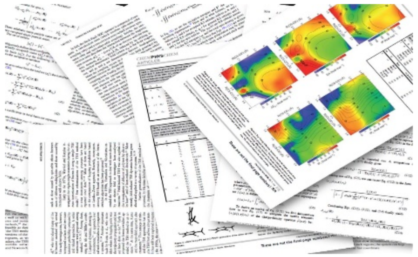
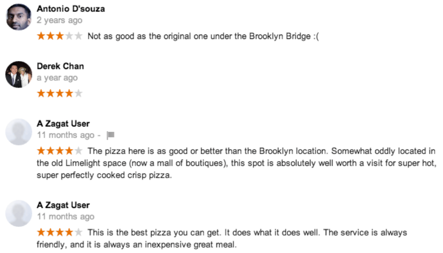
Deep Learning for Natural Language Understanding

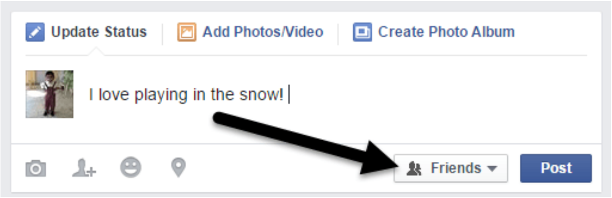
Jian Tang

tangjianpku@gmail.com

A huge amount of text data …





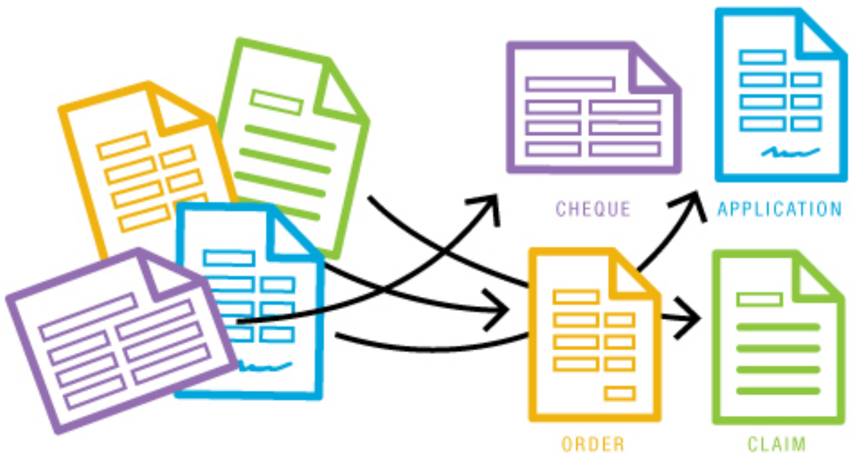


Traditional media

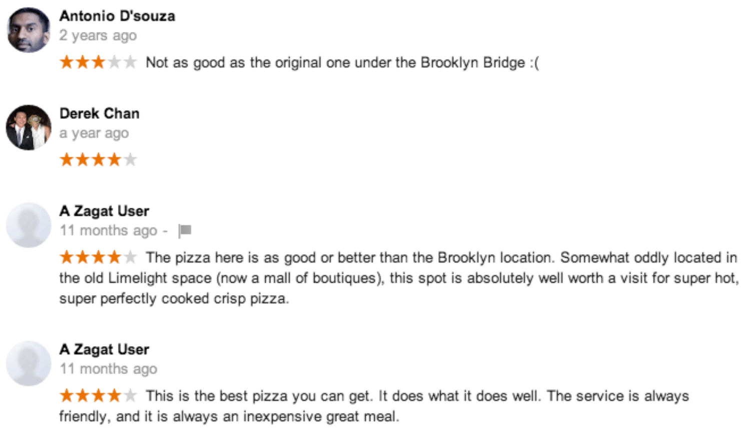
Social media

Electronic Health Records

Document (Sentence) Classification



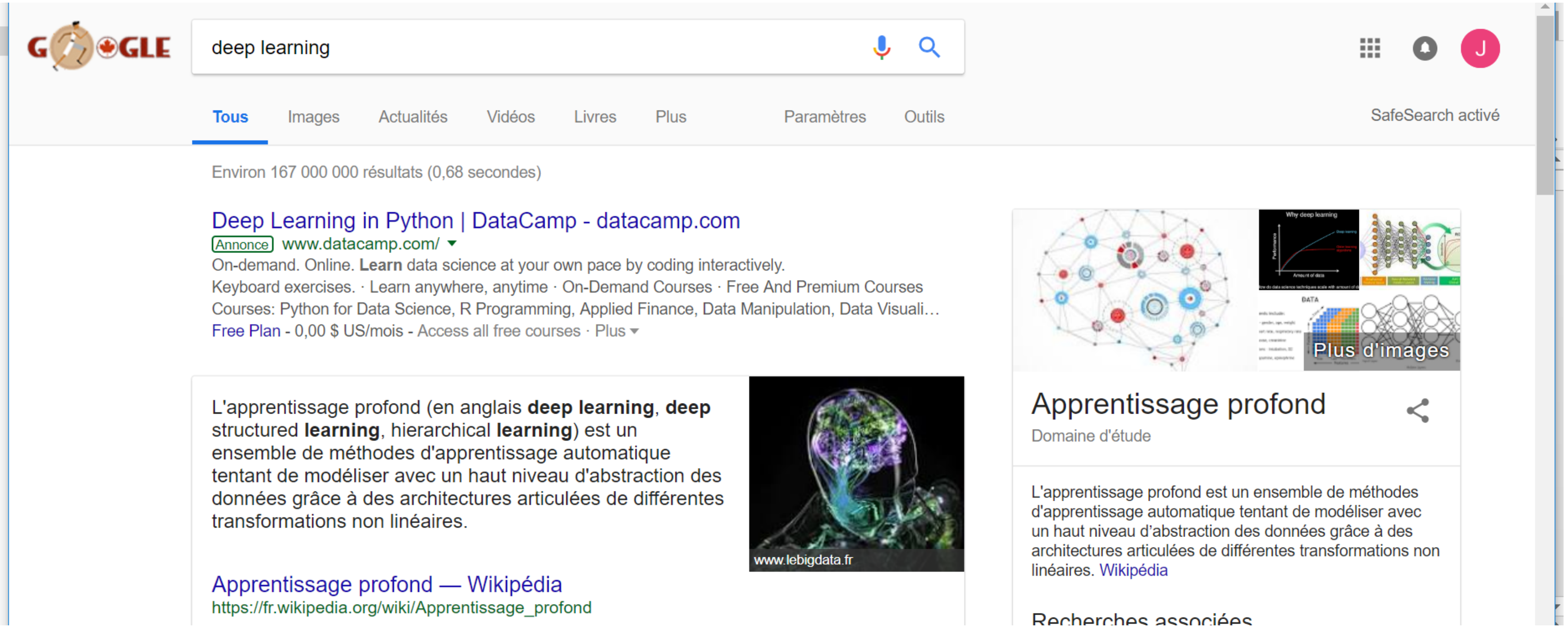
Topic classification



Sentiment classification

Document Clustering

Information Retrieval



Question Answering

Passage: Tesla later approached Morgan to ask for more funds to build a more powerful transmitter. **When asked where all the money had gone, Tesla responded by saying that he was affected by the Panic of 1901**, which he (Morgan) had caused. Morgan was shocked by the reminder of his part in the stock market crash and by Tesla's breach of contract by asking for more funds. Tesla wrote another plea to Morgan, but it was also fruitless. Morgan still owed Tesla money on the original agreement, and Tesla had been facing foreclosure even before construction of the tower began.

Question: On what did Tesla blame for the loss of the initial money?

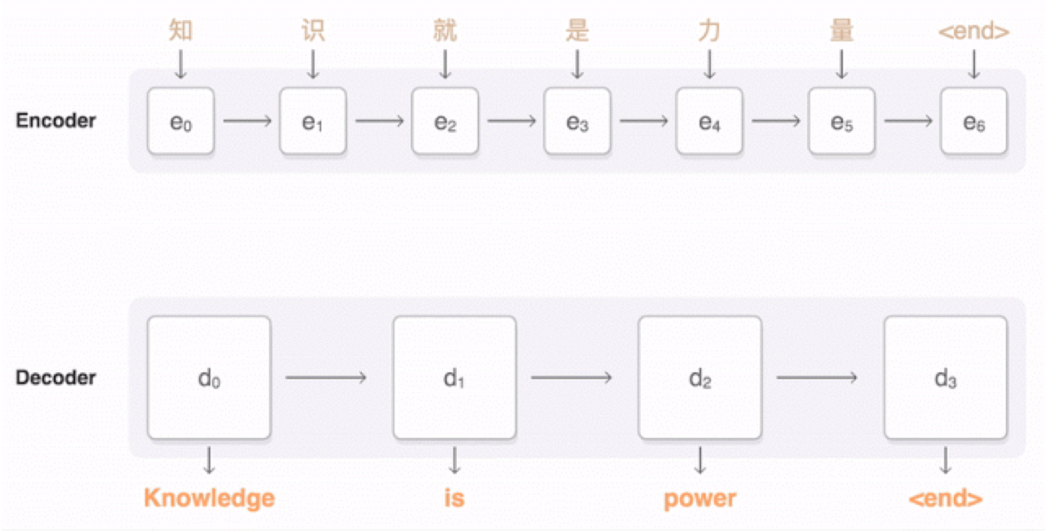
Answer: Panic of 1901

Text Summarization

Russian Defense Minister Ivanov called **Sunday** *for the creation of a joint front for combating global terrorism.*

Russia calls for joint front against terrorism.

Machine Translation



Outline

- Word Representation
  - Word2vec
- Sentence Representation
  - ParagraphVec
  - Skip-thought
  - CNN
  - LSTM & Tree-LSTM
- Machine Translation
  - Encoder-decoder
  - Attention-based encoder-decoder
  - Attention is all you need
- Question Answering
  - Memory Network
  - QANet

Classical Word Representations

- Words as atomic symbols: "*One-hot*" representation
- Documents: "*Bag-of-words*"

"network" = [0,1,0,0,0,0,0]

"networks" = [0,0,0,0,1,0,0]

AND = 0

- Ignore the *semantic relatedness* between words
- The *curse* of dimensionality
  - As large as *millions* in a large text corpus.

Neural Word Embeddings (Bengio et al. 2003)

- Represent each word with a *continuous dense* vector
  - Hundreds and thousands of dimensions
  - Words with similar meanings are represented with similar vectors
- Represent *phrases*, *sentences* and *documents* through word embedding

Distributional Hypothesis

- "You shall know a word by the company it keeps" (J.R. Firth 1957:11)
- The meaning of a word can be represented by its neighbors

A telecommunications network allows computers to exchange data

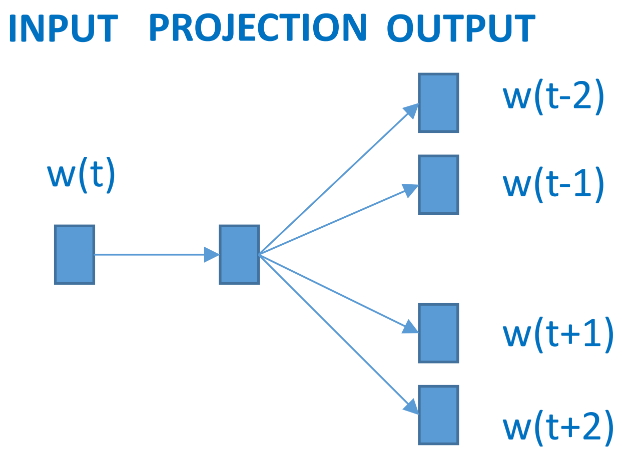
In information technology, a network is a series of points or nodes interconnected...

Represent "*network*" with the neighboring words

Word2VEC (Mikolov et al. 2013)

- **Skip-gram**: finding word representations that are useful for predicting the surrounding words in a sentence or a document

A telecommunications network allows computers to exchange data



Objective of Skip-gram

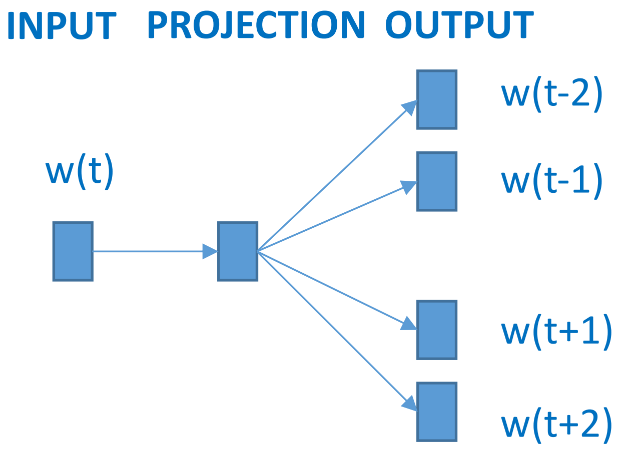
- Given a sequence of training words $w_1 w_2, \dots, w_T$, the objective of the skip-gram is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t)$$

- Where c is the size of the training context. $p(w_{t+j}|w_t)$ is defined with a softmax function

$$p(w_{t+j}|w_t) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W}\exp({v'_w}^T v_{w_I})}$$

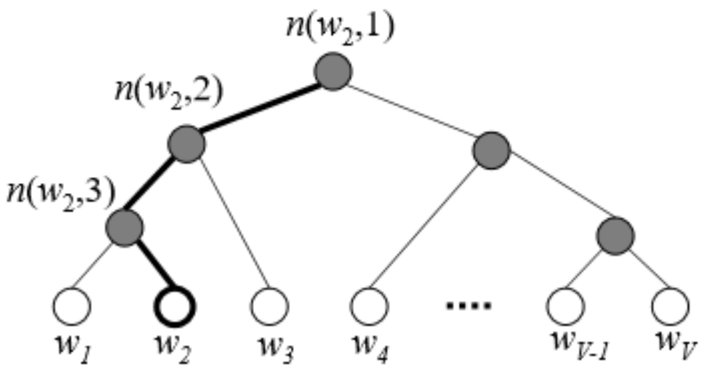
- Where v_w and v'_w are the "input" and "output" vector representations of w. W is the vocabulary size.
- Calculating $p(w_{t+j}|w_t)$ is very computational expensive



Hierarchical Softmax (Morin and Bengio 2005)

- Use a binary tree representation of the output layer with the W words as its leaves.
- Each word w can be reached with a path from the root node to the word
- n(w,j): the j-th node on the path from root to w
- L(w): the length of the path
- The hierarchical softmax defines the $p(w_O|w_I)$ as:

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w,j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^{\top} v_{w_I} \right)$$



- $\sigma(x) = 1/(1+\exp(-x))$, $[x]$ be 1 if x is true and -1 otherwise
- Computational complexity: log W

Negative Sampling (Mikolov et al. 2013)

- Modify the objective as:

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

- It aims to distinguish the target word w_O from draws from the noise distribution $P_n(w)$ using logistic regression. k is the number of negative samples for each input word (k is usually 5-20).
- $P_n(w)$ is usually set as the unigram distribution U(w) raised to the 3/4rd power, i.e.,

$$P_n(w) = U(w)^{0.75}/Z$$

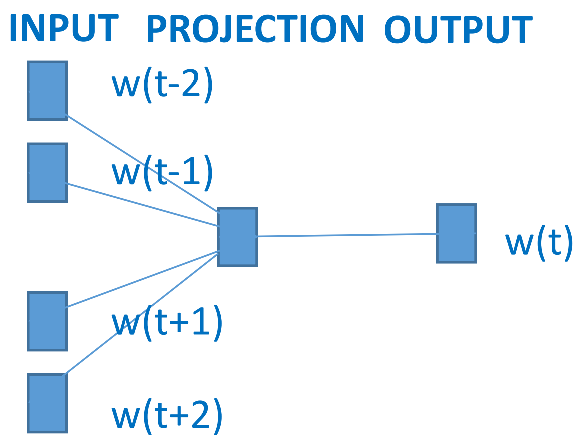
CBOW (Mikolov et al. 2013)

- Instead of using center words to predict nearby words, using nearby words to predict the center words
- Calculating the context embedding

$$v_c = \frac{1}{2c} \sum_{-c \le j \le c, j \ne 0} v_j$$

- Predict the center word:

$$p(w_t | w_{t-c}, \dots, w_{t-1}, w_{t+1}, .., w_{t+c}) = \frac{\exp({v'_{w_t}}^T v_c)}{\sum_{w=1}^{W} \exp({v'_w}^T v_c)}$$

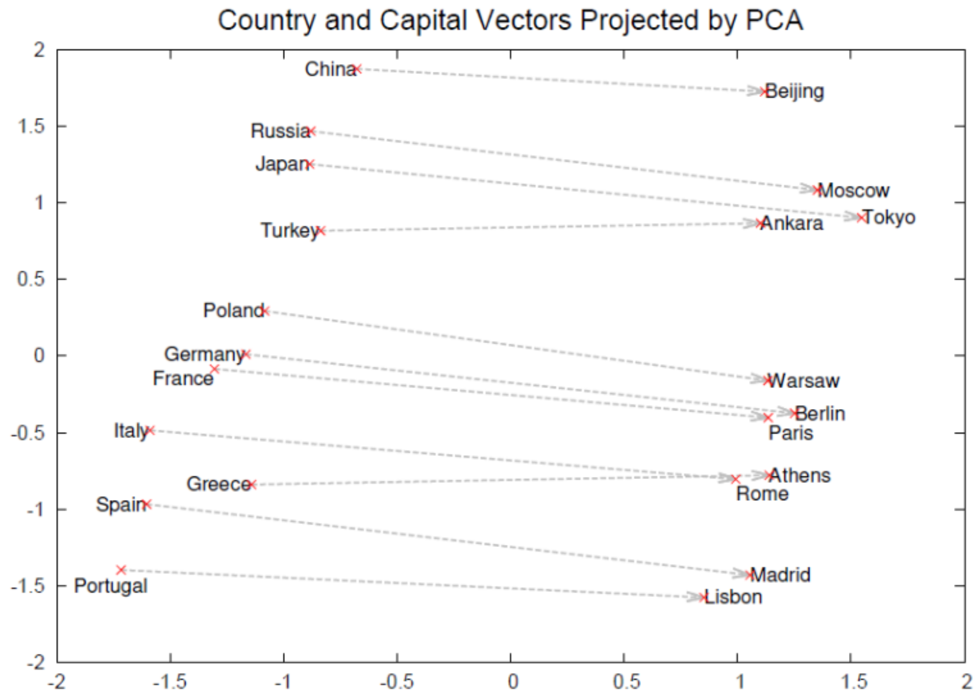


Word Analogy

- Find a word that is similar to *small* in the same sense as *biggest* is similar to *big.*
- Compute vector X=vector("biggest")-vector("big") + vector("small")
- Then search the vector space for the word closest to X measured by cosine distance, and use it as the answer.

Examples

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

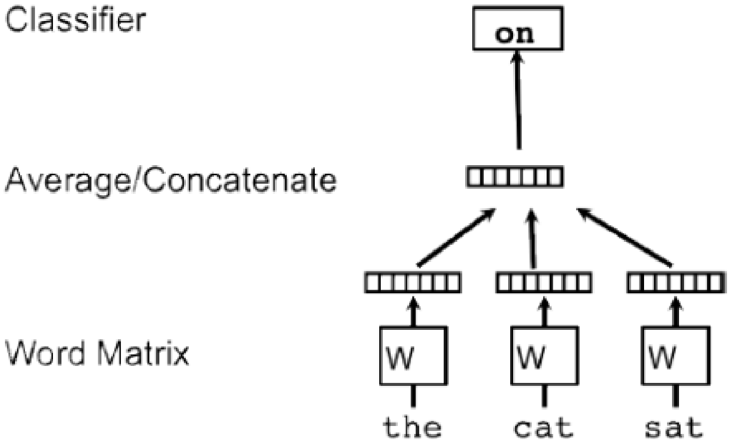


Outline

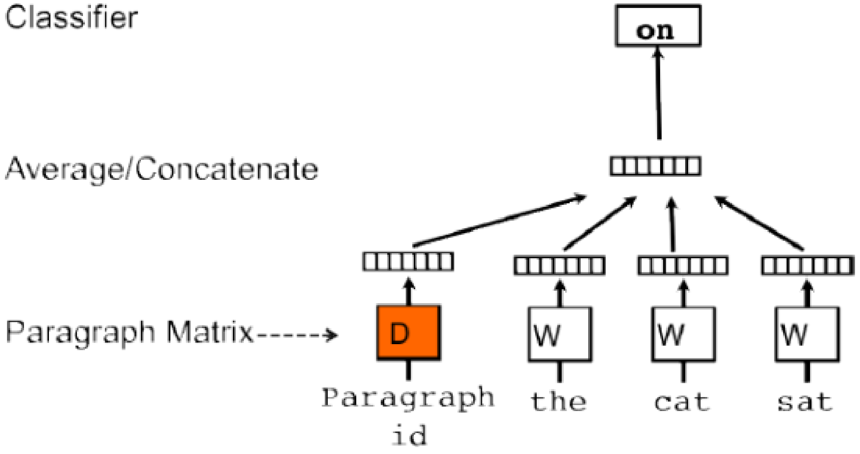
- Word Representation
  - Word2vec
- Sentence Representation
  - ParagraphVec
  - Skip-thought
  - CNN
  - LSTM & Tree-LSTM
- Machine Translation
  - Encoder-decoder
  - Attention-based encoder-decoder
  - Attention is all you need
- Question Answering
  - Memory Network
  - QANet

Unsupervised Sentence Representation: Paragraph Vector (Le et al. 2014)

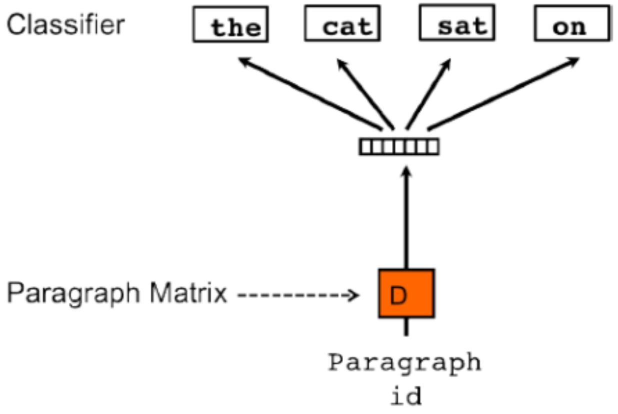
- CBOW: using context words => predict center word
- PV-DM: (context words + paragraph id) => predict center word
- PV-BOW: paragraph id => each word in the sentence



CBOW



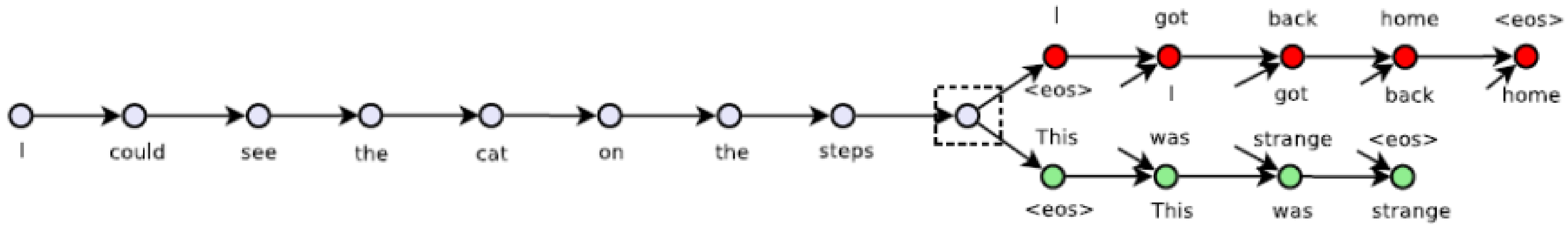
PV-DM



PV-BOW

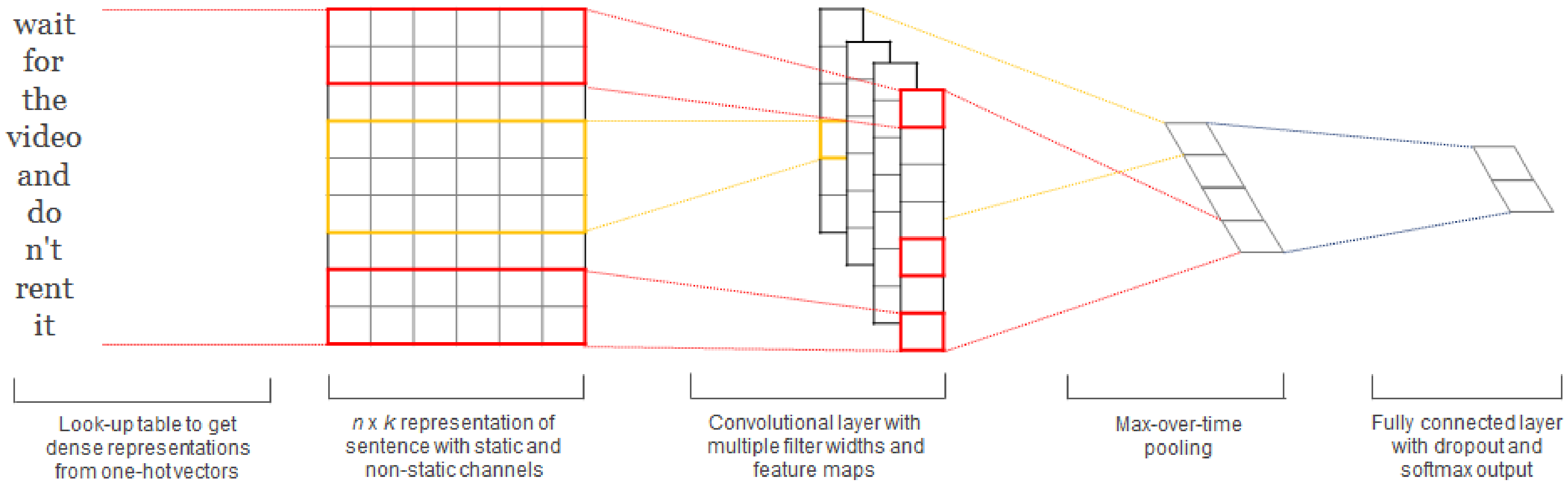
Skip-thoughts (Kiros et al. 2015)

- Given a tuple (s_{i-1}, s_i, s_{i+1}) of continuous sentences in a book, with s_i is the i-th sentence of the book. The sentence s_i is encoded with a RNN and tries to reconstruct the previous sentence s_{i-1} and next sentence s_{i+1} with another RNN



CNN for Sentence Representation (Kim 2013)

- Words are represented as word embeddings
- Multiple feature maps with different widths (modeling different n-grams)



Different Variants of CNN

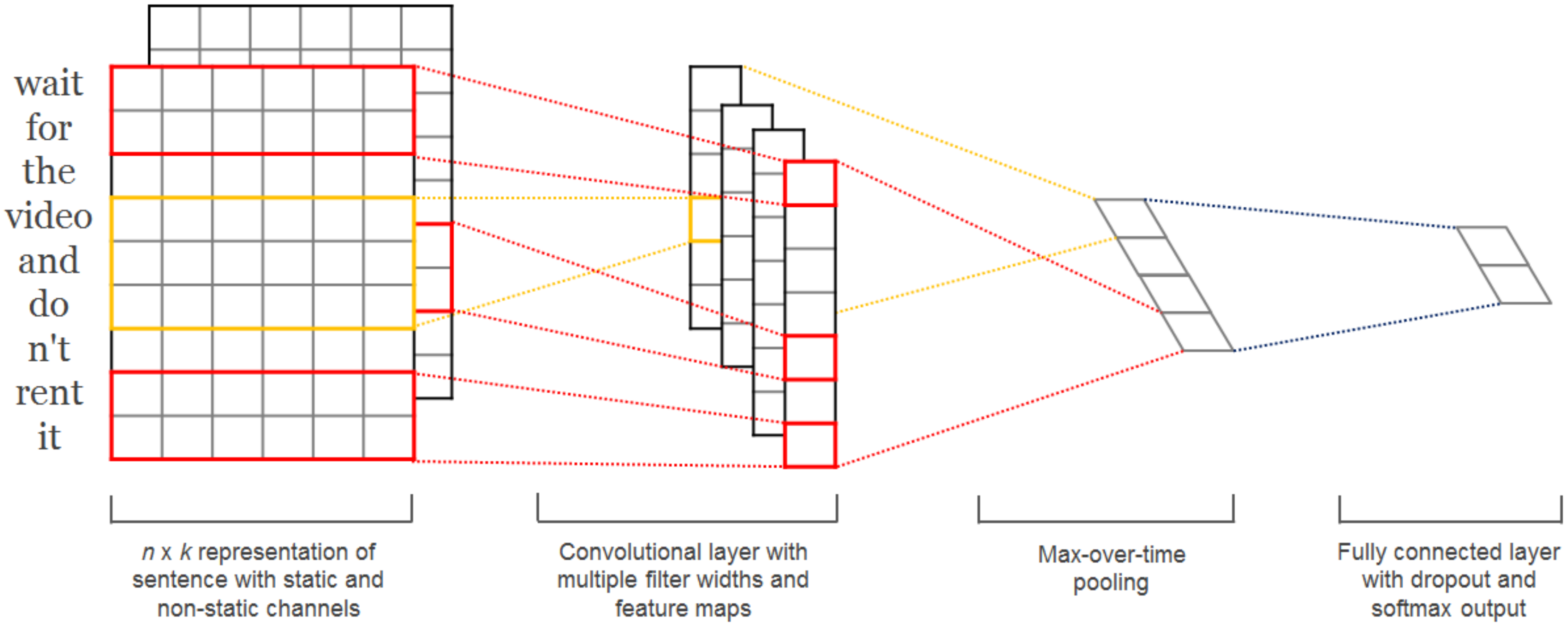
- CNN-rand: the word embeddings are randomly initialized
- CNN-static: the word embeddings are initialized by Word2VEC and fixed during training
- CNN-nonstatic: fine tuning the word embeddings by Word2VEC

Results on Sentiment Classification

Data	Prev SotA	CNN-rand	CNN-static	CNN-nonstatic
MR	79.5	76.1	81.0	81.5
SST-1	48.7	45.0	45.5	48.0
SST-2	87.8	82.7	86.8	87.2
Subj	93.6	89.6	93.0	93.4
TREC	95.0	91.2	92.8	93.6
CR	82.7	79.8	84.7	84.3
MPQA	87.2	83.4	89.6	89.5

Multi-channel CNN

- Two “channels” of embeddings
- One is allowed to change,and the other is fixed
- Both initialized with Word2VEC



Results on Sentiment Classification (Cont')

Data	Prev SotA	CNN-nonstatic	CNN-multichannel
MR	79.5	81.5	81.1
SST-1	48.7	48.0	47.4
SST-2	87.8	87.2	88.1
Subj	93.6	93.4	93.2
TREC	95.0	93.6	92.2
CR	82.7	84.3	85.0
MPQA	87.2	89.5	89.4

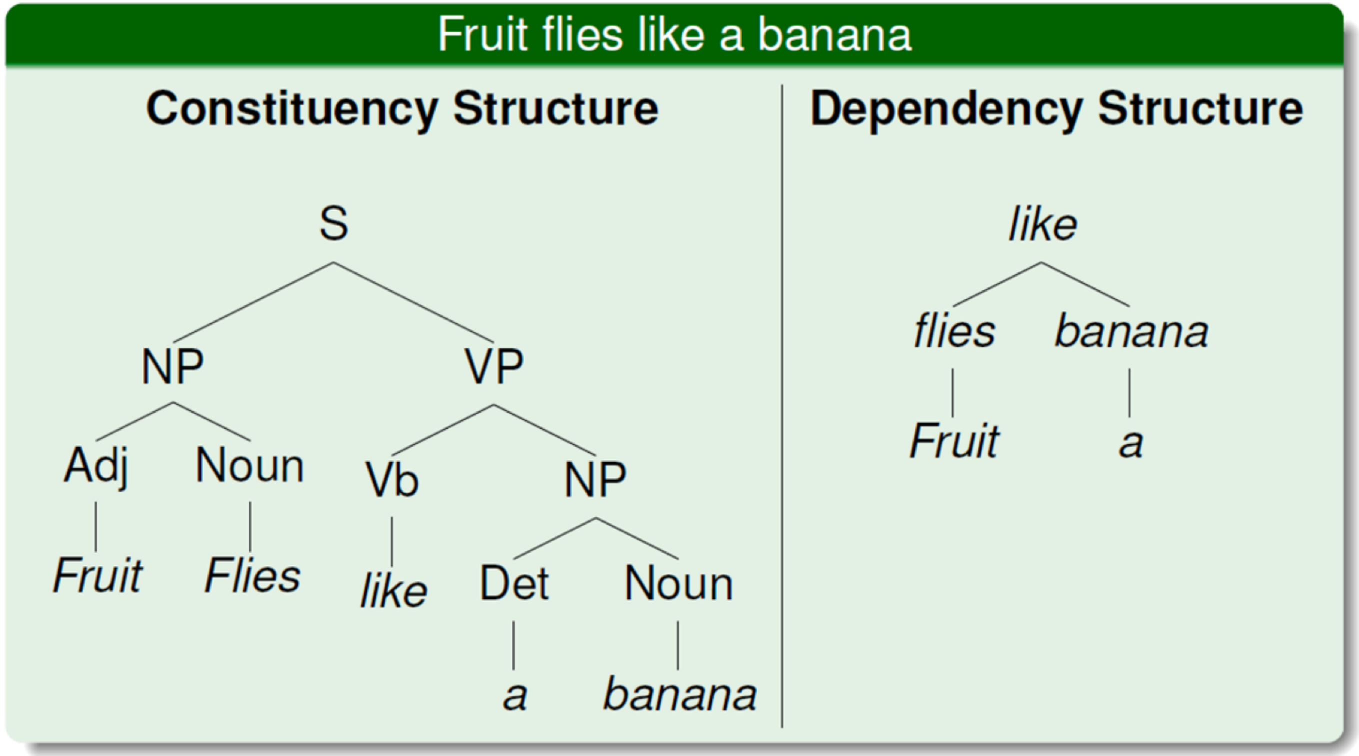
The performance are mixed

Fine-tuned Word Embeddings

	Most Similar Words for	
	Static	Non-static
bad	*good* *terrible* *horrible* *lousy*	*terrible* *horrible* *lousy* *stupid*
good	*great* *bad* *terrific* *decent*	*nice* *decent* *solid* *terrific*

Tree-Structured LSTM for Sentence Representation

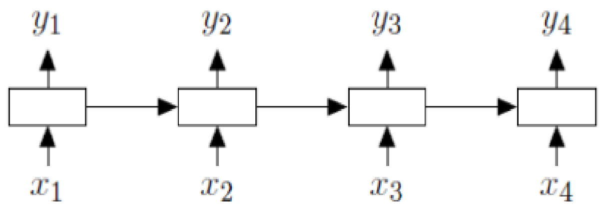
- Representing sentences as trees instead of linear chains
  - Leverage different types of dependency structures between words



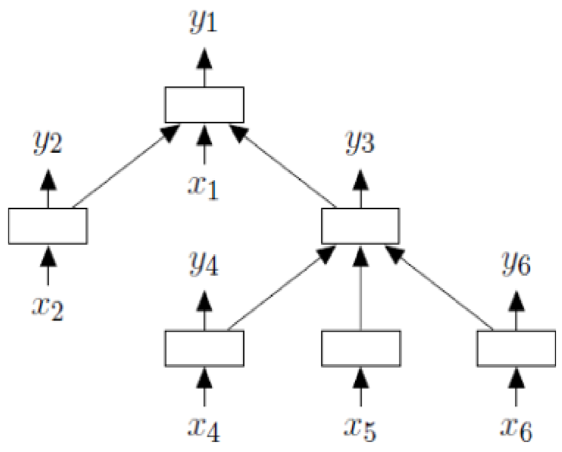
Source from internet

Tree-LSTM

- A generalization of LSTMs to tree-structured network topologies



Linear-chain LSTM



Tree-structure LSTM

Child-Sum Tree-LSTMs

Information from the children of node j
$$\tilde{h}_j = \sum_{k \in C(j)} h_k, \quad (2)$$

Input gate of node j
$$i_j = \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right), \quad (3)$$

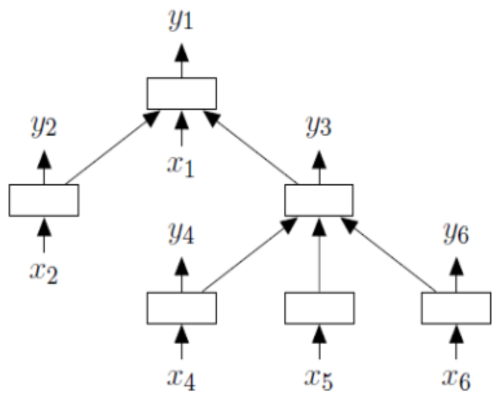
Forget gate of child k of node j
$$f_{jk} = \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right), \quad (4)$$

Output gate of node j
$$o_j = \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right), \quad (5)$$

Input of node j
$$u_j = \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right), \quad (6)$$

memory of node j
$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \quad (7)$$

Output of node j
$$h_j = o_j \odot \tanh(c_j), \quad (8)$$



- Well suited for trees with high branching factor or whose children are unordered
- Good choice for dependency trees, where the number of children of a head can be highly variable
  - Referred to as Dependency Tree-LSTM

N-ary Tree-LSTM

Input gate of node j

$$i_j = \sigma\left(W^{(i)}x_j + \sum_{\ell=1}^{N} U_{\ell}^{(i)} h_{j\ell} + b^{(i)}\right), \tag{9}$$

Forget gate of child k of node j

$$f_{jk} = \sigma\left(W^{(f)}x_j + \sum_{\ell=1}^{N} U_{k\ell}^{(f)} h_{j\ell} + b^{(f)}\right), \tag{10}$$

Output gate of node j

$$o_j = \sigma\left(W^{(o)}x_j + \sum_{\ell=1}^{N} U_{\ell}^{(o)} h_{j\ell} + b^{(o)}\right), \tag{11}$$

Input of node j

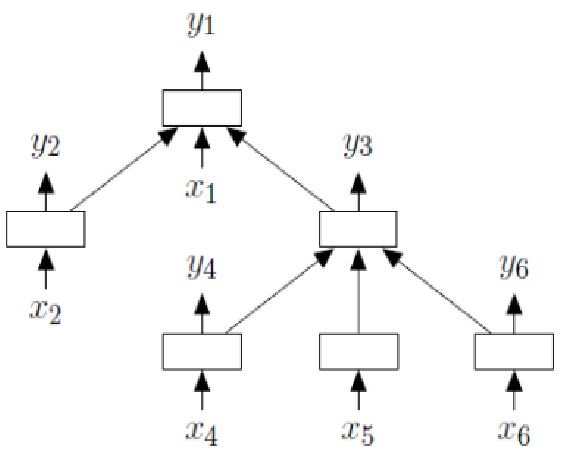
$$u_j = \tanh\left(W^{(u)}x_j + \sum_{\ell=1}^{N} U_{\ell}^{(u)} h_{j\ell} + b^{(u)}\right), \tag{12}$$

memory of node j

$$c_j = i_j \odot u_j + \sum_{\ell=1}^{N} f_{j\ell} \odot c_{j\ell}, \tag{13}$$

Output of node j

$$h_j = o_j \odot \tanh(c_j), \tag{14}$$



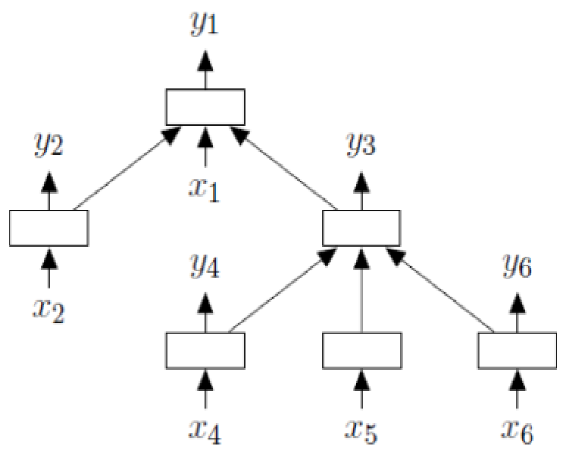
- Good for tree structures where the branching factor is at most N and where the children are ordered
  - Constituency Tree-LSTM

Task : Tree-LSTM Sentiment Classification

- Predict the labels for a subset of nodes in a tree
- Output layer for each node:

$$\hat{p}_\theta(y \mid \{x\}_j) = \text{softmax}\left(W^{(s)} h_j + b^{(s)}\right),$$

$$\hat{y}_j = \arg\max_y \hat{p}_\theta\left(y \mid \{x\}_j\right).$$



- Lost function:

$$J(\theta) = -\frac{1}{m}\sum_{k=1}^{m} \log \hat{p}_\theta\left(y^{(k)} \,\middle|\, \{x\}^{(k)}\right) + \frac{\lambda}{2}\|\theta\|_2^2,$$

Experiments: Sentiment Classification

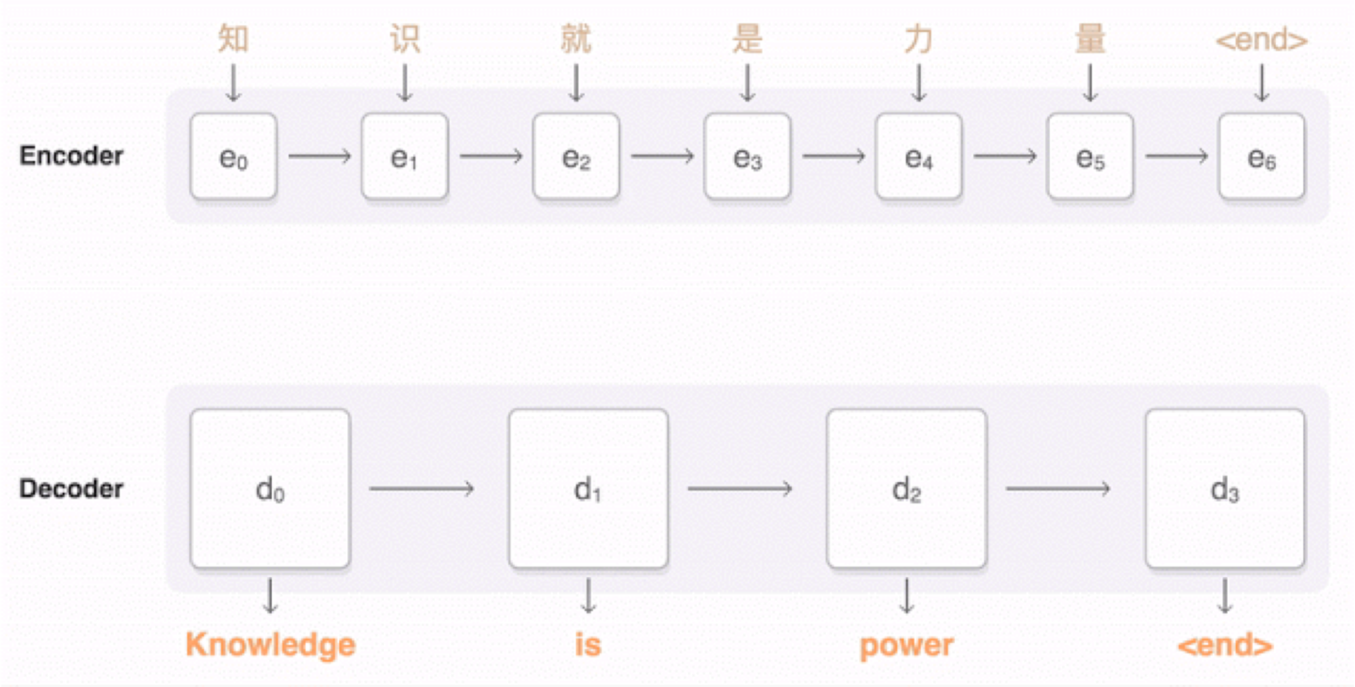
Method	Fine-grained	Binary
RAE (Socher et al., 2013)	43.2	82.4
MV-RNN (Socher et al., 2013)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
DCNN (Blunsom et al., 2014)	48.5	86.8
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	**88.1**
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
LSTM	46.4 (1.1)	84.9 (0.6)
Bidirectional LSTM	49.1 (1.0)	87.5 (0.5)
2-layer LSTM	46.0 (1.3)	86.3 (0.6)
2-layer Bidirectional LSTM	48.5 (1.0)	87.2 (1.0)
Dependency Tree-LSTM	48.4 (0.4)	85.7 (0.4)
Constituency Tree-LSTM		
– randomly initialized vectors	43.9 (0.6)	82.0 (0.5)
– Glove vectors, fixed	49.7 (0.4)	87.5 (0.8)
– Glove vectors, tuned	**51.0** (0.5)	88.0 (0.3)

Table: Results on the Stanford Sentiment Treebank

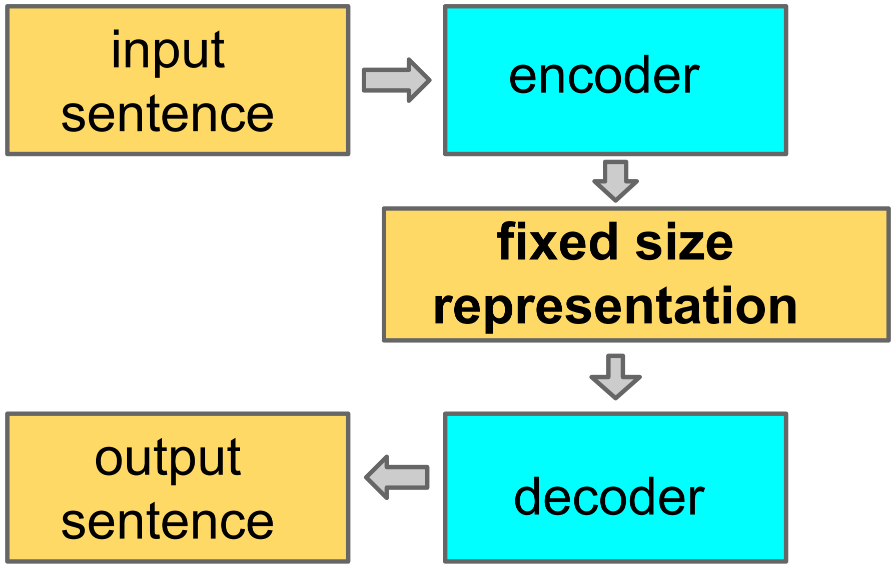
Outline

- Word Representation
  - Word2vec
- Sentence Representation
  - ParagraphVec
  - Skip-thought
  - CNN
  - LSTM & Tree-LSTM
- Machine Translation
  - Encoder-decoder
  - Attention-based encoder-decoder
  - Attention is all you need
- Question Answering
  - Memory Network
  - QANet

Neural Machine Translation



Sequence2Sequence (Encoder-Decoder)



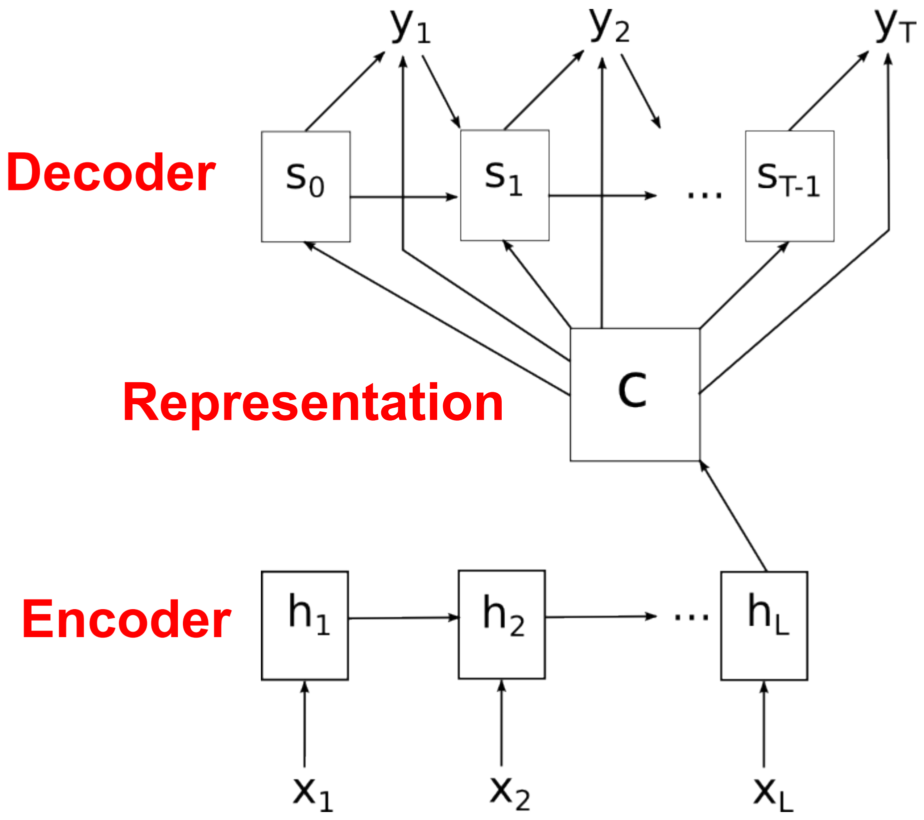
(Ñeco&Forcada, 1997)

(Kalchbrenner et al., 2013)

(Cho et al., 2014)

(Sutskever et al., 2014)

RNN Encoder-Decoder (Cho et al. 2014):



Results on Machine Translation

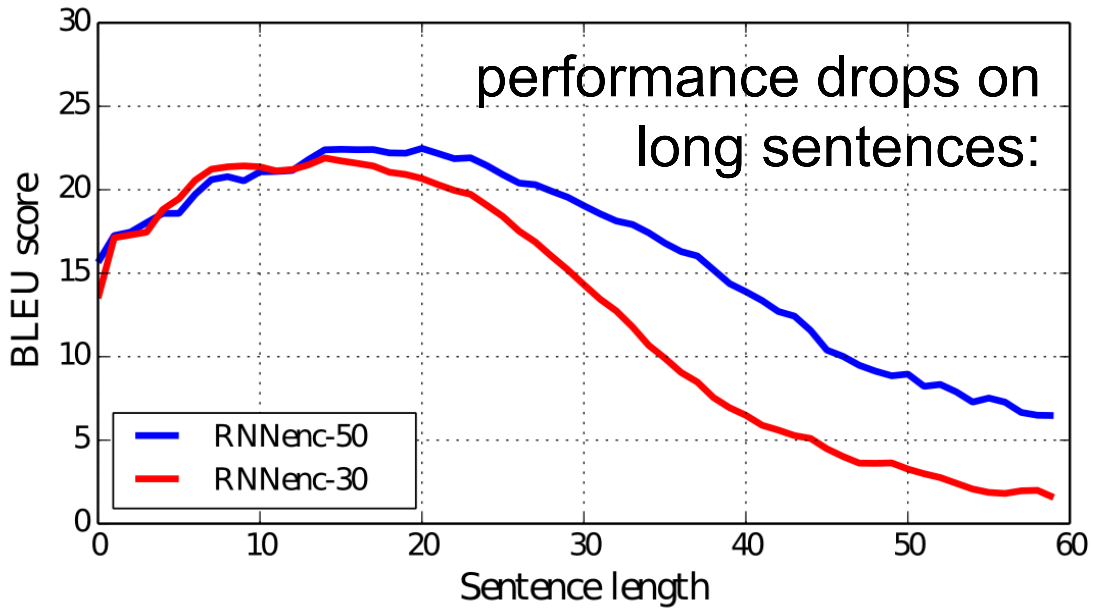
- Both encoder and decoder are RNNs

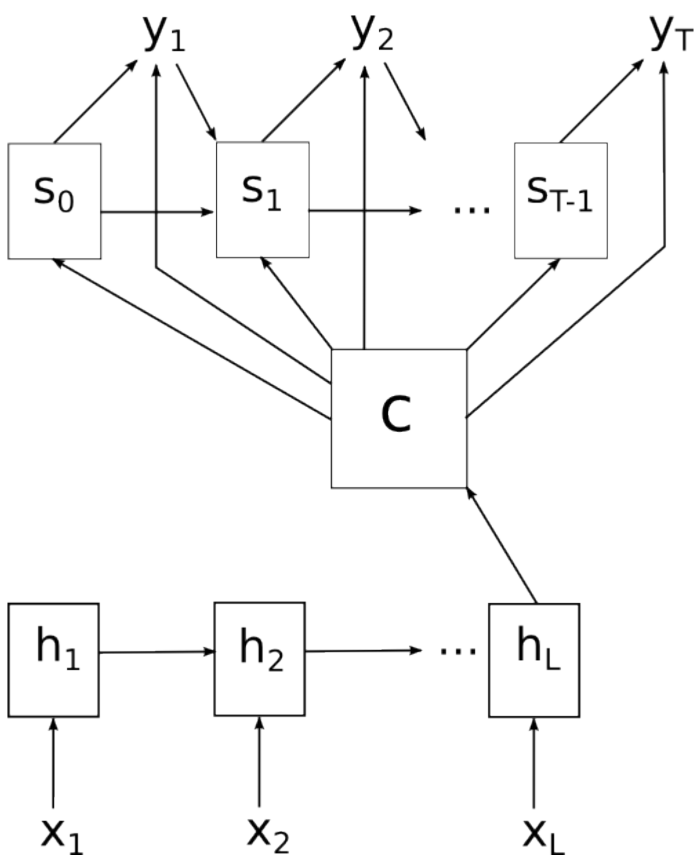
Method	**test BLEU score (ntst14)**
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

Table: Results from *English* to *French*

RNN Encoder-Decoder Issues

- has to remember the whole sentence
- fixed size representation can be the bottleneck
- humans do it differently





Attention-based Encoder-Decoder

Tell Decoder what is now translated:

The agreement on European Economic Area was signed in August 1992.

L'accord sur ???

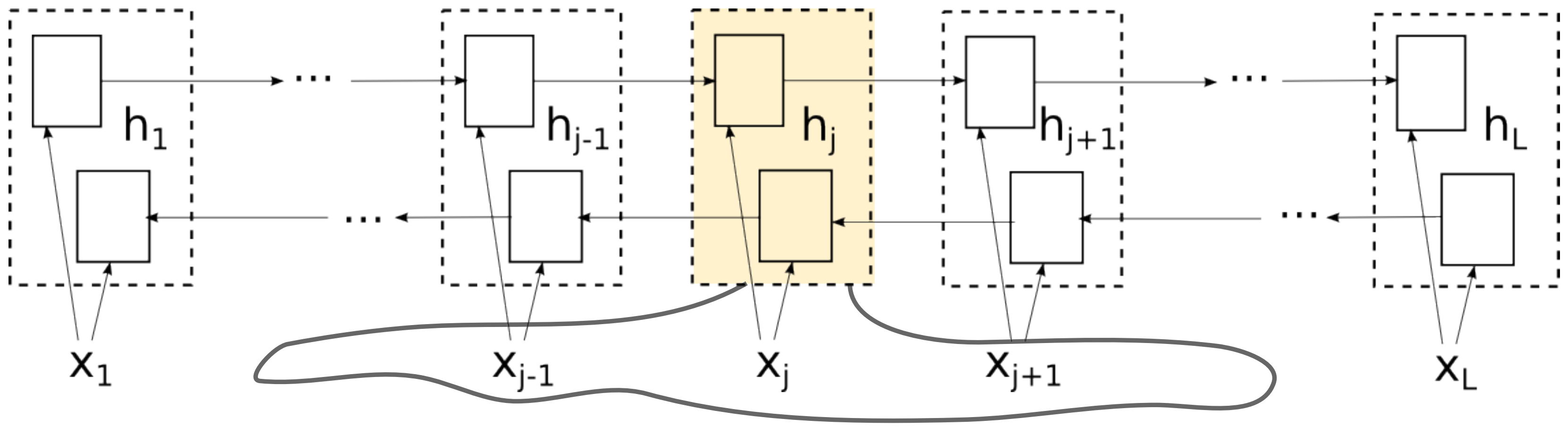
L'accord sur l'Espace économique européen a été signé en ???

Have such hints computed by the net itself!

New Encoder

Bidirectional RNN: h_j contains x_j together with its context (..., x_{j-1}, x_{j+1}, …).

(h_1, …, h_L) is the new *variable-length* representation instead of *fixed-length* c.



New Decoder

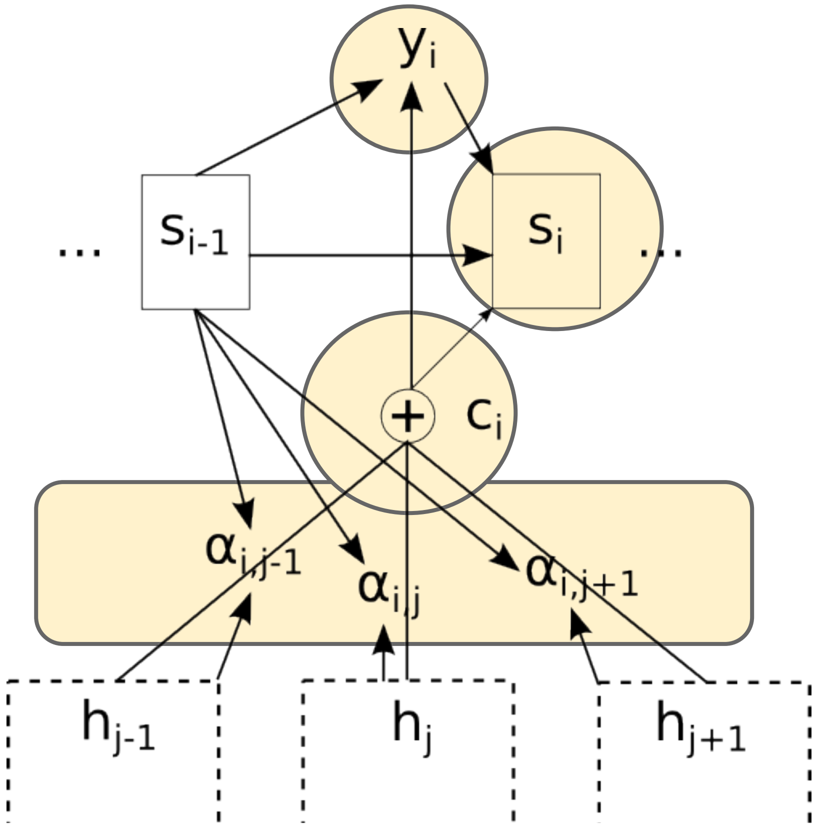
Step i:

compute alignment

compute context

generate new output

compute new decoder state



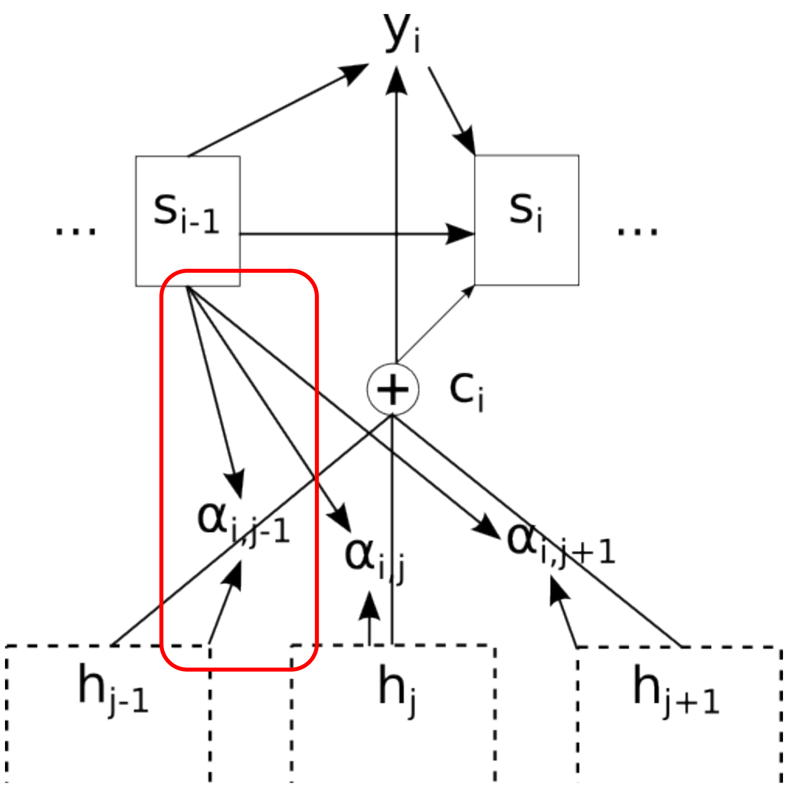
Alignment Model

$$e_{ij} = v^T \tanh(W s_{i-1} + V h_j) \qquad (1)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{L} \exp(e_{ik})} \qquad (2)$$

- nonlinearity (tanh) is crucial!
- simplest model possible
- $V h_j$ is precomputed => quadratic complexity with low constant

Calculate context: $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$



Output model

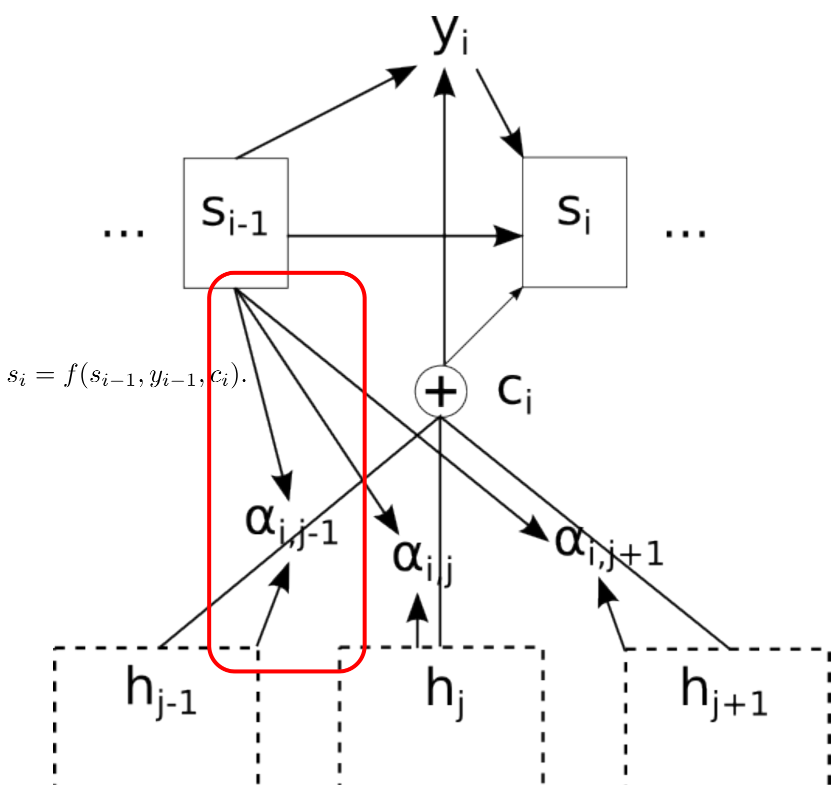
$$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1}, s_i, c_i),$$

Previous output

Current context

Previous hidden state

Architecture: Fully connected + Maxout



Update hidden state

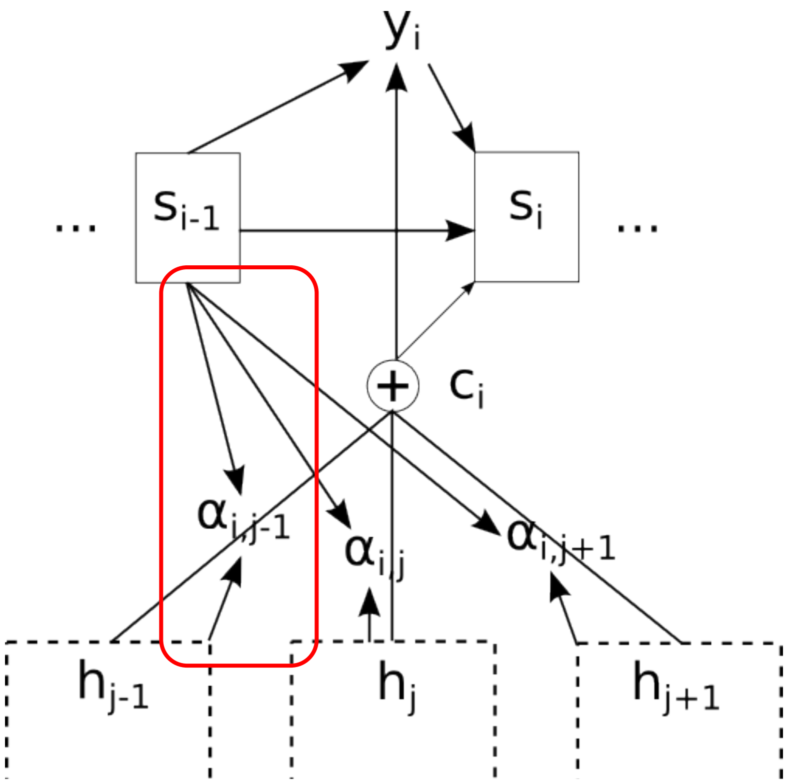
$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

Previous hidden state

Previous output

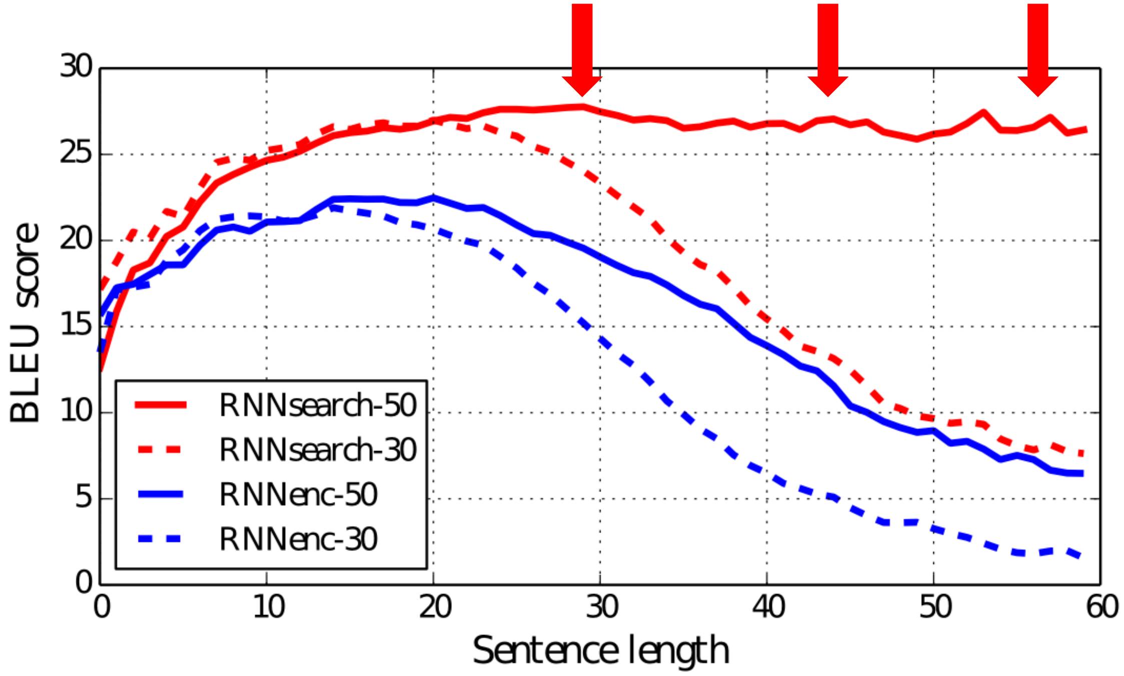
Current context

Architecture: GRU



Quantitative Results

no performance drop on long sentences



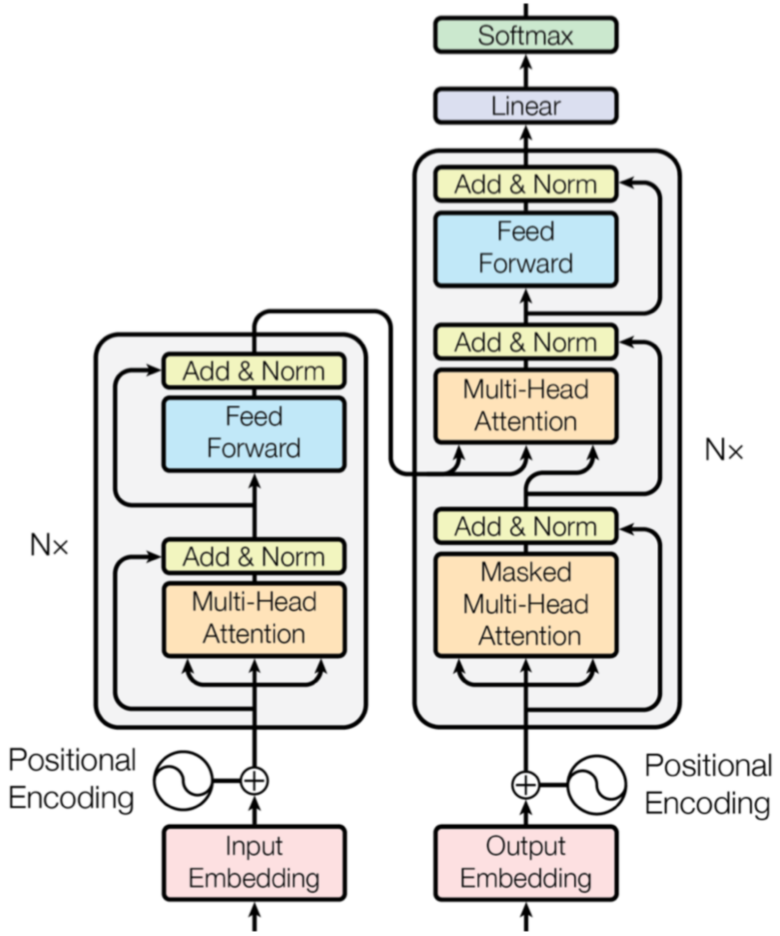
much better than RNN Encoder-Decoder

Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

without unknown words comparable with the SMT system

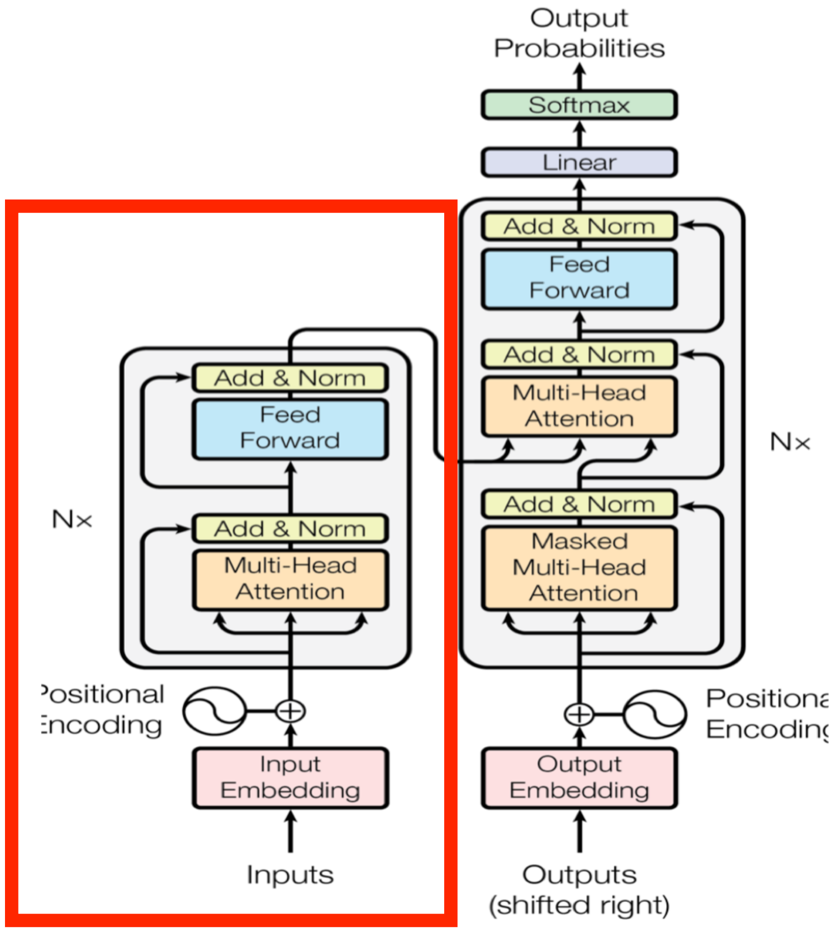
Attention is all you need (Vaswani et al. 2017)

- Most existing models for neural machine translation
  - RNN or CNN for encoder and decoder
  - Attention is used to connect encoder and decoder
- The Transformer (Vaswani et al. 2017)
  - Only attention is used
  - Parallelizable



Encoder

- A stack of N=6 identical layers
- Each layer are composed of two sublayers
  - Multi-head self-head attention
  - Position-wise fully connected feed-forward network
- Residual connection followed by normalization are used in both sublayers
  - $LayerNorm(x + Sublayer(x))$



Multi-head Attention

- Attention
  - Mapping a query and a set of key-value pairs to an output
  - Query, Keys, and Values are all vectors
  - The output is a weighted sum of the values, with the weights calculated according to a softmax function depending on the similarities between queries and keys

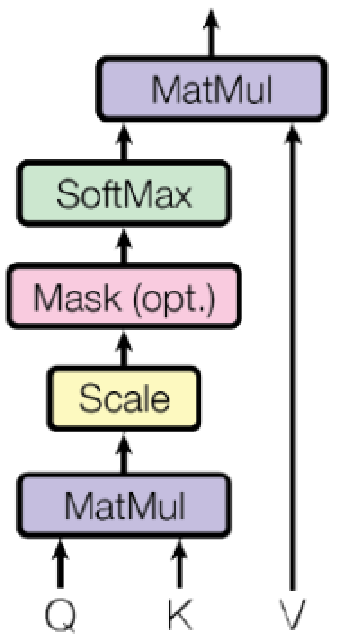
Multi-head Attention

- Scaled Dot-Product Attention
  - Avoiding pushing the softmax function into regions where it has extremely small gradients.

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

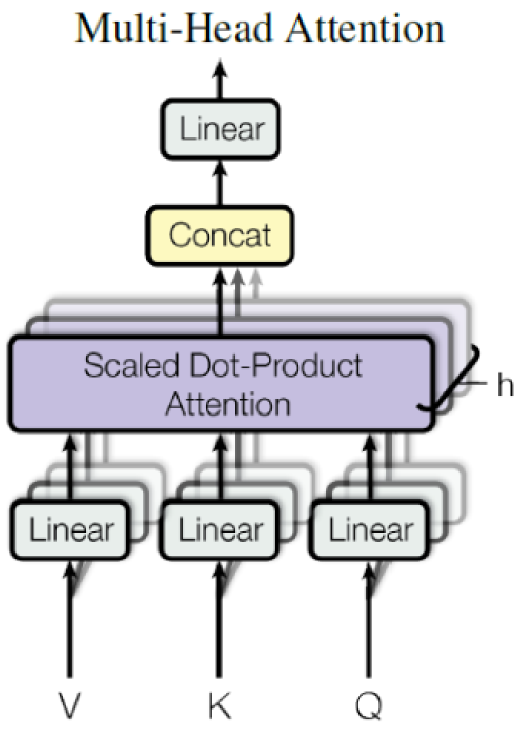
d_k: dimension of keys and queries

Scaled Dot-Product Attention



Multi-head Attention

- Multi-head Attention
  - Linearly project the queries, keys, and values h times with different, learned linear projects respectively
  - Concatenate the outputs and project again



$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, \ldots, \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Position-wise Feed-Forward Network

- Applied to each position separately and identically
  - Two linear transformations with RELU as the activation in between
  - Different parameters are used across different layers

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Positional Encoding

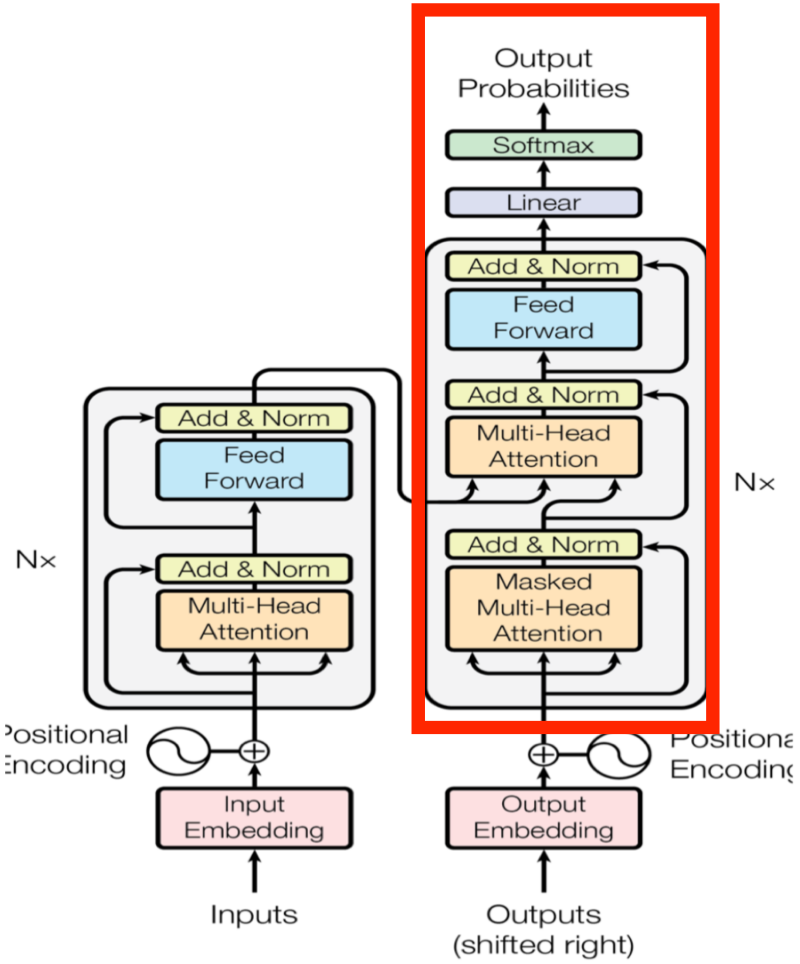
- Without recurrence and convolution, the order information is lost
- Need to encode the relative or absolute position of the tokens in the sequence
- Position encodings are added to both the embeddings of the tokens in both encoder and decoder
- Sine and cosine functions of different frequencies are used:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- Pos is the position and i is the dimension

Decoder

- N=6 identical layers
- Each layer
  - Masked multi-head attention
  - Position-wise fully connected feed-forward network
  - Multi-head attention over the output of the encoder stack
- Residual connection followed by normalization are used in all the three sublayers



Discussion: advantages of Self-Attention

- Complexity
- Short-range v.s. long-range dependency
- Interpretability

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	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Outline

- Word Representation
  - Word2vec
- Sentence Representation
  - ParagraphVec
  - Skip-thought
  - CNN
  - LSTM & Tree-LSTM
- Machine Translation
  - Encoder-decoder
  - Attention-based encoder-decoder
  - Attention is all you need
- Question Answering
  - Memory Network
  - QANet

bAbi Dataset by Facebook

- 20 tasks for text understanding and reasoning
  - Context sentences
  - Question
  - Answer

Sam walks into the kitchen.	Brian is a lion.	Mary journeyed to the den.
Sam picks up an apple.	Julius is a lion.	Mary went back to the kitchen.
Sam walks into the bedroom.	Julius is white.	John journeyed to the bedroom.
Sam drops the apple.	Bernhard is green.	Mary discarded the milk.
Q: Where is the apple?	Q: What color is Brian?	Q: Where was the milk before the den?
A. Bedroom	A. White	A. Hallway

End-to-End Memory Network (Architecture)

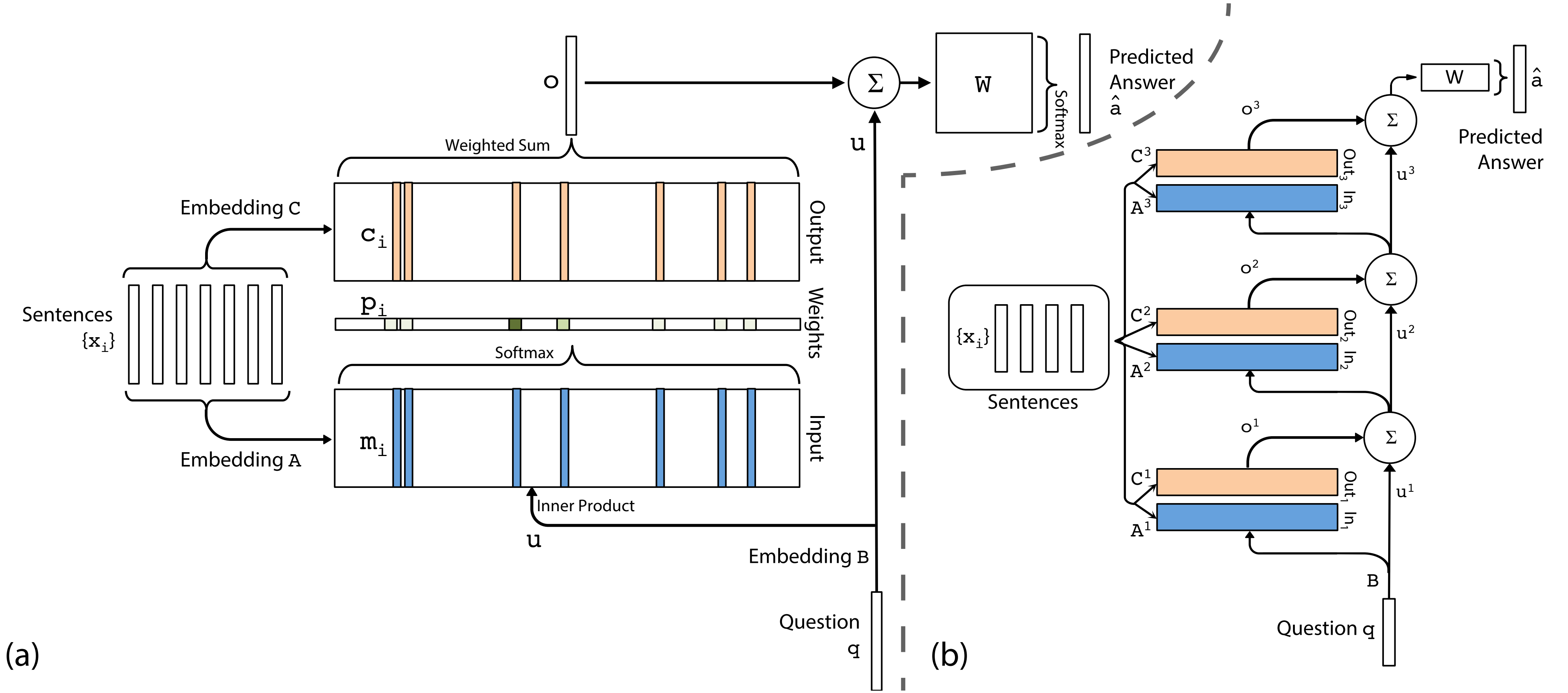


Figure 1: (a): A single layer version of our model. (b): A three layer version of our model. In practice, we can constrain several of the embedding matrices to be the same (see Section 2.2).

Input Memory Representations

- Suppose we are given an input set x1, x2, ..., xi to be stored in memory
- Convert each x_i to a d-dimensional vector m_i (e.g., with an embedding matrix $A \in R^{d \times V}$)
- The query q is also embedded (e.g., with another embedding matrix B) as the internal state u.
- Calculate the attention according to

$$p_i = \mathrm{Softmax}(u^T m_i).$$

Output memory representations

- Each x_i has a corresponding output vector c_i (with another embedding matrix C).
- The response vector from the memory o:

$$o = \sum_i p_i c_i.$$

Generating the final prediction

- The sum of the output vector o and the input embedding u is then passed through a final weight matrix W (of size V x d) and a softmax to produce the predicted label:

$$\hat{a} = \text{Softmax}(W(o + u))$$

End-to-End Memory Network (Embedding)

- Bag of Words (BoW)
  - $m_i = \sum_j Ax_{ij}$
- Position Encoding (PE)
  - $m_i = \sum_j l_j \cdot Ax_{ij}$
- Temporal Encoding (TE)
  - $m_i = \sum_j Ax_{ij} + T_A(i)$
- Random Noise (RN)
  - For regularizing T_A

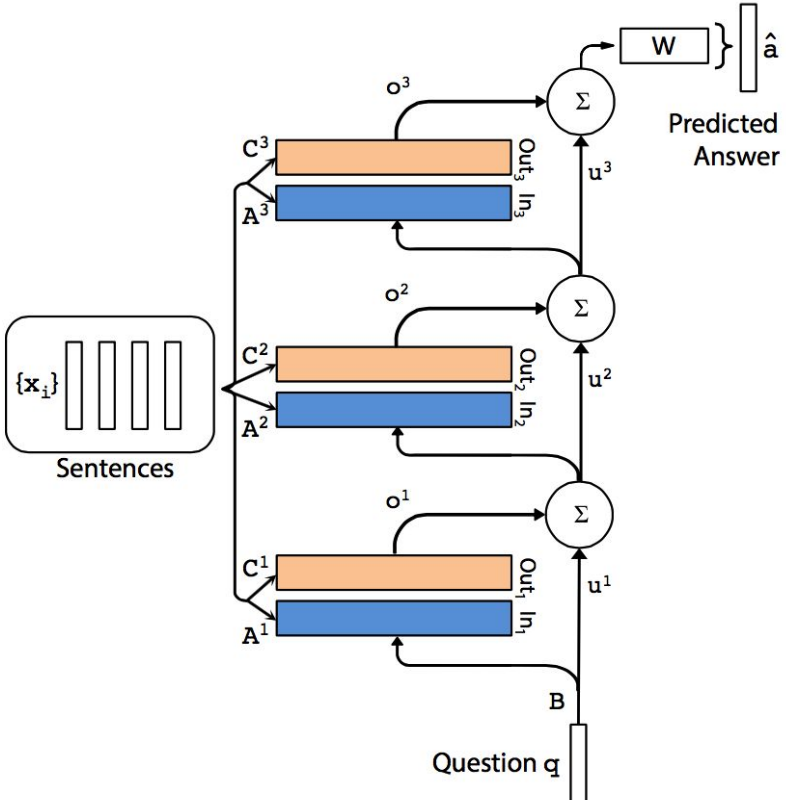
Multiple Layers:

- The input to layers above the first is the sum of output o^k and the input u^k from layer k:

$$u^{k+1} = u^k + o^k.$$

- Each layer has its own embedding matrices A^k, C^k

End-to-End Memory Network (Multiple Multiple Layers)



Types of Weight-Tying:

Adjacent:

- $A^{k+1} = C^k$
- $W^T = C^K$
- $B = A$

Layer-wise (RNN-Like):

- $A^1 = A^2 = \ldots = A^K$
- $C^1 = C^2 = \ldots = C^K$

bAbi Results

	Baseline			MemN2N								
Task	Strongly Supervised MemNN [22]	LSTM [22]	MemNN WSH	BoW	PE	PE LS	PE LS RN	1 hop PE LS joint	2 hops PE LS joint	3 hops PE LS joint	PE LS RN joint	PE LS LW joint
1: 1 supporting fact	0.0	50.0	0.1	0.6	0.1	0.2	0.0	0.8	0.0	0.1	0.0	0.1
2: 2 supporting facts	0.0	80.0	42.8	17.6	21.6	12.8	8.3	62.0	15.6	14.0	11.4	18.8
3: 3 supporting facts	0.0	80.0	76.4	71.0	64.2	58.8	40.3	76.9	31.6	33.1	21.9	31.7
4: 2 argument relations	0.0	39.0	40.3	32.0	3.8	11.6	2.8	22.8	2.2	5.7	13.4	17.5
5: 3 argument relations	2.0	30.0	16.3	18.3	14.1	15.7	13.1	11.0	13.4	14.8	14.4	12.9
6: yes/no questions	0.0	52.0	51.0	8.7	7.9	8.7	7.6	7.2	2.3	3.3	2.8	2.0
7: counting	15.0	51.0	36.1	23.5	21.6	20.3	17.3	15.9	25.4	17.9	18.3	10.1
8: lists/sets	9.0	55.0	37.8	11.4	12.6	12.7	10.0	13.2	11.7	10.1	9.3	6.1
9: simple negation	0.0	36.0	35.9	21.1	23.3	17.0	13.2	5.1	2.0	3.1	1.9	1.5
10: indefinite knowledge	2.0	56.0	68.7	22.8	17.4	18.6	15.1	10.6	5.0	6.6	6.5	2.6
11: basic coreference	0.0	38.0	30.0	4.1	4.3	0.0	0.9	8.4	1.2	0.9	0.3	3.3
12: conjunction	0.0	26.0	10.1	0.3	0.3	0.1	0.2	0.4	0.0	0.3	0.1	0.0
13: compound coreference	0.0	6.0	19.7	10.5	9.9	0.3	0.4	6.3	0.2	1.4	0.2	0.5
14: time reasoning	1.0	73.0	18.3	1.3	1.8	2.0	1.7	36.9	8.1	8.2	6.9	2.0
15: basic deduction	0.0	79.0	64.8	24.3	0.0	0.0	0.0	46.4	0.5	0.0	0.0	1.8
16: basic induction	0.0	77.0	50.5	52.0	52.1	1.6	1.3	47.4	51.3	3.5	2.7	51.0
17: positional reasoning	35.0	49.0	50.9	45.4	50.1	49.0	51.0	44.4	41.2	44.5	40.4	42.6
18: size reasoning	5.0	48.0	51.3	48.1	13.6	10.1	11.1	9.6	10.3	9.2	9.4	9.2
19: path finding	64.0	92.0	100.0	89.7	87.4	85.6	82.8	90.7	89.9	90.2	88.0	90.6
20: agent's motivation	0.0	9.0	3.6	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.2
Mean error (%)	6.7	51.3	40.2	25.1	20.3	16.3	13.9	25.8	15.6	13.3	12.4	15.2
Failed tasks (err. $> 5\%$)	4	20	18	15	13	12	11	17	11	11	11	10

bAbi Results

Story (1: 1 supporting fact)	Support	Hop 1	Hop 2	Hop 3
Daniel went to the bathroom.		0.00	0.00	0.03
Mary travelled to the hallway.		0.00	0.00	0.00
John went to the bedroom.		0.37	0.02	0.00
John travelled to the bathroom.	yes	0.60	0.98	0.96
Mary went to the office.		0.01	0.00	0.00
Where is John? Answer: bathroom Prediction: bathroom				

Story (2: 2 supporting facts)	Support	Hop 1	Hop 2	Hop 3
John dropped the milk.		0.06	0.00	0.00
John took the milk there.	yes	0.88	1.00	0.00
Sandra went back to the bathroom.		0.00	0.00	0.00
John moved to the hallway.	yes	0.00	0.00	1.00
Mary went back to the bedroom.		0.00	0.00	0.00
Where is the milk? Answer: hallway Prediction: hallway				

Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow				

Story (18: size reasoning)	Support	Hop 1	Hop 2	Hop 3
The suitcase is bigger than the chest.	yes	0.00	0.88	0.00
The box is bigger than the chocolate.		0.04	0.05	0.10
The chest is bigger than the chocolate.	yes	0.17	0.07	0.90
The chest fits inside the container.		0.00	0.00	0.00
The chest fits inside the box.		0.00	0.00	0.00
Does the suitcase fit in the chocolate? Answer: no Prediction: no				

Language Model

- Adaptation to LM
  - Inputs are words, not sentences
  - Question q is assumed to have constant embeddings (0.1)
  - Output softmax is applied to the whole dictionary
- Layer-wise weight sharing

Results

Model	Penn Treebank # of hidden	# of hops	memory size	Valid. perp.	Test perp.	Text8 # of hidden	# of hops	memory size	Valid. perp.	Test perp.
RNN [15]	300	-	-	133	129	500	-	-	-	184
LSTM [15]	100	-	-	120	115	500	-	-	122	154
SCRN [15]	100	-	-	120	115	500	-	-	-	161
MemN2N	150	2	100	128	121	500	2	100	152	187
	150	3	100	129	122	500	3	100	142	178
	150	4	100	127	120	500	4	100	129	162
	150	5	100	127	118	500	5	100	123	154
	150	6	100	122	115	500	6	100	124	155
	150	7	100	120	114	500	7	100	118	**147**
	150	6	25	125	118	500	6	25	131	163
	150	6	50	121	114	500	6	50	132	166
	150	6	75	122	114	500	6	75	126	158
	150	6	100	122	115	500	6	100	124	155
	150	6	125	120	112	500	6	125	125	157
	150	6	150	121	114	500	6	150	123	154
	150	7	200	118	**111**	-	-	-	-	-

SQUAD (Rajpurkar et al. 2016)

- 500 Wikipedia articles, 20k paragraphs
- The questions and answers are collected by crowdsourcing
  - Given a paragraph, the workers are required to return 5 questions and answers
  - Each answer is a span in the given paragraph
  - 100k questions in total, covering a wide range of topics

Passage: Tesla later approached Morgan to ask for more funds to build a more powerful transmitter. **When asked where all the money had gone, Tesla responded by saying that he was affected by the Panic of 1901**, which he (Morgan) had caused. Morgan was shocked by the reminder of his part in the stock market crash and by Tesla's breach of contract by asking for more funds. Tesla wrote another plea to Morgan, but it was also fruitless. Morgan still owed Tesla money on the original agreement, and Tesla had been facing foreclosure even before construction of the tower began.

Question: On what did Tesla blame for the loss of the initial money?

Answer: Panic of 1901

Different Types of Questions and Answers

Answer type	Percentage	Example
Date	8.9%	19 October 1512
Other Numeric	10.9%	12
Person	12.9%	Thomas Coke
Location	4.4%	Germany
Other Entity	15.3%	ABC Sports
Common Noun Phrase	31.8%	property damage
Adjective Phrase	3.9%	second-largest
Verb Phrase	5.5%	returned to Earth
Clause	3.7%	to avoid trivialization
Other	2.7%	quietly

Reasoning	Description	Example	Percentage
Lexical variation (synonymy)	Major correspondences between the question and the answer sentence are synonyms.	Q: What is the Rankine cycle sometimes **called**? Sentence: The Rankine cycle is sometimes **referred** to as a <u>practical Carnot cycle.</u>	33.3%
Lexical variation (world knowledge)	Major correspondences between the question and the answer sentence require world knowledge to resolve.	Q: Which **governing bodies** have veto power? Sen.: <u>**The European Parliament and the Council of the European Union**</u> have powers of amendment and veto during the legislative process.	9.1%
Syntactic variation	After the question is paraphrased into declarative form, its syntactic dependency structure does not match that of the answer sentence even after local modifications.	Q: What Shakespeare scholar **is currently on the faculty**? Sen.: **Current faculty include** the anthropologist Marshall Sahlins, ..., Shakespeare scholar <u>David Bevington</u>.	64.1%
Multiple sentence reasoning	There is anaphora, or higher-level fusion of multiple sentences is required.	Q: What collection does **the V&A Theatre & Performance galleries** hold? Sen.: **The V&A Theatre & Performance galleries** opened in March 2009. ... **They** hold the UK's biggest national collection of <u>material about live performance</u>.	13.6%
Ambiguous	We don't agree with the crowd-workers' answer, or the question does not have a unique answer.	Q: What is the main goal of criminal punishment? Sen.: **Achieving crime control via <u>incapacitation</u> and deterrence** is a major goal of criminal punishment.	6.1%

Baselines

- candidate answer + sentence lexical feature

	Exact Match		F1	
	Dev	Test	Dev	Test
Random Guess	1.1%	1.3%	4.1%	4.3%
Sliding Window	13.2%	12.5%	20.2%	19.7%
Sliding Win. + Dist.	13.3%	13.0%	20.2%	20.0%
Logistic Regression	40.0%	40.4%	51.0%	51.0%
Human	80.3%	77.0%	90.5%	86.8%

The current Leaderboard (2018.06.07)

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Mar 19, 2018	QANet (ensemble) *Google Brain & CMU*	**83.877**	**89.737**
2 May 10, 2018	MARS (ensemble) *YUANFUDAO research NLP*	83.520	89.612
3 Mar 06, 2018	QANet (ensemble) *Google Brain & CMU*	82.744	89.045
4 May 09, 2018	MARS (single model) *YUANFUDAO research NLP*	82.587	88.880
4 Jan 22, 2018	Hybrid AoA Reader (ensemble) *Joint Laboratory of HIT and iFLYTEK Research*	82.482	89.281
4 Feb 19, 2018	Reinforced Mnemonic Reader + A2D (ensemble model) *Microsoft Research Asia & NUDT*	82.849	88.764

QANet (Yu et al. 2018)

- Most existing models for question answering
  - RNN are used for encoding the paragraphs and queries
  - Slow for both training and inference
- A new encoder
  - Convolution: model local interaction
  - Self-attention: model global interaction

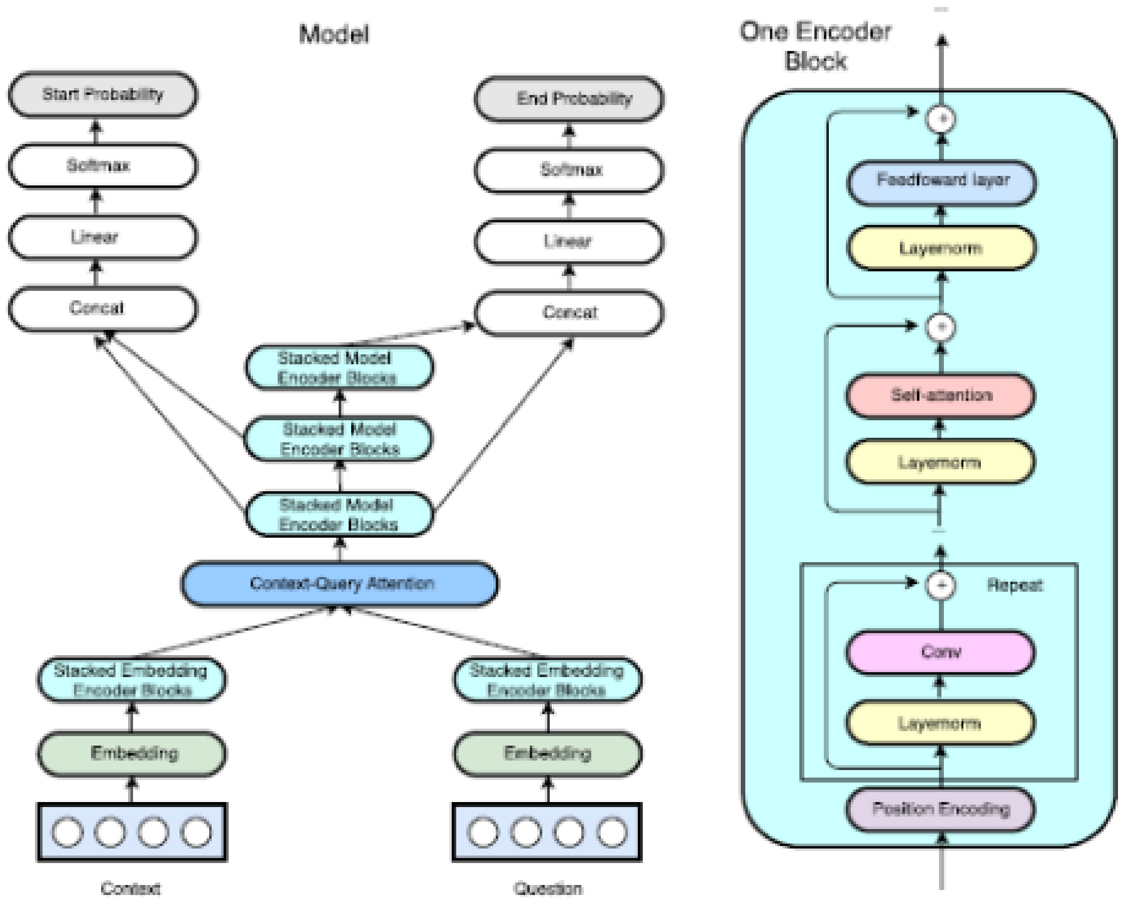


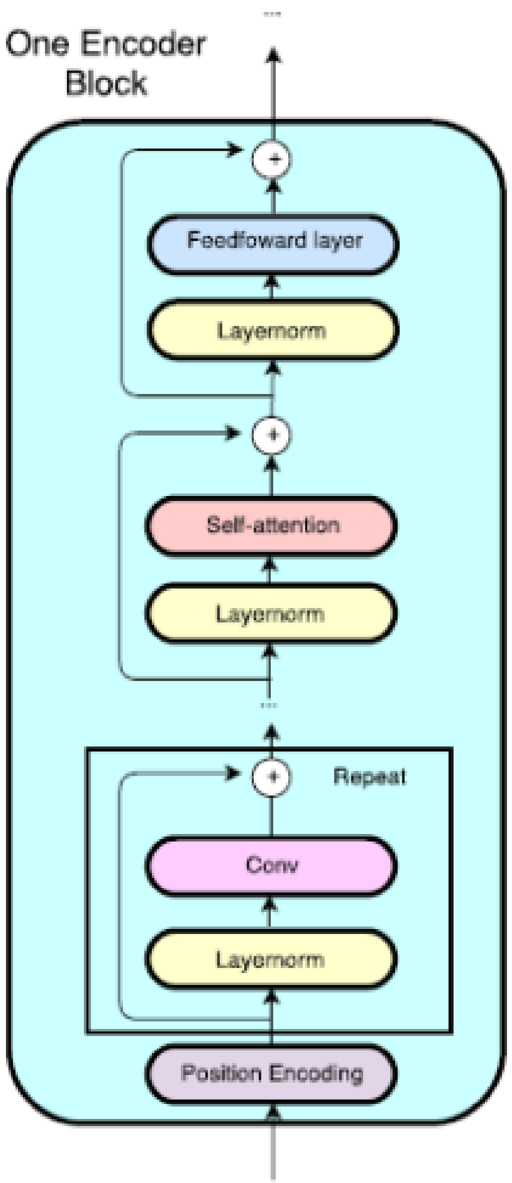
Figure 1: An overview of the QANet architecture (left) which has several Encoder Blocks. We use the same Encoder Block (right) throughout the model, only varying the number of convolutional layers for each block. We use layernorm and residual connection between every layer in the Encoder Block. We also share weights of the context and question encoder, and of the three output encoders. A positional encoding is added to the input at the beginning of each encoder layer consisting of sin and cos functions at varying wavelengths, as defined in (Vaswani et al., 2017a). Each sub-layer after the positional encoding (one of convolution, self-attention, or feed-forward-net) inside the encoder structure is wrapped inside a residual block.

Input Embedding Layer

- Word embedding: concatenating word embedding and character embedding
  - Fixed word embedding during training and initialized with pre-trained Glove vector
  - All the out-of-vocabulary words are mapped to <UNK> with trainable embedding
  - CNN on character embeddings

Embedding Encoder Layer

- [convolution-layer x # + self-attention-layer + feed-forward-layer]
- Similar to the Transformer



Context-Query Attention Layer

- Denote the encoded context and query as C and Q
- Construct a similarity matrix $S \in R^{nxm}$ between each pair of words in the context and query
  - The similarity function: $f(q, c) = W_0[q, c, q \odot c]$,
  - W_0 is trainable
- Normalize each row of S by applying the softmax function, yielding the matrix $\bar{S}$.
- Context-to-query attention $\mathrm{A} = \bar{S}Q^T \in R^{nxd}$
- Query-to-context attention $\mathrm{B} = \bar{S}\bar{\bar{S}}^T C^T \in R^{nxd}$
  - $\bar{\bar{S}}$ column normalized matrix of S by softmax function

Model Encoder Layer

- The input at each position is $[c, a, c \odot a, c \odot b]$, where a and b are a respectively a row of attention matrix A and B.
- Apply 3 layers of encoder block

Output Layer

- Predict the probability of each position in the context being the start or end of an answer span. The probability of the starting and ending position are modeled as:

$$p^1 = softmax(W_1[M_0; M_1]), \quad p^2 = softmax(W_2[M_0; M_2]),$$

- Where W1 and W2 are two trainable variables, and M0, M1, M2 are respectively the outputs of the three model encoders from bottom to top.
- The final objective function:

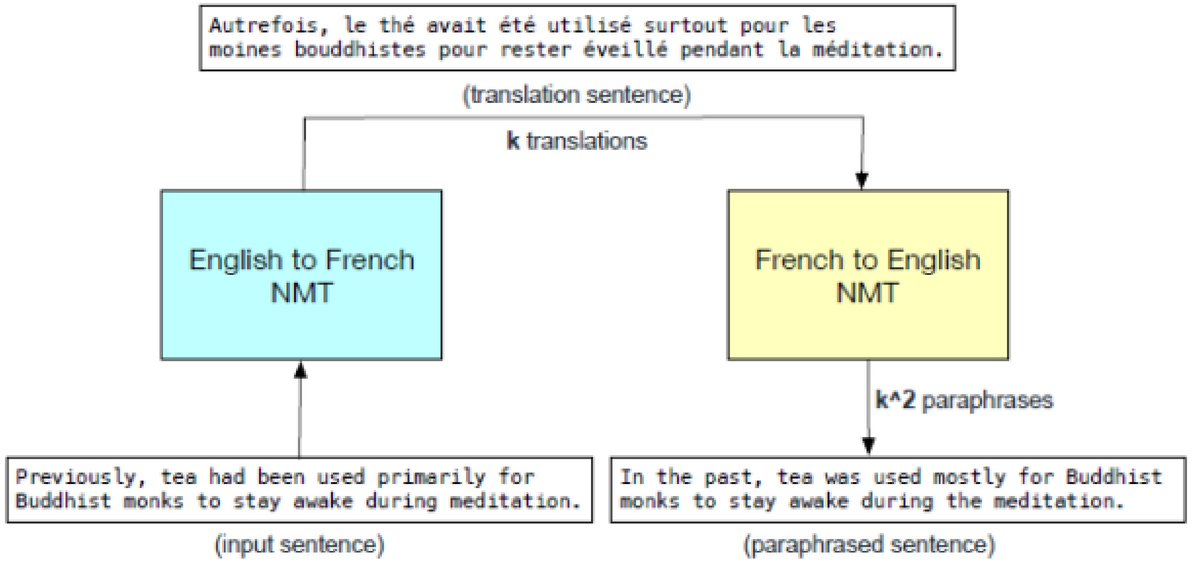
$$L(\theta) = -\frac{1}{N}\sum_{i}^{N}\left[\log(p^1_{y^1_i}) + \log(p^2_{y^2_i})\right]$$

Inference

- The predicted span (s,e) is chosen such that $p_s^1 p_e^2$ is maximized and $s \leq e$. Standard dynamic programming can be used with linear time complexity

Data Augmentation with Machine Translation

- Obtain paraphrases with machine translation models
    - One from English to French, and another from French to English



Results on SQUAD

	Published[12]	LeaderBoard[13]
Single Model	EM / F1	EM / F1
LR Baseline (Rajpurkar et al., 2016)	40.4 / 51.0	40.4 / 51.0
Dynamic Chunk Reader (Yu et al., 2016)	62.5 / 71.0	62.5 / 71.0
Match-LSTM with Ans-Ptr (Wang & Jiang, 2016)	64.7 / 73.7	64.7 / 73.7
Multi-Perspective Matching (Wang et al., 2016)	65.5 / 75.1	70.4 / 78.8
Dynamic Coattention Networks (Xiong et al., 2016)	66.2 / 75.9	66.2 / 75.9
FastQA (Weissenborn et al., 2017)	68.4 / 77.1	68.4 / 77.1
BiDAF (Seo et al., 2016)	68.0 / 77.3	68.0 / 77.3
SEDT (Liu et al., 2017a)	68.1 / 77.5	68.5 / 78.0
RaSoR (Lee et al., 2016)	70.8 / 78.7	69.6 / 77.7
FastQAExt (Weissenborn et al., 2017)	70.8 / 78.9	70.8 / 78.9
ReasoNet (Shen et al., 2017b)	69.1 / 78.9	70.6 / 79.4
Document Reader (Chen et al., 2017)	70.0 / 79.0	70.7 / 79.4
Ruminating Reader (Gong & Bowman, 2017)	70.6 / 79.5	70.6 / 79.5
jNet (Zhang et al., 2017)	70.6 / 79.8	70.6 / 79.8
Conductor-net	N/A	72.6 / 81.4
Interactive AoA Reader (Cui et al., 2017)	N/A	73.6 / 81.9
Reg-RaSoR	N/A	75.8 / 83.3
DCN+	N/A	74.9 / 82.8
AIR-FusionNet	N/A	76.0 / 83.9
R-Net (Wang et al., 2017)	72.3 / 80.7	76.5 /84.3
BiDAF + Self Attention + ELMo	N/A	**77.9/ 85.3**
Reinforced Mnemonic Reader (Hu et al., 2017)	73.2 / 81.8	73.2 / 81.8
Dev set: QANet	**73.6 / 82.7**	N/A
Dev set: QANet + data augmentation ×2	**74.5 / 83.2**	N/A
Dev set: QANet + data augmentation ×3	**75.1 / 83.8**	N/A
Test set: QANet + data augmentation ×3	**76.2 / 84.6**	76.2 / 84.6

Table 2: The performances of different models on SQuAD dataset.

Summary

- Embedding
  - Word, sentence, and document embedding
- Key techniques
  - CNN
  - RNN
  - Attention
  - Self-Attention

Thanks!