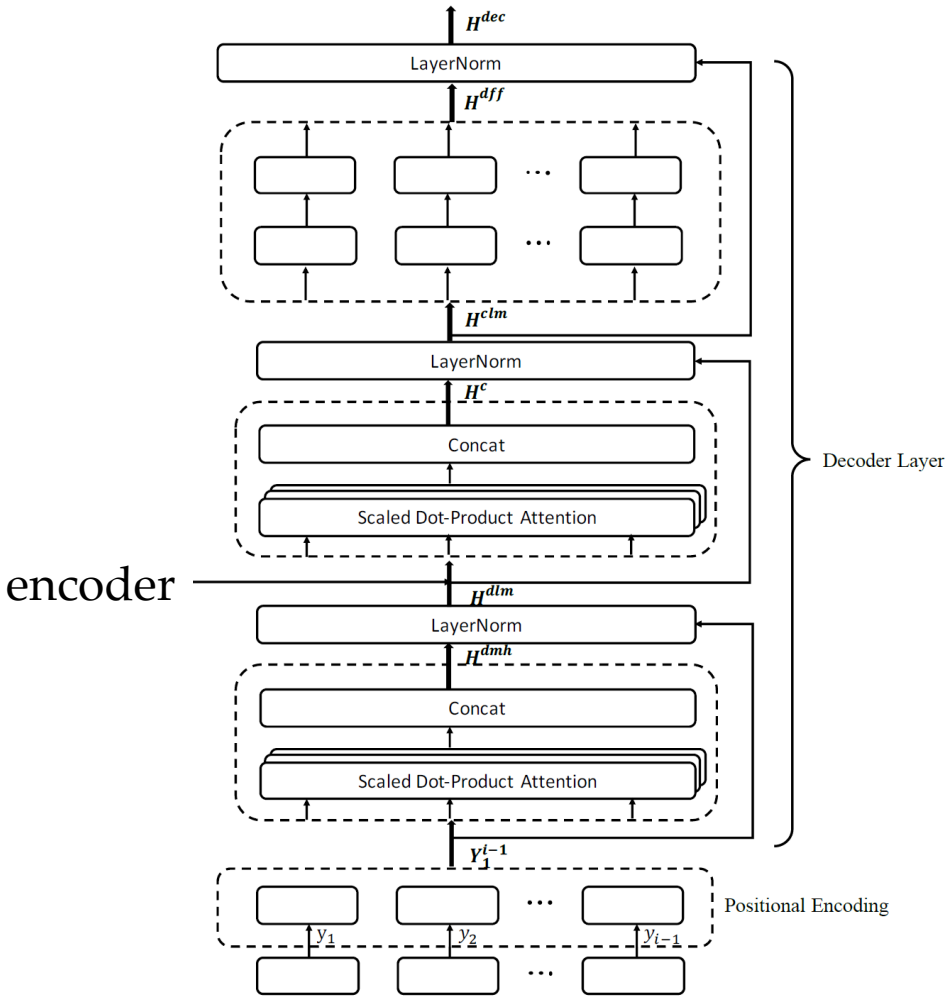
$$\mathbf{V}_i^p[2j] = \sin(i/10000^{2j/d_h})$$

$$\mathbf{V}_i^p[2j+1] = \cos(i/10000^{2j/d_h})$$

Seq2seq using Attention Network

WestlakeNLP

- Decoder: similar to encoder



- Obtain the next target word

$$\mathbf{o}_i = \mathbf{W}^{dec}\mathbf{h}_{i-1}^{dec}$$
$$\mathbf{p}_i = softmax(\mathbf{o}_i)$$

- MLP layer

$$\mathbf{H}^{dff'} = ReLU(\mathbf{W}^{dff'}\mathbf{H}^{clm} + \mathbf{b}^{dff'})$$
$$\mathbf{H}^{dff} = \mathbf{W}^{dff}\mathbf{H}^{dff'} + \mathbf{b}^{dff'}$$
$$\mathbf{H}^{dec} = LayerNorm(\mathbf{H}^{dff} + \mathbf{H}^{clm})$$

- Target to source attention

$$\mathbf{H}^{c} = MultiHead(\mathbf{H}^{dlm}, \mathbf{H}^{enc}, \mathbf{H}^{enc})$$
$$\mathbf{H}^{clm} = LayerNorm(\mathbf{H}^{c} + \mathbf{H}^{dlm})$$

- Target self-attention

$$\mathbf{H}^{dmh} = MultiHead(\mathbf{Y}_{1:i-1}, \mathbf{Y}_{1:i-1}, \mathbf{Y}_{1:i-1})$$
$$\mathbf{H}^{dlm} = LayerNorm(\mathbf{Y}_{1:i-1}, \mathbf{H}^{dmh})$$

- Embedding & positional encoding

$$\mathbf{Y}_{1:i-1} = [emb'(y_1), emb'(y_2), \ldots, emb'(y_{i-1})]$$

Seq2seq using Attention Network

WestlakeNLP

- Training:

Given a set of input-output sentence pairs: $D = \{(X_i, Y_i)\}|_{i=1}^{N}$

$$X_i = x_1, x_2, \ldots, x_{n_i}$$

$$Y_i = y_1^i, y_2^i, \ldots, y_{m_i}^i$$

Object function:

$$L = -\sum_{i=1}^{N} \sum_{j=1}^{m_i} \log\left(\mathbf{p}_j[y_j^i]\right) = \sum_{i=1}^{N} \sum_{j=1}^{m_i} \log\left(P(y_j^i | X_i, Y_{i:j-1}^i)\right)$$

Seq2seq using Attention Network

WestlakeNLP

- Why self attention:

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$

n = sequence length d = depth k = kernel size

Contents

WestlakeNLP

Text Matching Models

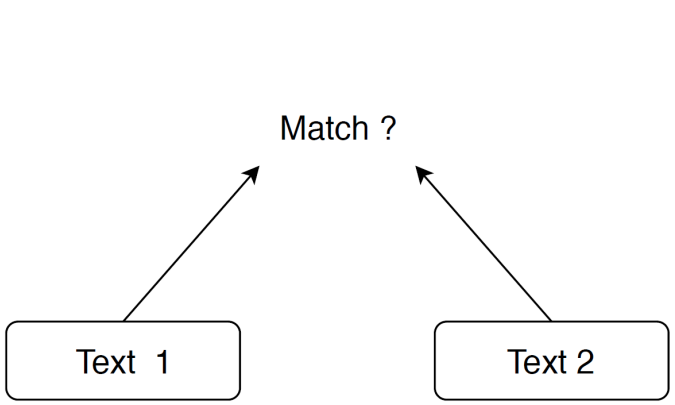
WestlakeNLP

- Text matching: whether two pieces of text semantically match each other.

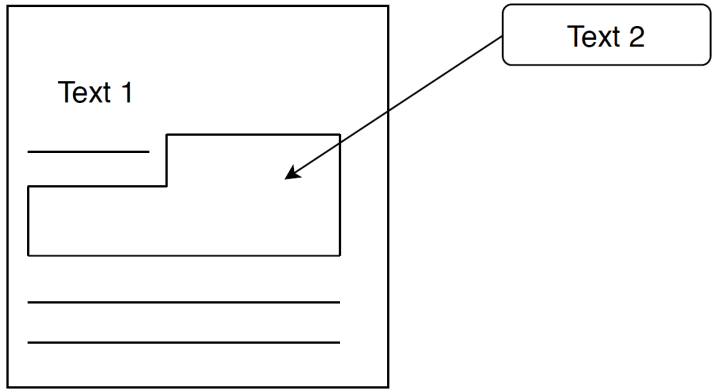
  Paraphrase detection, text entailment detection…

- Searching for a match: find a section in a given piece of text that matches the meaning of another piece of text.

  Machine reading comprehension…



(a) Matching two pieces of text



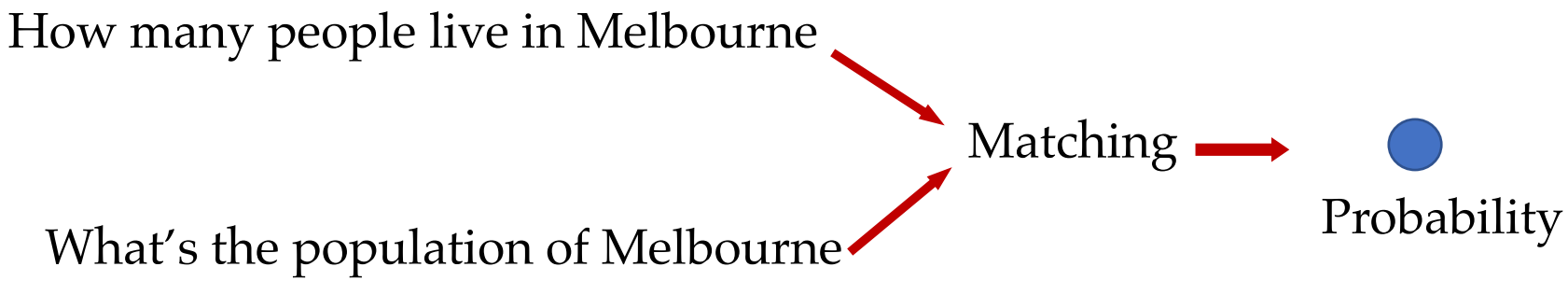
(b) Searching for a match

Contents

Text Matching

WestlakeNLP

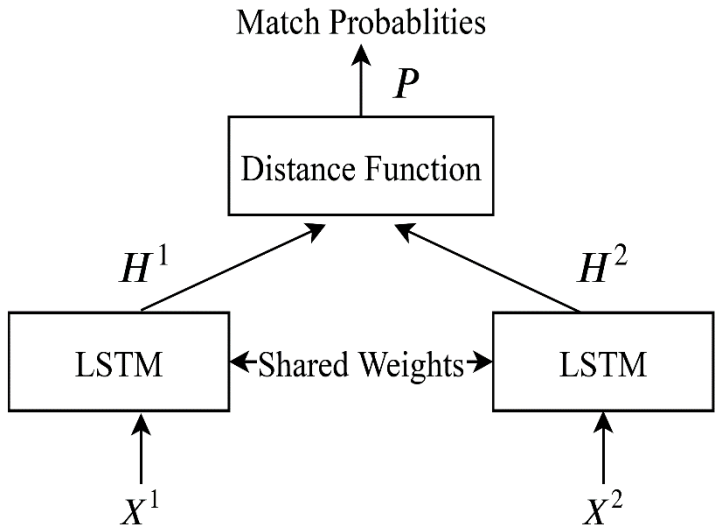
- Text matching:



Text Matching

WestlakeNLP

- Siamese network:
  - Apply identical encoders to represent both sentences.
  - Parameters are shared between both encoders.



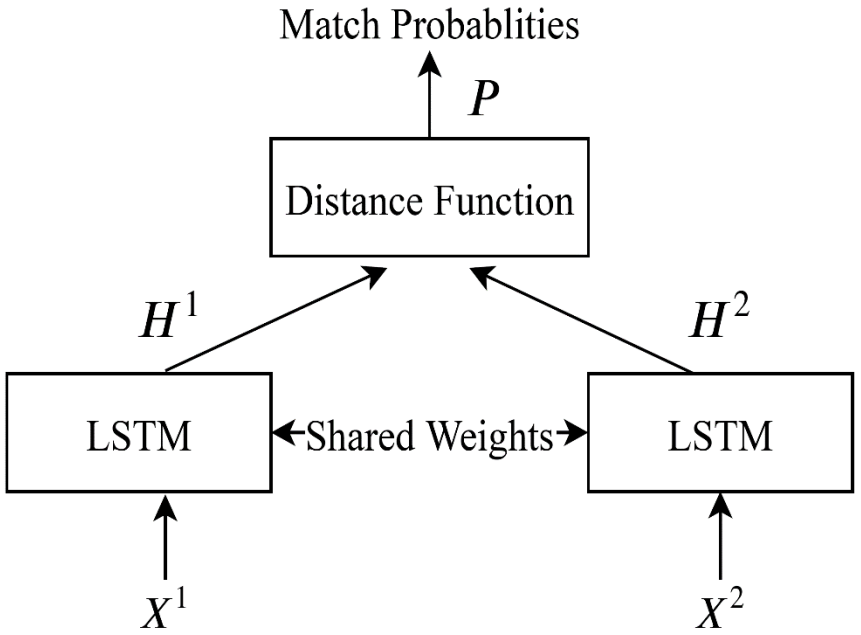
How many people live in Melbourne

What's the population of Melbourne

Text Matching

WestlakeNLP

- Siamese network:

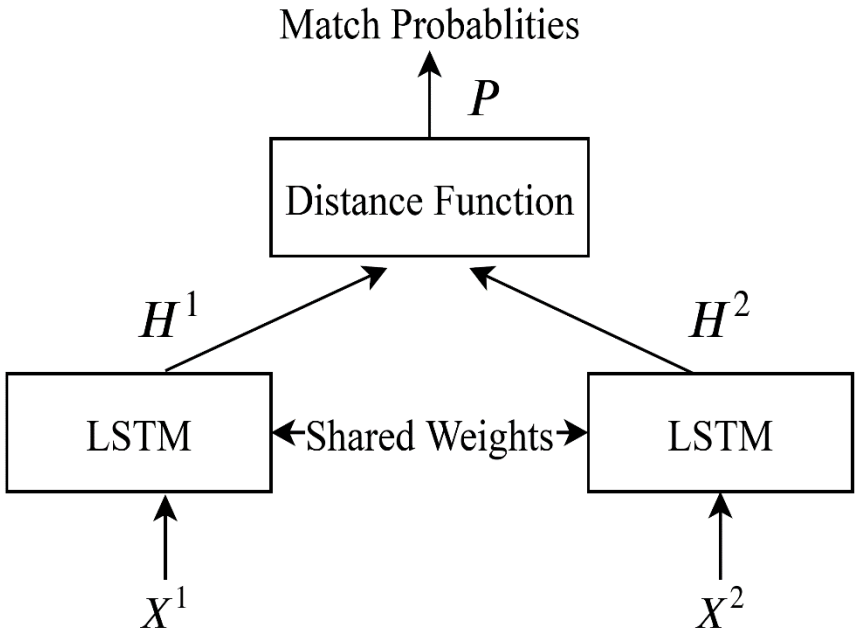


$$W^1_{1:n_1} = w^1_1, w^1_2, \dots, w^1_{n_1} \quad emb(W^1) = [emb(w^1_1); emb(w^1_2); \dots; emb(w^1_{n_1})]$$
$$W^2_{1:n_2} = w^2_1, w^2_2, \dots, w^2_{n_2} \quad emb(W^2) = [emb(w^2_1); emb(w^2_2); \dots; emb(w^2_{n_2})]$$

Text Matching

WestlakeNLP

- Siamese network:



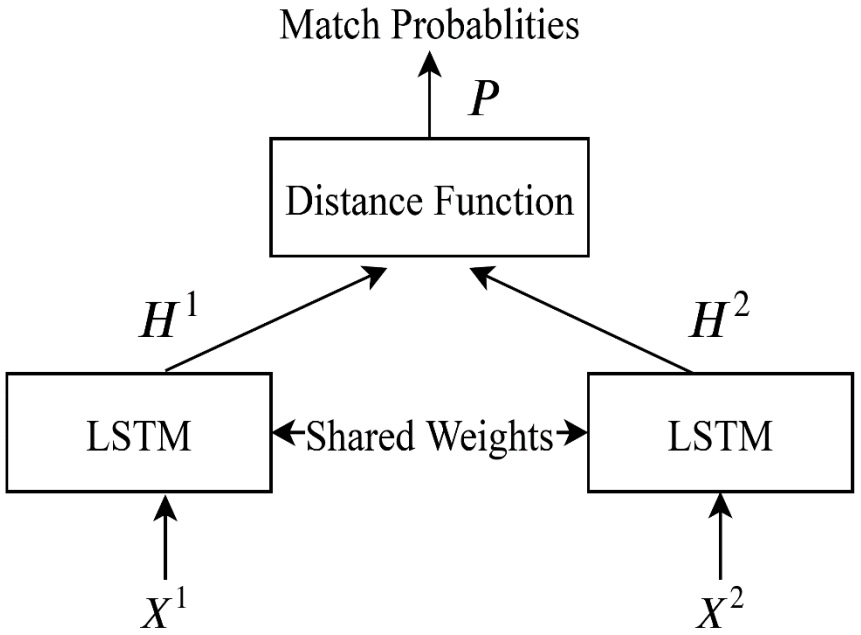
$$\mathbf{H}^1 = BiLSTM(emb(W^1))$$
$$\mathbf{H}^2 = BiLSTM(emb(W^2))$$

$$W^1_{1:n_1} = w^1_1, w^1_2, \dots, w^1_{n_1} \quad emb(W^1) = [emb(w^1_1); emb(w^1_2); \dots; emb(w^1_{n_1})]$$
$$W^2_{1:n_2} = w^2_1, w^2_2, \dots, w^2_{n_2} \quad emb(W^2) = [emb(w^2_1); emb(w^2_2); \dots; emb(w^2_{n_2})]$$

Text Matching

WestlakeNLP

- Siamese network:



$$dist(\mathbf{h}^1, \mathbf{h}^2) \in \mathbb{R}^1$$

$$\mathbf{h}^1 = [\overrightarrow{\mathbf{h}}^1_{n_1}; \overleftarrow{\mathbf{h}}^1_1]$$

$$\mathbf{h}^2 = [\overrightarrow{\mathbf{h}}^2_{n_2}; \overleftarrow{\mathbf{h}}^2_1]$$

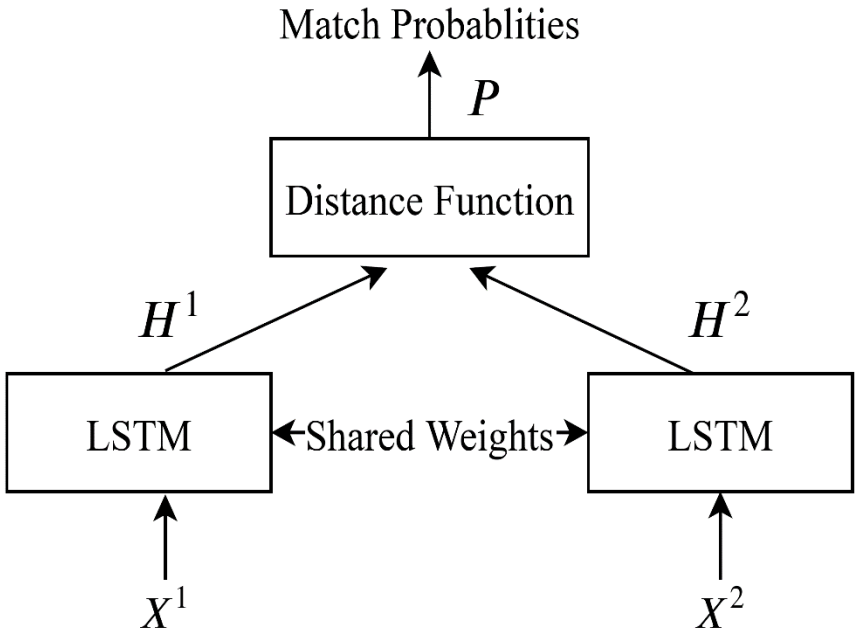
$$W^1_{1:n_1} = w^1_1, w^1_2, \dots, w^1_{n_1} \quad emb(W^1) = [emb(w^1_1); emb(w^1_2); \dots; emb(w^1_{n_1})]$$

$$W^2_{1:n_2} = w^2_1, w^2_2, \dots, w^2_{n_2} \quad emb(W^2) = [emb(w^2_1); emb(w^2_2); \dots; emb(w^2_{n_2})]$$

Text Matching

WestlakeNLP

- Siamese network:



$$P(match(W^1, W^2)) = \sigma(\mathbf{o})$$
$$\mathbf{o} = MLP(\mathbf{h}^1 \oplus \mathbf{h}^2)$$

$$\mathbf{h}^1 = [\overrightarrow{\mathbf{h}}^1_{n_1}; \overleftarrow{\mathbf{h}}^1_1]$$

$$\mathbf{h}^2 = [\overrightarrow{\mathbf{h}}^2_{n_2}; \overleftarrow{\mathbf{h}}^2_1]$$

$$W^1_{1:n_1} = w^1_1, w^1_2, \dots, w^1_{n_1} \quad emb(W^1) = [emb(w^1_1); emb(w^1_2); \dots; emb(w^1_{n_1})]$$
$$W^2_{1:n_2} = w^2_1, w^2_2, \dots, w^2_{n_2} \quad emb(W^2) = [emb(w^2_1); emb(w^2_2); \dots; emb(w^2_{n_2})]$$

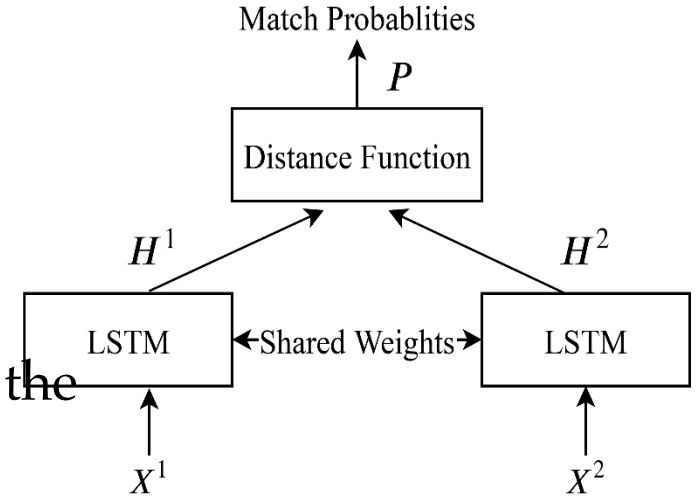
Text Matching

WestlakeNLP

- Training:

Given a set of labeled training data

$$D = \{(W_i^1, W_i^2, y_i)\}|_{i=1}^N$$



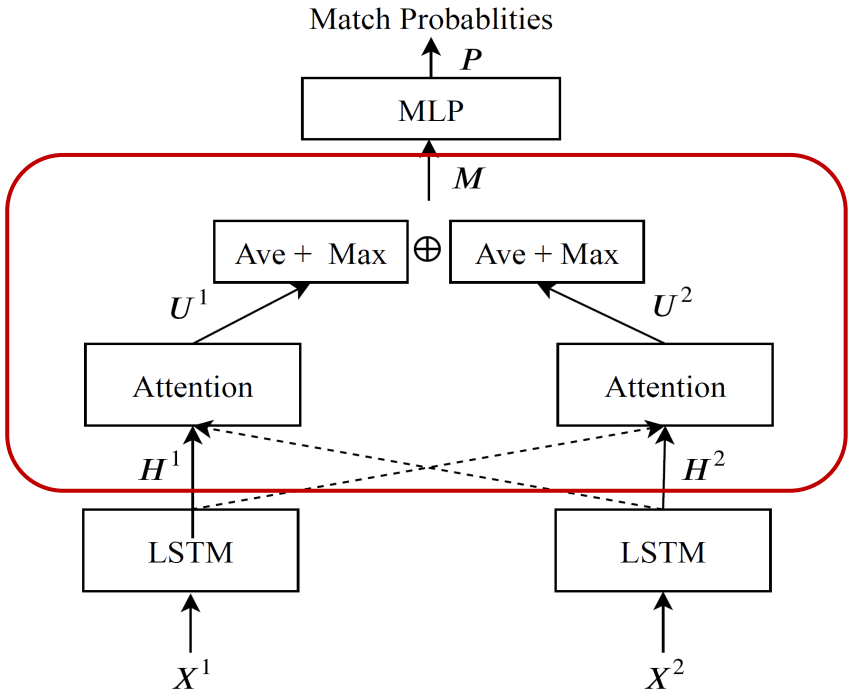
The training objective function can be to minimize the negative log-likelihood loss function

$$L = -\sum_{i=1}^{N}(y_i \log P(match(W_i^1, W_i^2)) + (1 - y_i)\log(1 - P(match(W_i^1, W_i^2))))$$

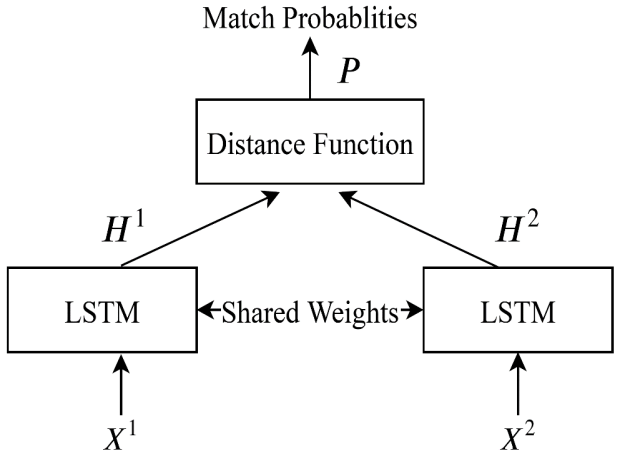
Text Matching

WestlakeNLP

- Attention matching network:



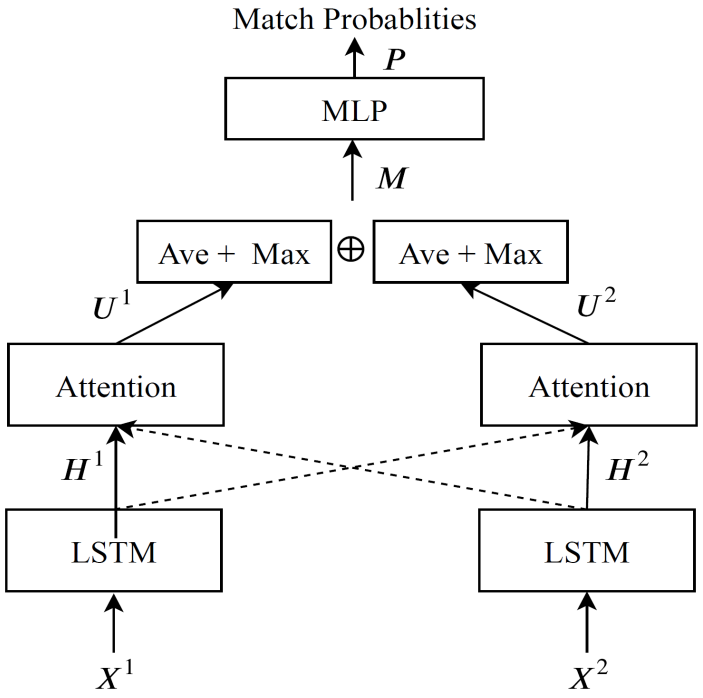
Compared to Siamese network:



Text Matching

WestlakeNLP

- Attention matching network:



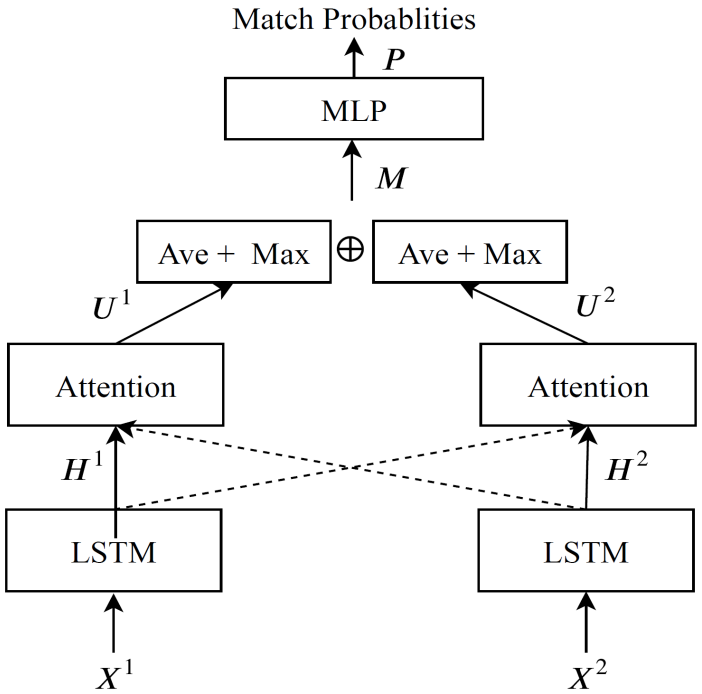
Same to the Siamese network

$$emb(W^1) = [emb(w_1^1); emb(w_2^1); \dots; emb(w_{n_1}^1)]$$
$$emb(W^2) = [emb(w_1^2); emb(w_2^2); \dots; emb(w_{n_2}^2)]$$

Text Matching

WestlakeNLP

- Attention matching network:



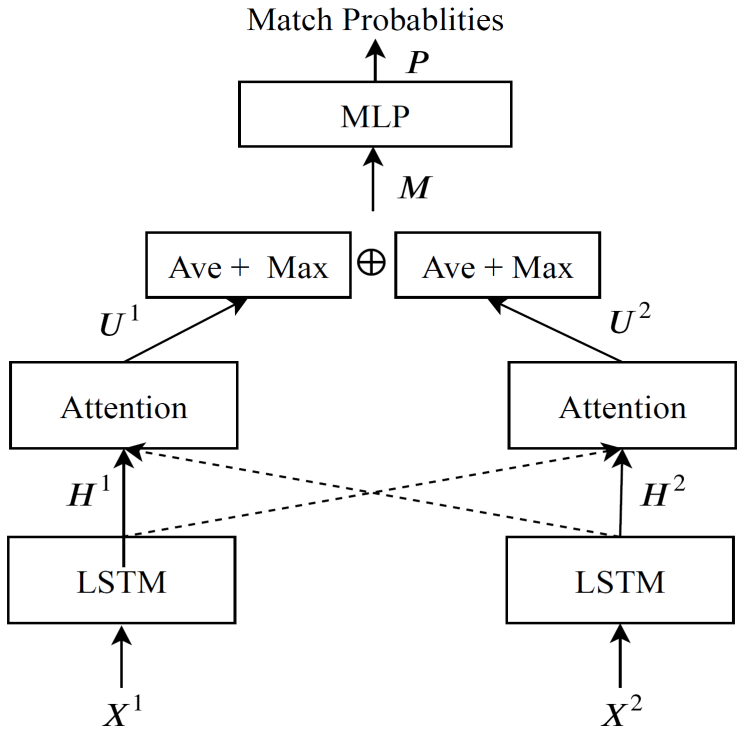
$$\mathbf{H}^1 = BiLSTM(emb(W^1))$$
$$\mathbf{H}^2 = BiLSTM(emb(W^2))$$

$$emb(W^1) = [emb(w_1^1); emb(w_2^1); \dots; emb(w_{n_1}^1)]$$
$$emb(W^2) = [emb(w_1^2); emb(w_2^2); \dots; emb(w_{n_2}^2)]$$

Text Matching

WestlakeNLP

- Attention matching network:



$$s_{ij} = \mathbf{V}^T tanh(\mathbf{W}^1\mathbf{h}_i^1 + \mathbf{W}^2\mathbf{h}_j^2 + \mathbf{b})$$

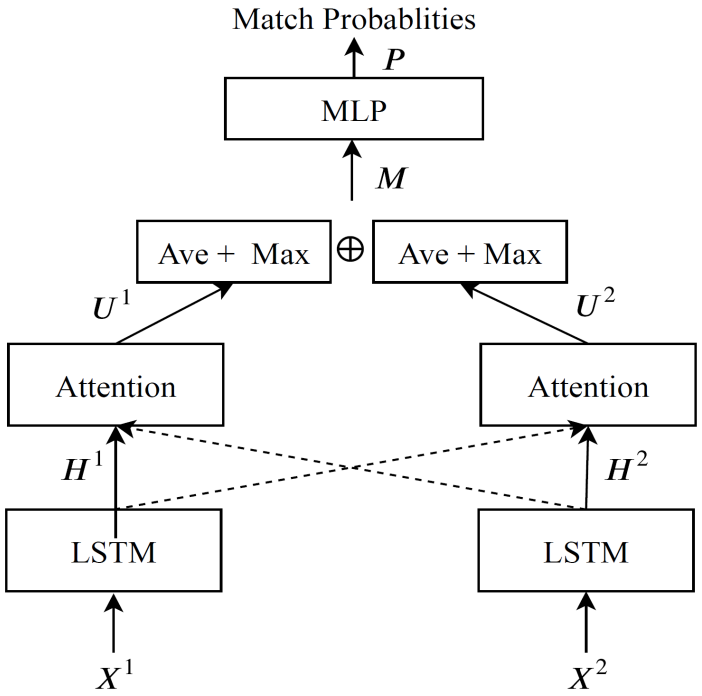
$$\alpha_{ij} = \frac{\exp(s_{ij})}{\sum_{k=1}^{n_2} \exp(s_{ik})}$$

$$\hat{\mathbf{h}}_i^1 = \sum_{j=1}^{n_2} \alpha_{ij}$$

Text Matching

WestlakeNLP

- Attention matching network:



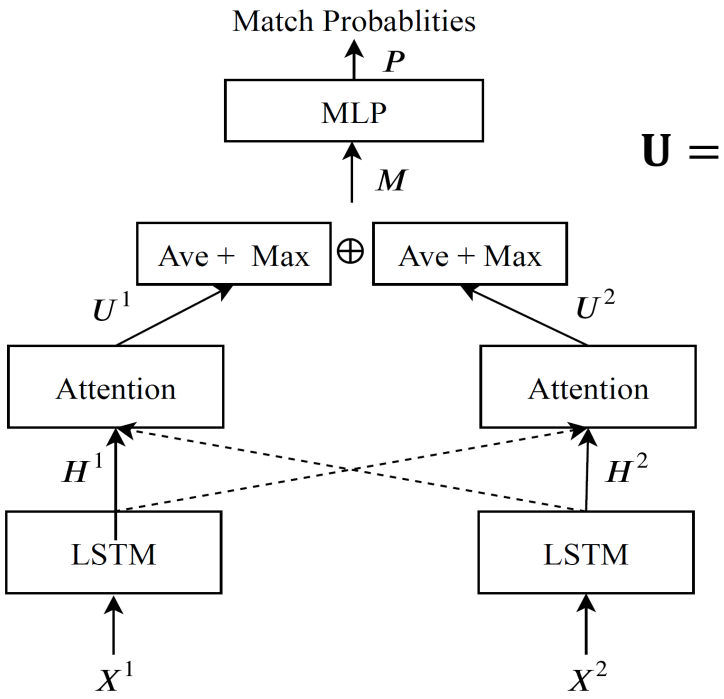
Additive attention
Denoted:

$$\widehat{\mathbf{H}}^1 = sent\_att(\mathbf{H}^1, \mathbf{H}^2)$$
$$\widehat{\mathbf{H}}^2 = sent\_att(\mathbf{H}^2, \mathbf{H}^1)$$

Text Matching

WestlakeNLP

- Attention matching network:



Concatenate the original vectors and the matching vectors:

$$\mathbf{U} = avg(\mathbf{Q}^1) \oplus max(\mathbf{Q}^1) \oplus avg(\mathbf{Q}^2) \oplus max(\mathbf{Q}^2)$$

$$\mathbf{Q}^1 = [\mathbf{h}_1^1 \oplus \hat{\mathbf{h}}_1^1; \mathbf{h}_2^1 \oplus \hat{\mathbf{h}}_2^1; \ldots; \mathbf{h}_{n_1}^1 \oplus \hat{\mathbf{h}}_{n_1}^1]$$

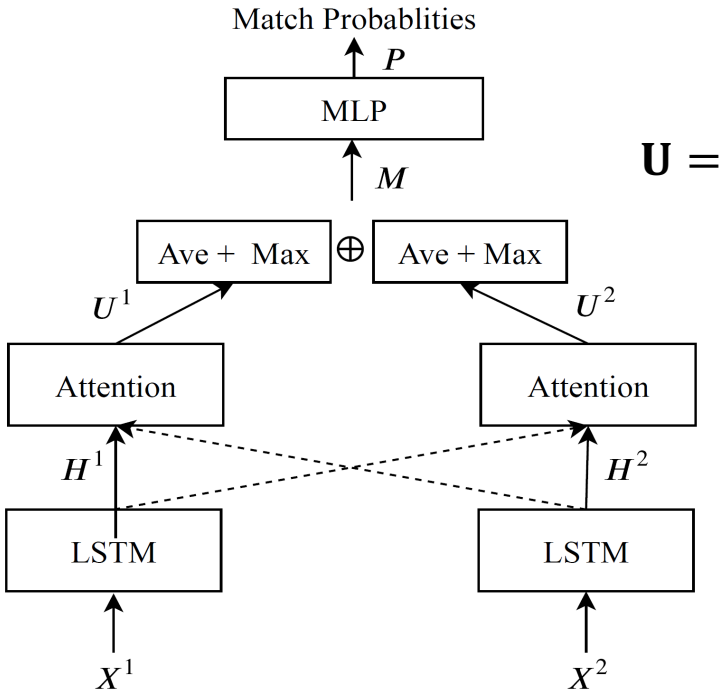
$$\mathbf{Q}^2 = [\mathbf{h}_1^2 \oplus \hat{\mathbf{h}}_1^2; \mathbf{h}_2^2 \oplus \hat{\mathbf{h}}_2^2; \ldots; \mathbf{h}_{n_2}^2 \oplus \hat{\mathbf{h}}_{n_2}^2]$$

ave: Bitwise average pooling
max: Max pooling
⊕: Cancatenate

Text Matching

WestlakeNLP

- Attention matching network:



Final multilayer perceptron for matching probability:

$$P(match(W^1, W^2)) = \sigma(MLP(\mathbf{U}))$$

$$\mathbf{U} = avg(\mathbf{Q}^1) \oplus max(\mathbf{Q}^1) \oplus avg(\mathbf{Q}^2) \oplus max(\mathbf{Q}^2)$$

$$\mathbf{Q}^1 = [\mathbf{h}_1^1 \oplus \hat{\mathbf{h}}_1^1; \mathbf{h}_2^1 \oplus \hat{\mathbf{h}}_2^1; \dots; \mathbf{h}_{n_1}^1 \oplus \hat{\mathbf{h}}_{n_1}^1]$$

$$\mathbf{Q}^2 = [\mathbf{h}_1^2 \oplus \hat{\mathbf{h}}_1^2; \mathbf{h}_2^2 \oplus \hat{\mathbf{h}}_2^2; \dots; \mathbf{h}_{n_2}^2 \oplus \hat{\mathbf{h}}_{n_2}^2]$$

ave: Bitwise average pooling
max: Max pooling
⊕: Cancatenate

Text Matching

WestlakeNLP

- Bidirectional attention matching network:
  - Similarly, Bi-direction attention (*co-attention*) can be used to extract the mutual information between H^1 and H^2.

$$\widetilde{\mathbf{H}}^1 = Dup(\mathbf{H}^1, n_2)$$
$$\widetilde{\mathbf{H}}^2 = Dup(\mathbf{H}^2, n_1)$$
$$\mathbf{S} = avg_1(\mathbf{V}(\widetilde{\mathbf{H}}^1 \oplus (\widetilde{\mathbf{H}}^2)T_{2,3} \oplus (\widetilde{\mathbf{H}}^1 \otimes (\widetilde{\mathbf{H}}^2)T_{2,3})))$$

  - Attention weight matrix:

$$\alpha_1 = softmax_2(\mathbf{S})$$
$$\alpha_2 = softmax_2(\mathbf{S^T})$$

  - The bi-directional attention output $\boldsymbol{U}$

$$\mathbf{U} = [\mathbf{H}^1 \oplus \mathbf{H}^2 \oplus [\mathbf{H}^1 \otimes \boldsymbol{\alpha}_1] \oplus [\mathbf{H}^2 \otimes \boldsymbol{\alpha}_2]]$$

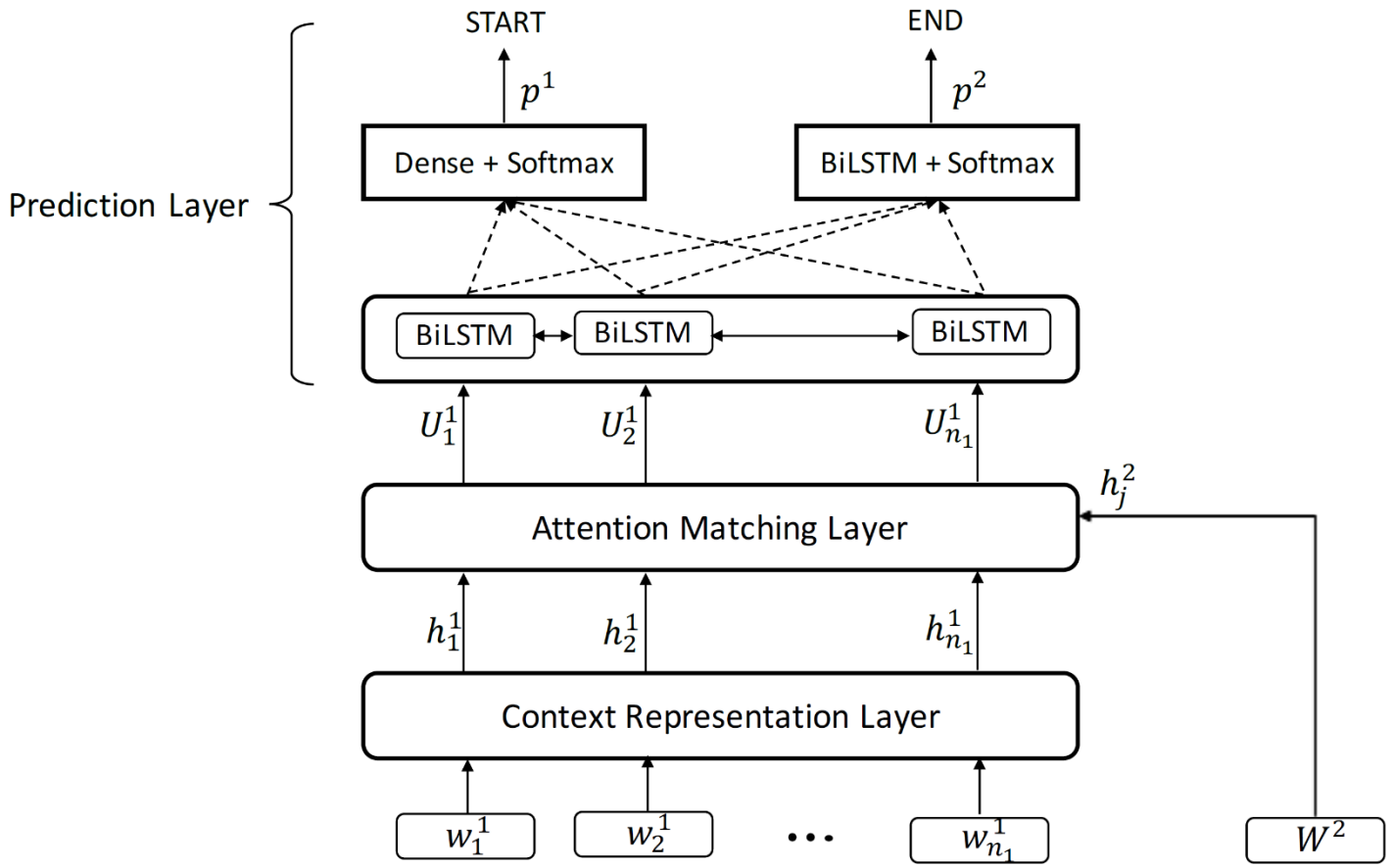
Contents

WestlakeNLP

Searching for a Match

WestlakeNLP

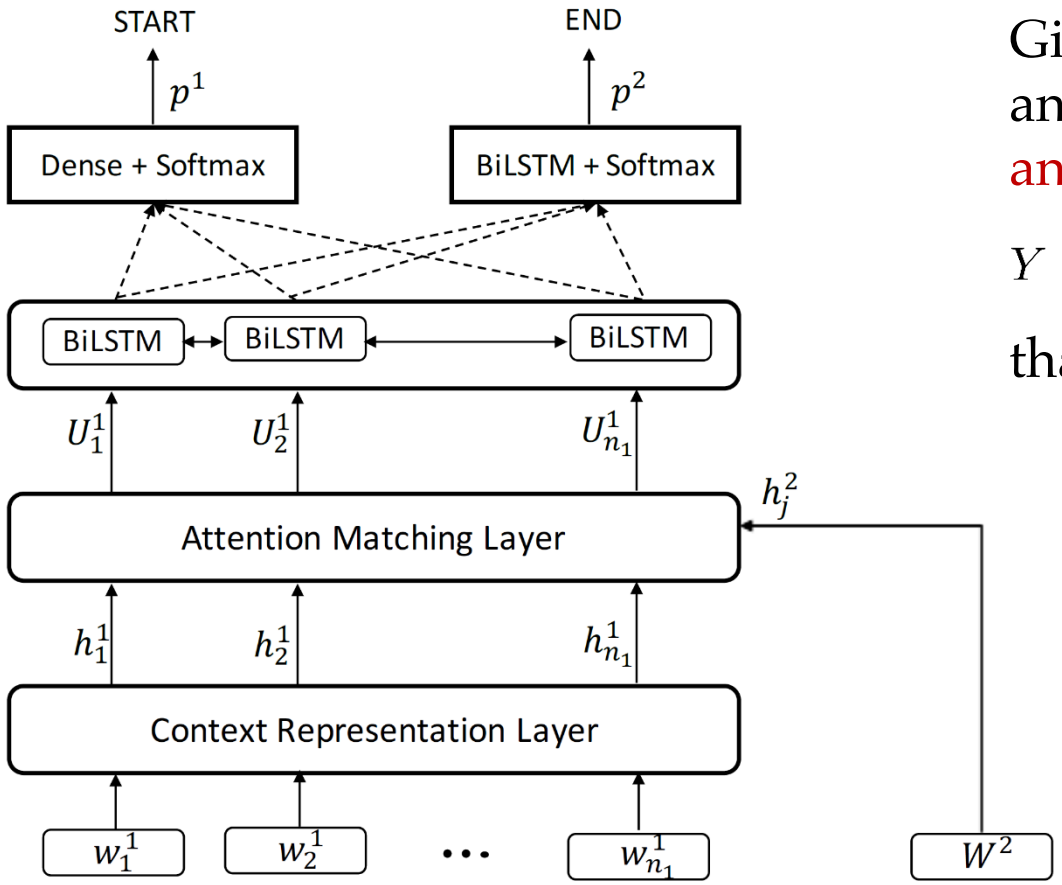
- Network structure searching for a match:



Searching for a Match

WestlakeNLP

- Network structure searching for a match:



Given a context sequence W^1 and a query W^2, the task is to find an answer span

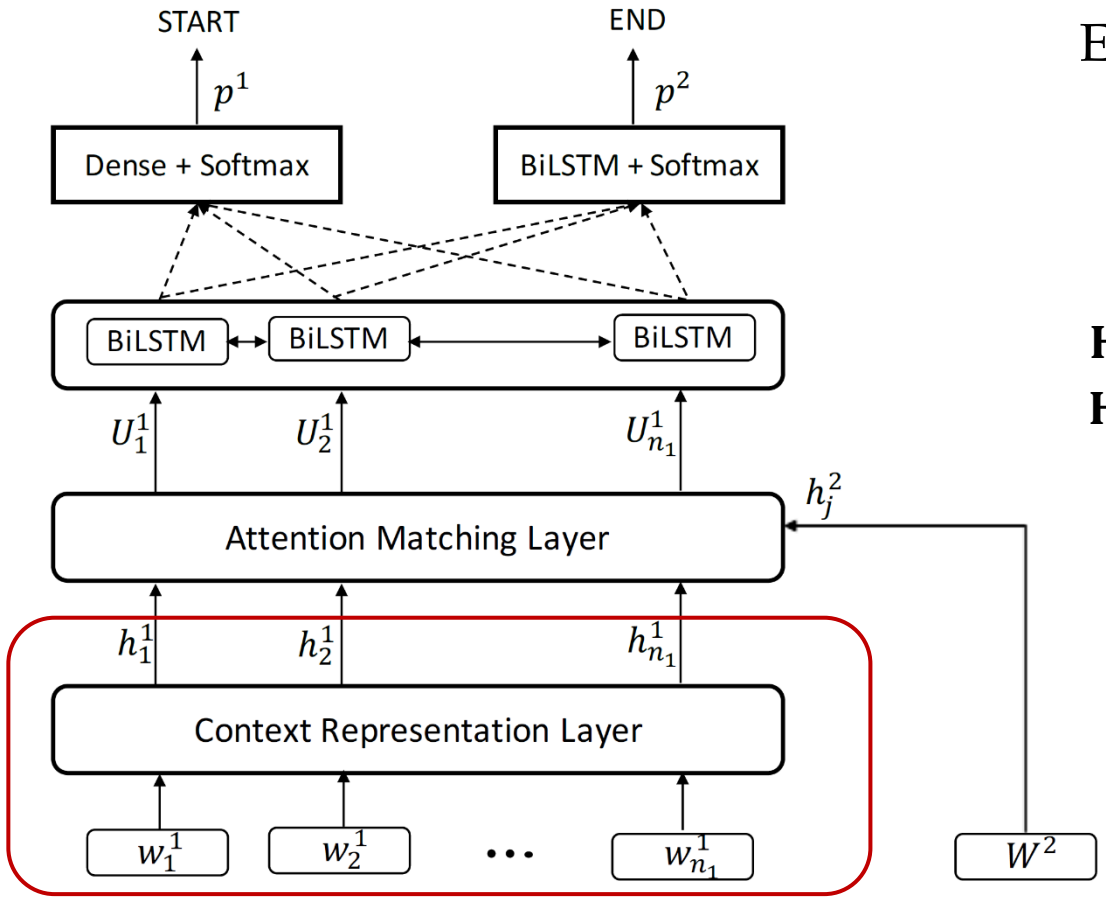
$$Y = (W^1)_b^e = w_b^1, ..., w_e^1$$

that matches W^2.

Searching for a Match

WestlakeNLP

- Network structure searching for a match:



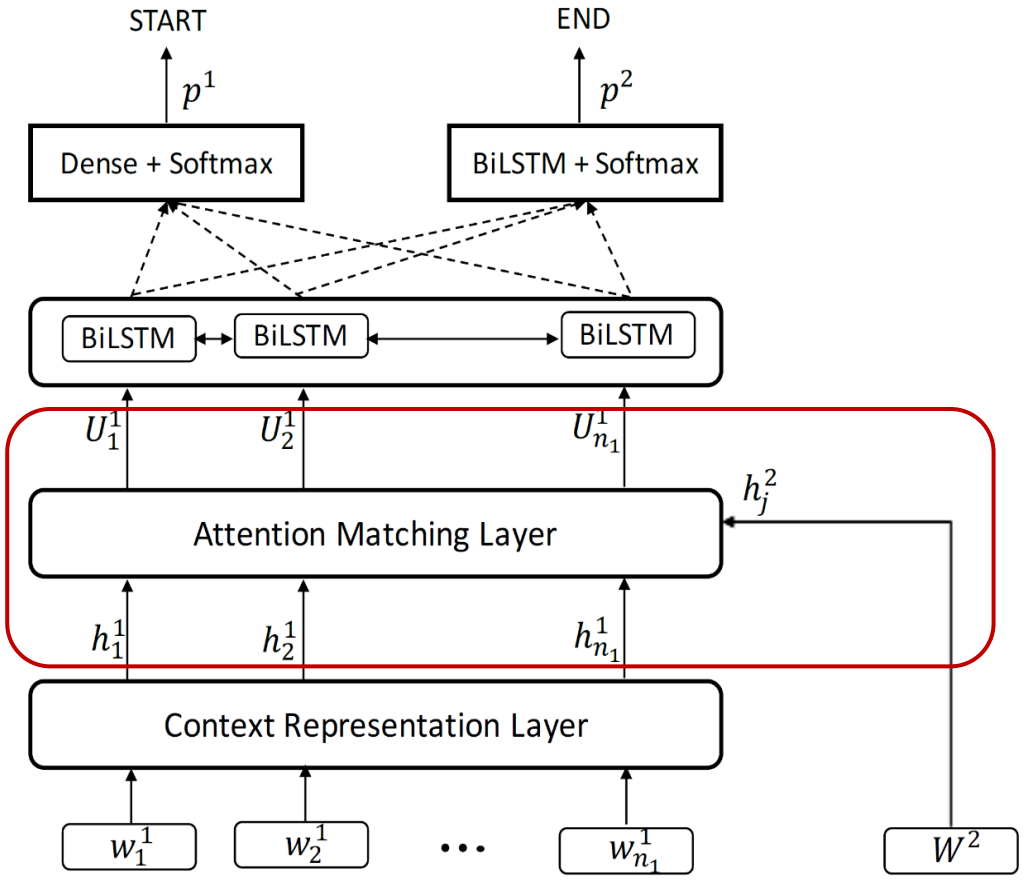
Encoding with Bi-LSTM

$$\mathbf{H}^1 = BiLSTM(emb(W^1))$$
$$\mathbf{H}^2 = BiLSTM(emb(W^2))$$

Searching for a Match

WestlakeNLP

- Network structure searching for a match:



Additive attention for matching:

$$s_{ij} = \mathbf{V}^T tanh(\mathbf{W}^1\mathbf{h}_i^1 + \mathbf{W}^2\mathbf{h}_j^2 + \mathbf{b})$$

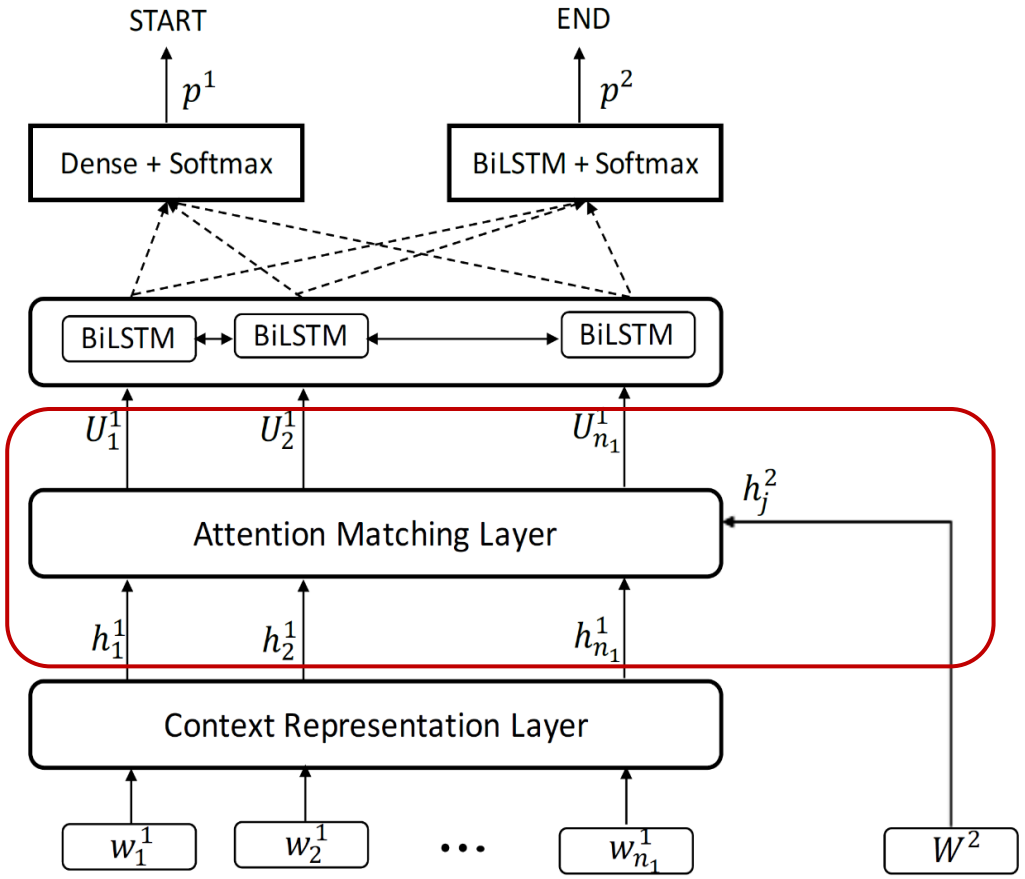
$$\boldsymbol{\alpha}_{ij} = \frac{\exp(s_{ij})}{\sum_{k=1}^{n_2} \exp(s_{ik})}$$

$$\hat{\mathbf{h}}_i^1 = \sum_{j=1}^{n_2} \alpha_{ij}\, \mathbf{h}_j^2$$

Searching for a Match

WestlakeNLP

- Network structure searching for a match:



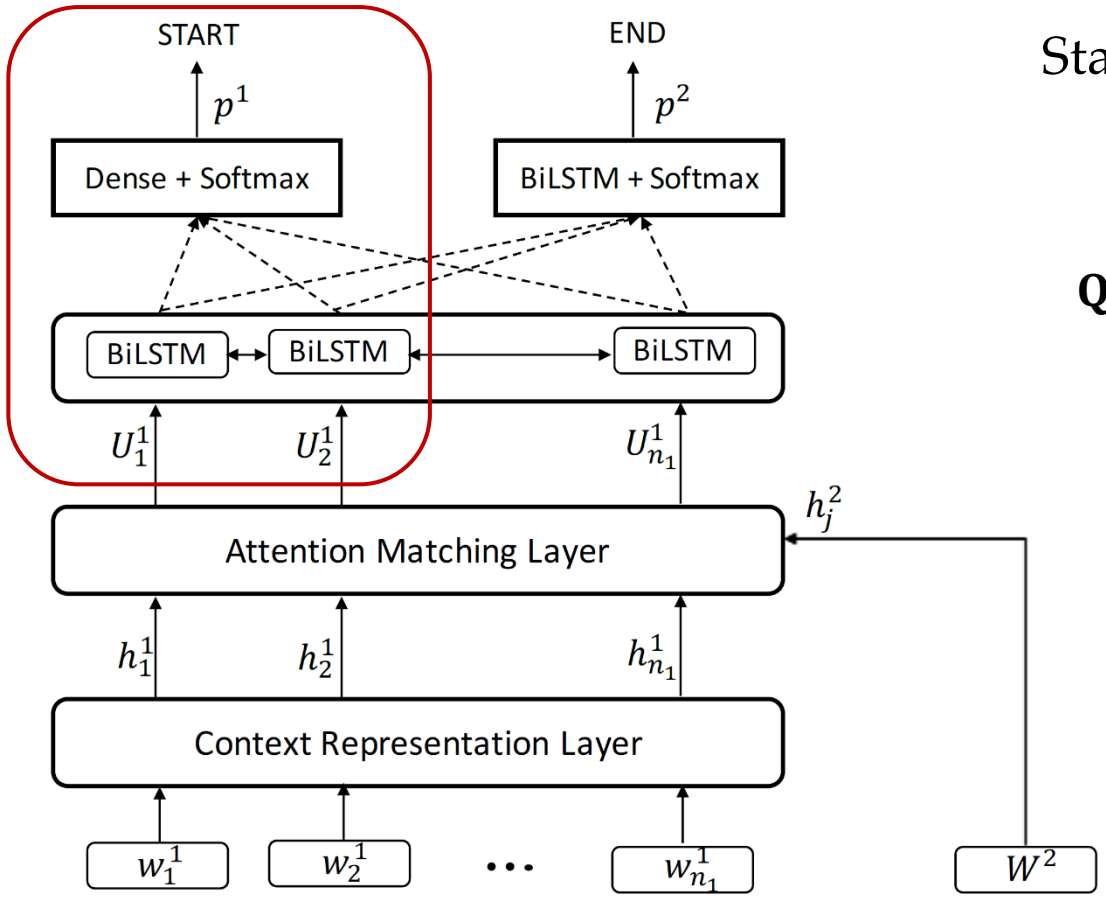
Final representation:

$$\mathbf{U}^1 = [\mathbf{h}_1^1 \oplus \hat{\mathbf{h}}_1^1; \mathbf{h}_2^1 \oplus \hat{\mathbf{h}}_2^1; \ldots; \mathbf{h}_{n_1}^1 \oplus \hat{\mathbf{h}}_{n_1}^1]$$

Searching for a Match

WestlakeNLP

- Network structure searching for a match:



Start Predicting:

$$\mathbf{M}^b = BiLSTM(\mathbf{U}^1)$$

$$\mathbf{Q}^b = [\mathbf{u}_1^1 \oplus \mathbf{m}_1^b; \mathbf{u}_2^1 \oplus \mathbf{m}_2^b; \ldots; \mathbf{u}_{n_1}^1 \oplus \mathbf{m}_{n_1}^b]$$

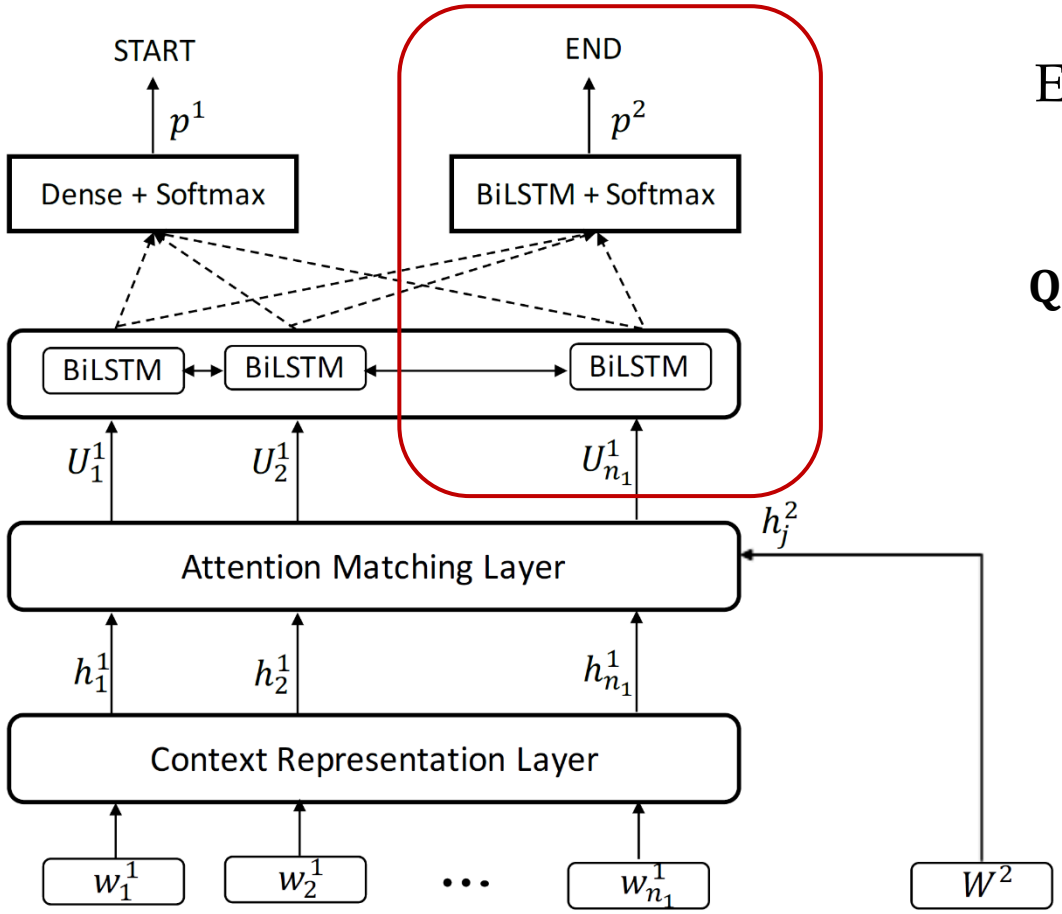
$$\mathbf{s}^b[i] = (\mathbf{v}^b)^T \mathbf{Q}_i^b, i \in [1, \ldots, n_1]$$

$$\mathbf{p}^b = softmax(\mathbf{s}^b)$$

Searching for a Match

WestlakeNLP

- Network structure searching for a match:



End Predicting:

$$\mathbf{Q}^e = [\mathbf{u}_1^1 \oplus \mathbf{m}_1^e; \mathbf{u}_2^1 \oplus \mathbf{m}_2^e; \ldots; \mathbf{u}_{n_1}^1 \oplus \mathbf{m}_{n_1}^e]$$

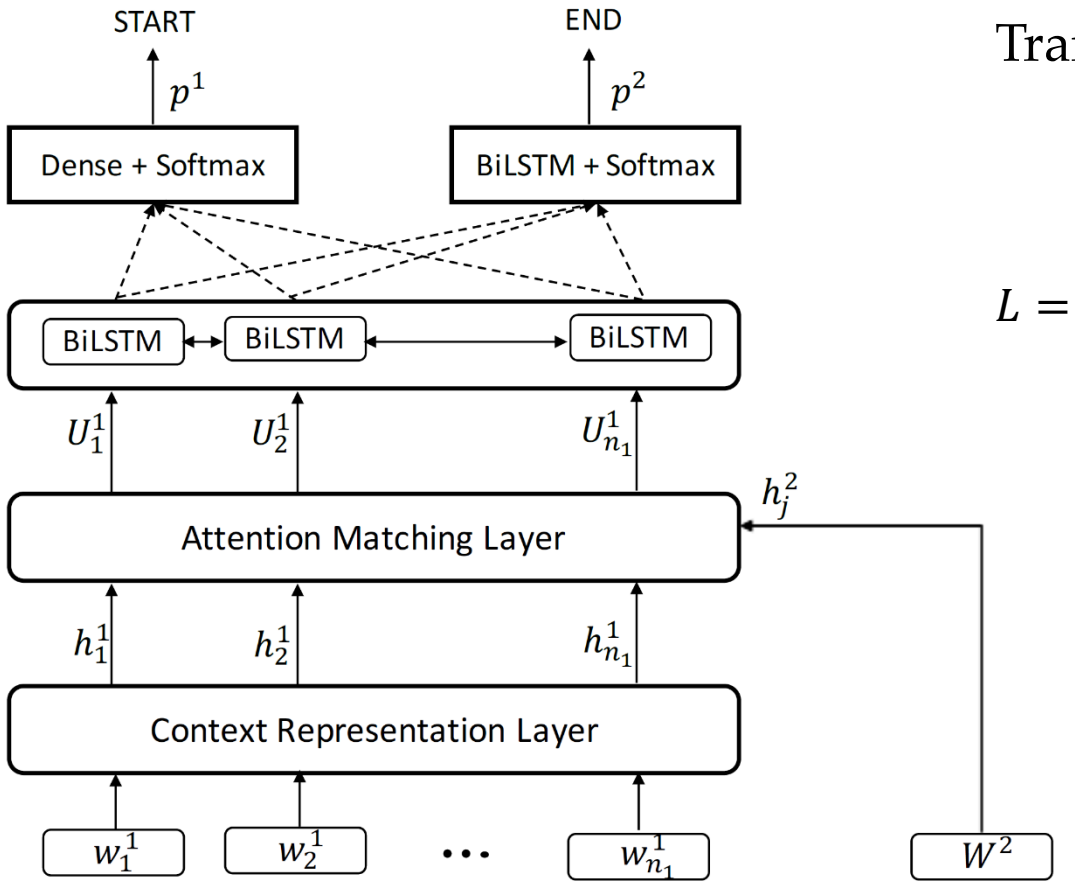
$$s^e[i] = (\mathbf{v}^e)^T \mathbf{q}_i^e, i \in [1, \ldots, n_1]$$

$$\mathbf{p}^e = softmax(\mathbf{s}^e)$$

Searching for a Match

WestlakeNLP

- Network structure searching for a match:



Training:

$$L = -\frac{1}{N}\sum_{i}^{N}(\,log(\mathbf{p}^b[b_i]) + log(\mathbf{p}^e[e_i]))$$

Contents

WestlakeNLP

Memory Network

WestlakeNLP

- Previous methods are weak in some case:
  - In many case, the correct answer to a question cannot be directly read off one evidence, but multi-stage inference is required instead.
  - For instance, suppose that the document is

Joe travelled to the office.
Joe left the milk.
Joe went to the bathroom.
Q: *Where is the milk now?*

one needs to find out the most relevant fact:

"*Joe left the milk*"

Then, one needs to consider the second relevant fact,

"*Joe travelled to the office*",

before finding a to describe the answer "*office*".

Memory Network

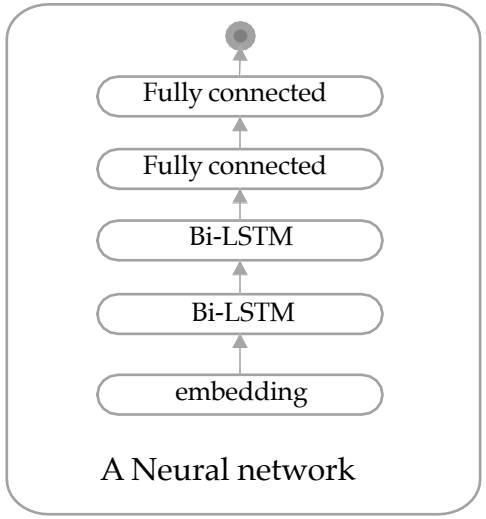
WestlakeNLP

- Why memory network:
  - Neural networks have hard times to capture long-range dependencies, even LSTMs.
  - Multi-stage inference is necessary in machine reading comprehension.
  - Memory networks try to overcome these problem using an external memory.

Memory Network

WestlakeNLP

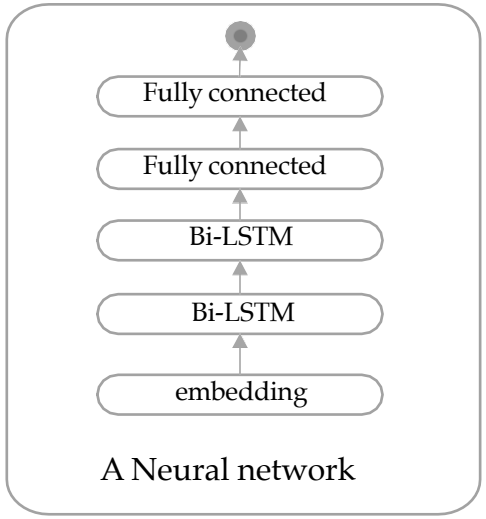
- Intuition of memory network:
  - A neural network: LSTM, SAN, …

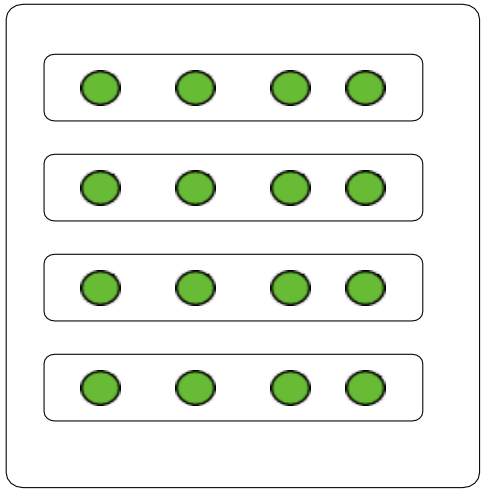


Memory Network

WestlakeNLP

- Intuition of memory network:
  - A neural network: LSTM, SAN, …
  - An external memory

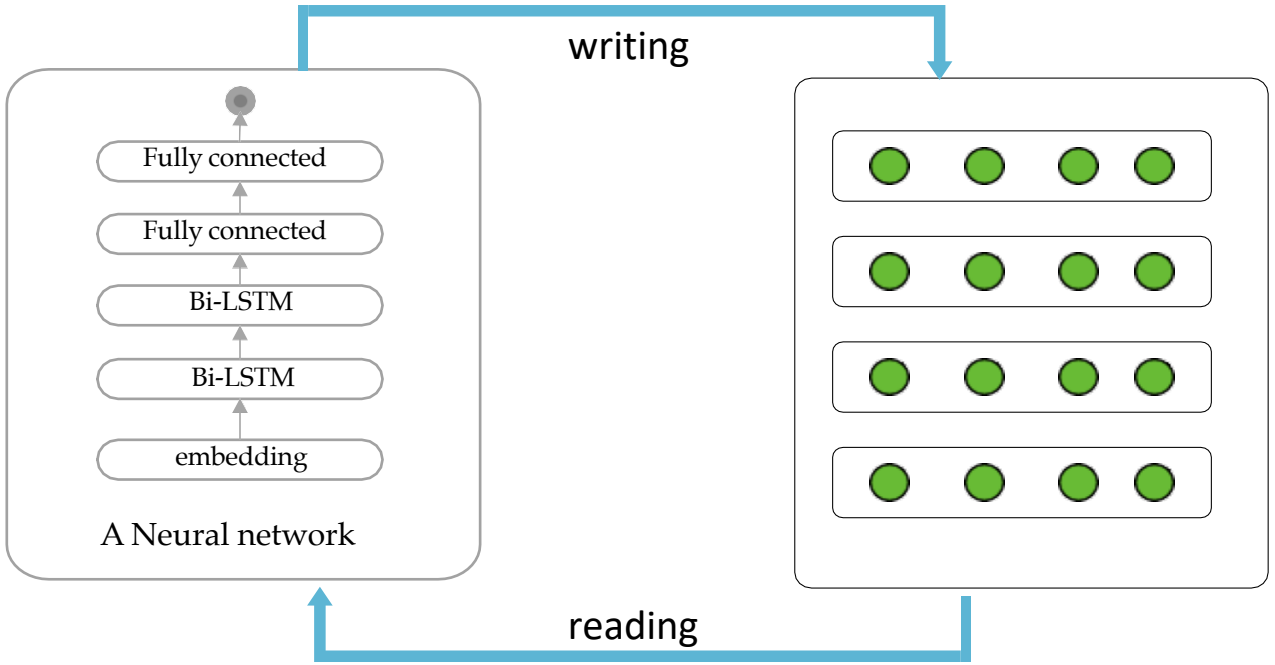




Memory Network

WestlakeNLP

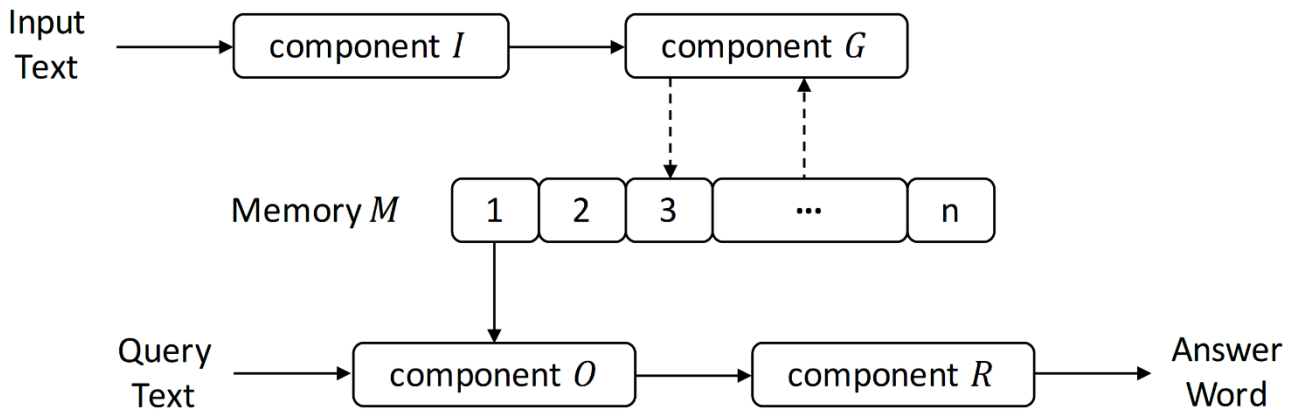
- Intuition of memory network:
  - A neural network: LSTM, SAN, …
  - An external memory where the neural network can write and read to



Memory Network

WestlakeNLP

- Memory network structure: a memory and 4 components
  - Input feature map(I): encodes the input into a hidden representations, e.g. bag of words
  - Generalization(G): stores the input into memory
  - Output feature map(O): read the most relevant memory slots
  - Response(R): given the info. read from the memory, return the output.

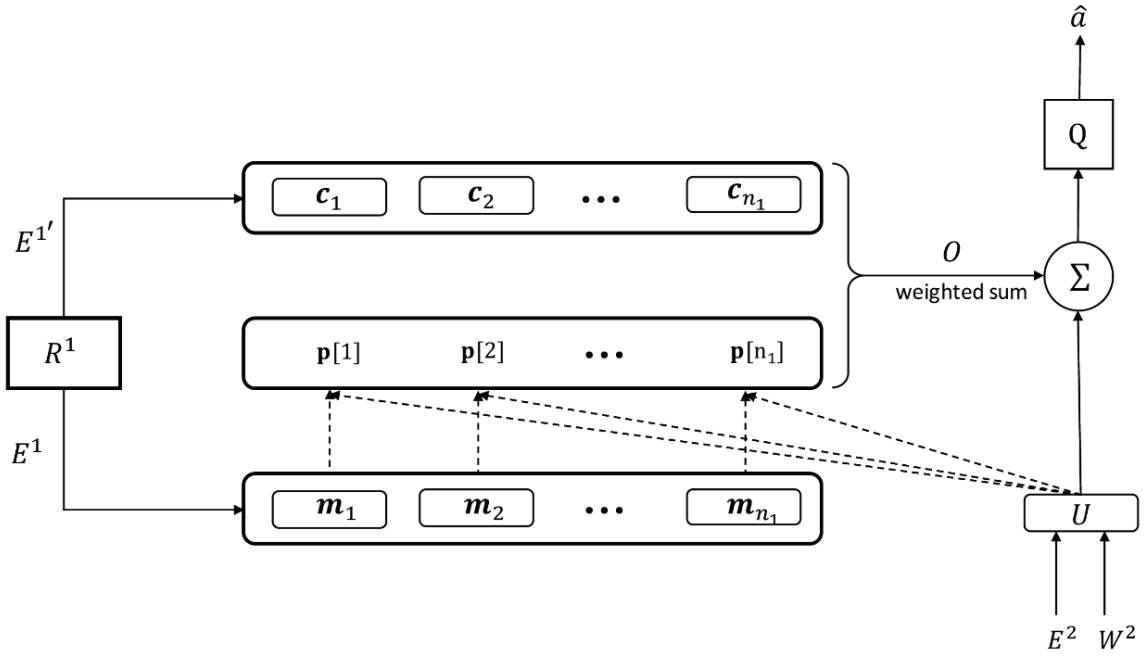


Memory Network

WestlakeNLP

- End2end memory network for QA: an example

Joe travelled to the office.
Joe left the milk.
Joe went to the bathroom.
Q: *Where is the milk now?*



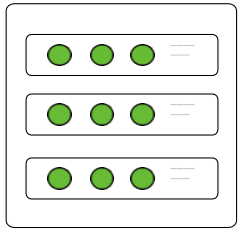
Memory Network

WestlakeNLP

- Component I: Input Representation

Raw text sentence is transformed in its vector representation e.g. BOW.

$$\mathbf{m}_i = \sum_{j=1}^{n_i} emb^{\mathbf{E}^R}\left(w_j^i\right)$$



Joe travelled to the office.
Joe left the milk.
Joe went to the bathroom.
Q: Where is the milk now?

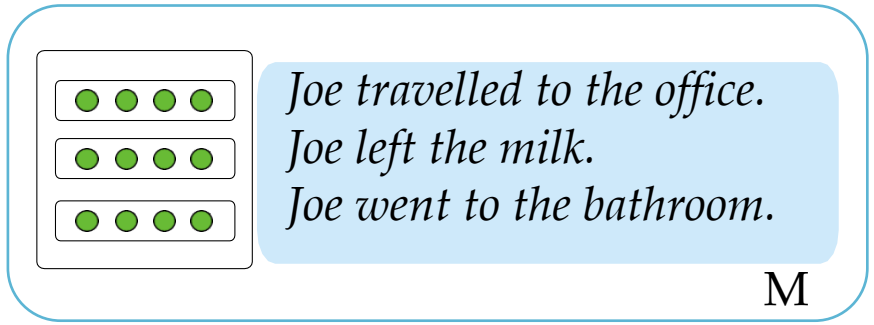
l_j is position encoding to capture the order of the words.

$$l_j^i = (1 - j/n_i) - (i/d_h)(1 - 2\,j/n_i)$$

Memory Network

WestlakeNLP

- Component G:
  - The sentences are then written to the memory sequentially, via the component G:



$$\mathbf{m}_i = \sum_{j=1}^{n_i} l_j^i \cdot emb^{\mathbf{E}^R}(w_j^i)$$

$$\mathbf{M} = [\mathbf{m}_1; \ldots; \mathbf{m}_{n_R}]$$

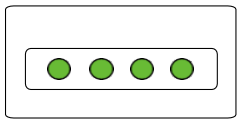
  - Notice that the memory is fixed in this approach once is written, it is not changed neither during learning nor during testing.

Memory Network

WestlakeNLP

- Query text Representation
  - Similar, query representation as follow:

$$\mathbf{u} = \sum_{i=1}^{n_Q} l_i^Q \cdot emb^{\mathbf{E}^Q}(W_i^Q)a$$



Joe travelled to the office.
Joe left the milk.
Joe went to the bathroom.
Q: *Where is the milk now?*

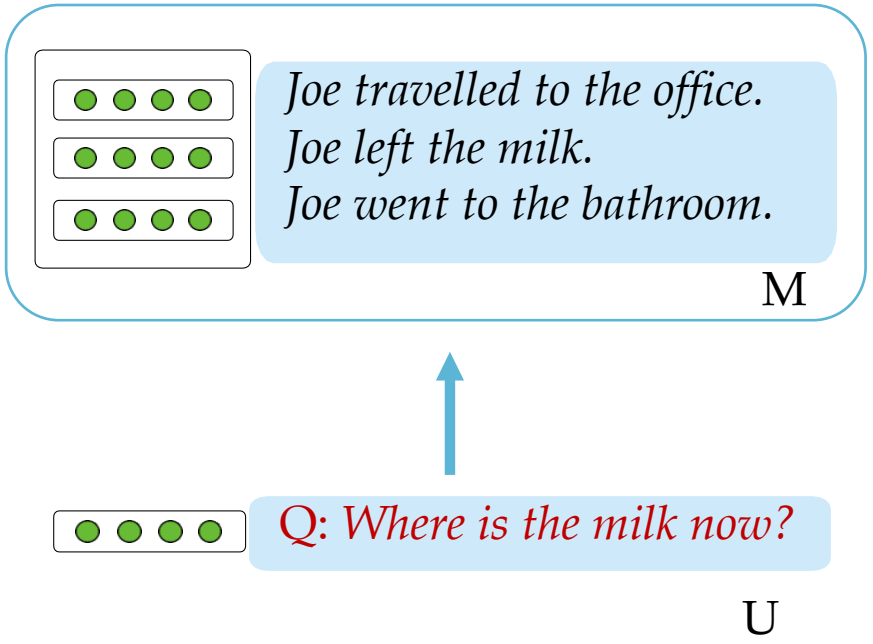
l_j is position encoding to capture the order of the words.

$$l_i^Q = (1 - i/n_Q) - (1/d_h)(1 - 2\,i/n_Q)$$

Memory Network

WestlakeNLP

- Component O: matching
  - The most relevant memory blocks are found by calculating p over memory cells m_i:



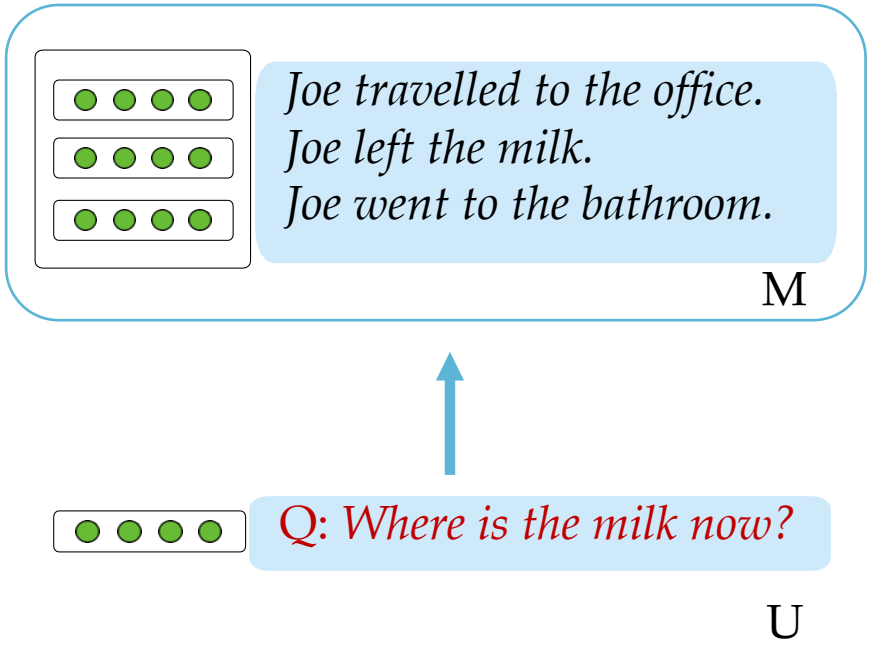
$$s_i = \mathbf{u}^T \mathbf{m}_i$$

$$\mathbf{p}[i] = \frac{\exp(s_i)}{\sum_{k=1}^{n_R} \exp(s_k)}$$

Memory Network

WestlakeNLP

- Component O: output
  - Then, we calculate a context vector o in accordance with p.



$$\mathbf{c}_i = \sum_{j=1}^{n_i} l_j^i \cdot emb^{\mathbf{E}^{R'}}(w_j^i)$$

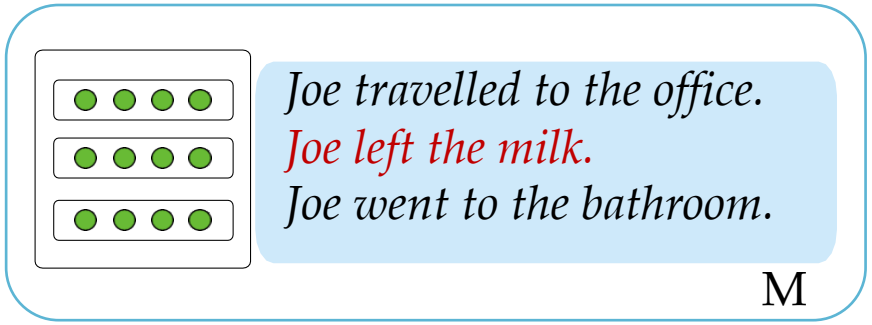
Finally, the output evidence vector:

$$\mathbf{o} = \sum_{i=1}^{n_Q} \mathbf{p}\,[i]\mathbf{c}_i$$

Memory Network

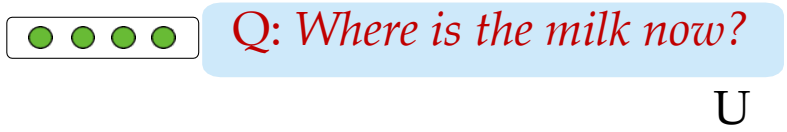
WestlakeNLP

- Component R: response with component O
  - $\boldsymbol{a}[i]$ represents the probability of the i th word from the vocabulary being the answer word with the evidence vector $\mathbf{o}$.



$$\mathbf{q} = \mathbf{W}(\mathbf{o} + \mathbf{u})$$
$$\hat{\mathbf{a}} = softmax(\mathbf{q})$$

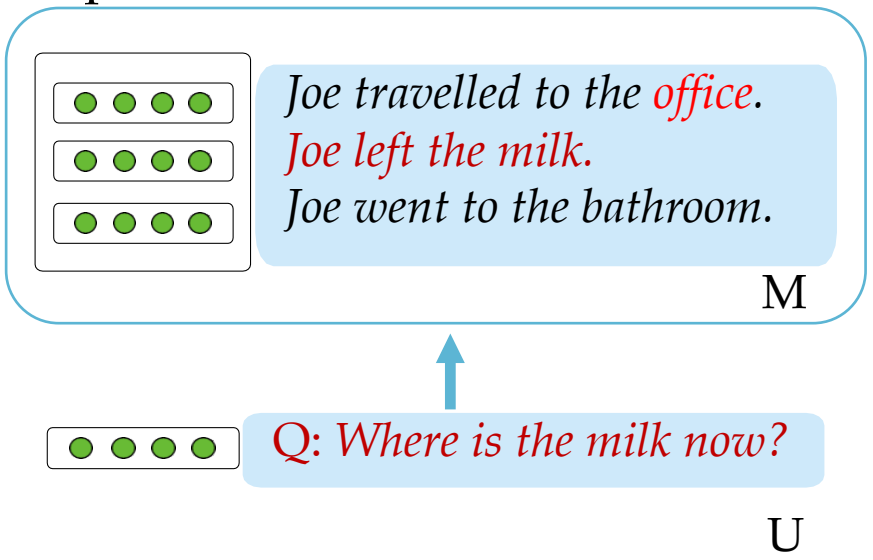
- But, we can not get the answer directly. More steps can be necessary.



Memory Network

WestlakeNLP

- Component R: response with multi-hop
  - We can leverage the existing evidence vector $\mathbf{o}$ for finding a new distribution $\boldsymbol{p}$ over memory cells, which in turn results in a new context representation of $\mathbf{o}$.



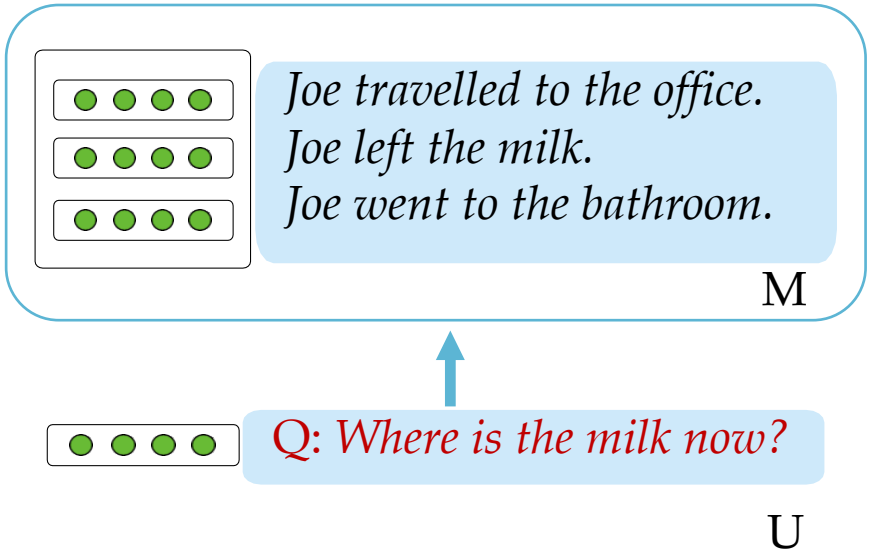
$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{o}_k$$
$$\mathbf{q}_K = \mathbf{W}(\mathbf{o}_K + \mathbf{u}_K)$$

  - This is typically called a *hop* in inference.

Memory Network

WestlakeNLP

- Training
  - Given a set of training data: $D = \{(D_i^R, W_i^Q, a_i)\}|_{i=1}^N$



Standard cross-entropy loss with the inference above:

$$L = -\sum_i^N \log\left(\hat{\mathbf{a}}_i[a_i]\right)$$