

Course number: 80240743

Deep Learning

Xiaolin Hu (胡晓林) & Jun Zhu (朱军)

Dept. of Computer Science and Technology

Tsinghua University

Last lecture review

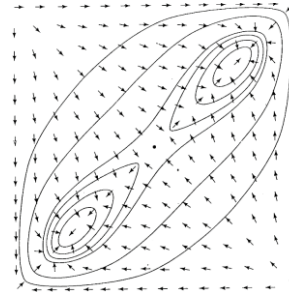
Knowledge

1. Dynamic systems

Model dynamic systems

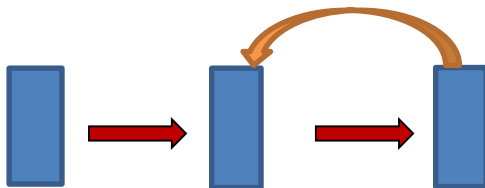


Model the brain*

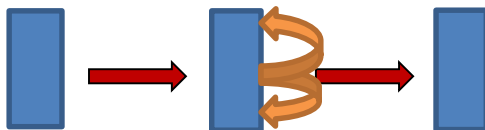


2. Simple RNNs

Jordan network



Elman network



Training

BPTT

Teacher forcing

Extensions

Bidirectional RNN

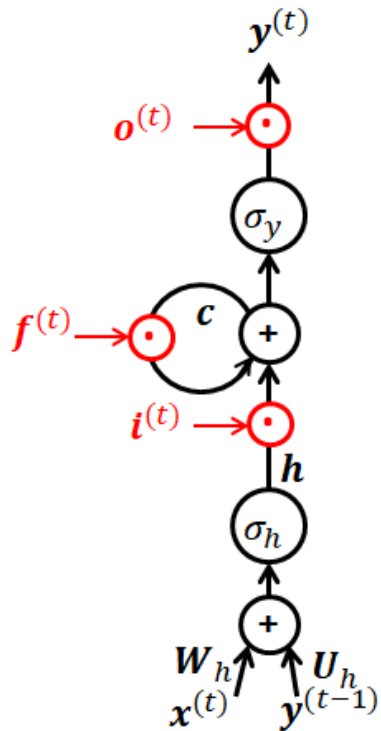
Deep RNN

Last lecture review

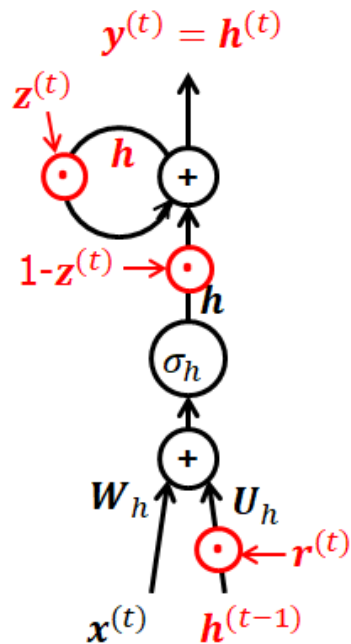
Knowledge

3. Gated RNNs

LSTM

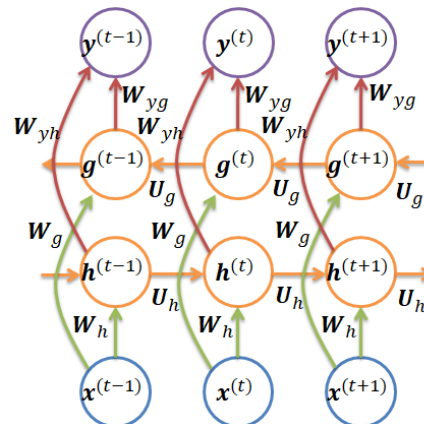


GRU

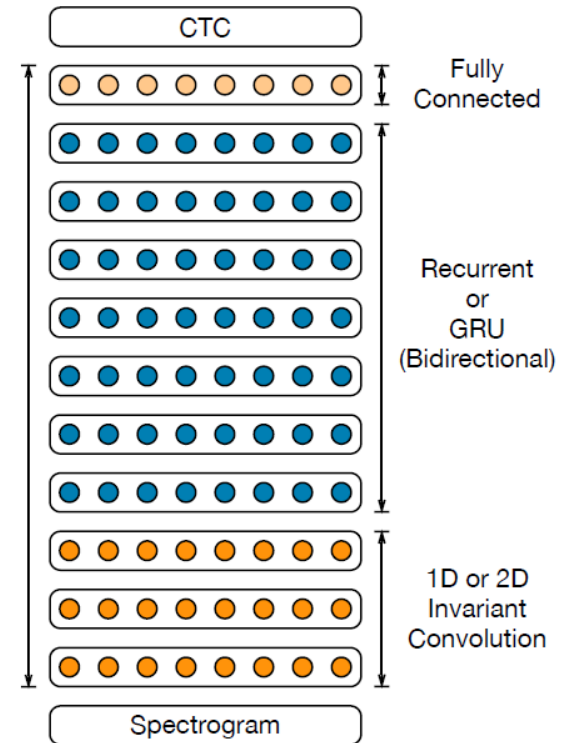


4. Applications to speech recognition

Bidirectional LSTM



Deep Speech 2



Presentation

- Form groups of 2 and every group prepares a 5-minute presentation with slides for the following paper
 - Greff, Srivastava, Koutník, Steunebrink, Schmidhuber, “LSTM: a search space odyssey,” IEEE Trans. on Neural Networks and Learning Systems, 2017
 - Liang, Hu, “Recurrent convolutional neural network for object recognition,” CVPR 2015

Project arrangement

- Apr 21 (Wednesday), 2021: Proposal
 - 5 scores
- Jun 9 (Wednesday), 2021: Presentation
 - 5 scores
- Jun 19 (Saturday), 2021: Final report
 - 30 scores
- Form groups of 2 or 3
 - Groups of 3 members are encouraged!
 - Group members get the same grade
- Topics
 - **Applications:** Apply the DL models learned in the course to a real world problem
 - **Models:** Design a new DL model, or a new variant of existing models, and apply it to tackle any task.

Proposal (200-400 words)

- What is the problem that you will be investigating? Why is it interesting?
- What reading will you examine to provide context and background?
- What data will you use? If you are collecting new data, how will you do it?
- What method or algorithm are you proposing? If there are existing implementations, will you use them and how? How do you plan to improve or modify such implementations? You don't have to have an exact answer at this point, but you should have a general sense of how you will approach the problem you are working on.
- How will you evaluate your results? Qualitatively, what kind of results do you expect (e.g. plots or figures)? Quantitatively, what kind of analysis will you use to evaluate and/or compare your results (e.g. what performance metrics or statistical tests)?

Lecture 8: NNs for Natural Language Processing

Xiaolin Hu

Dept. of Computer Science and
Technology

Tsinghua University

Outline

1. Typical tasks
2. Word representation
3. Text classification using NNs
4. Machine translation using NNs
5. Summary

Natural language processing

- **Natural language processing (NLP)** is a field of [computer science](#), [artificial intelligence](#) and [computational linguistics](#) concerned with the interactions between [computers](#) and [human \(natural\) languages](#), and, in particular, concerned with programming computers to fruitfully process large [natural language corpora](#).
- Involve [natural language understanding](#), [natural language generation](#) (frequently from [formal, machine-readable logical forms](#)), [connecting language and machine perception](#), [managing human-computer dialog systems](#), or some combination thereof.

From https://en.wikipedia.org/wiki/Natural_language_processing

Typical tasks

- Tagging and parsing
- Question answering and dialogue systems
- Text/document classification
- Sentiment analysis and opinion mining
- Machine translation
- Text generation
- Etc.

Question answering and dialogue systems

Q: How many people live in Beijing?

A: About **22 million**.

Q: What creates a cloud

A: *in meteorology , a cloud is **a visible mass of liquid droplets** or frozen crystals made of water or various chemicals suspended in the atmosphere above the surface of a planetary body.*

x_0	<i>good morning</i>
y_0	<i>hello what can i help you with today</i>
x_1	<i>can you book a table in a cheap price range in london</i>
y_1	<i>i'm on it</i>
x_2	<i><SILENCE></i>
y_2	<i>any preference on a type of cuisine</i>
x_3	<i>with french food</i>
y_3	<i>how many people would be in your party</i>
x_4	<i>for four please</i>
y_4	<i>ok let me look into some options for you</i>
x_5	<i><SILENCE></i>
y_5	<i>api_call french london four cheap</i>

Text/document classification

- Sentiment classification

Deep learning is so cool.

Positive

Dr Hu's English is very noisy!

Negative

I attended the lecture today.

Neutral

- Document classification



Machine translation

<https://translate.google.com>

The screenshot displays the Google Translate web interface. The top section shows the input language set to 'CHINESE' (highlighted in blue) and the output language set to 'ENGLISH'. The input text is '我觉得深度学习很酷，但是有太多东西要学了，感觉时间有点不够用。' (Wǒ juéde shēndù xuéxí hěn kù, dànshì yǒu tài duō dōngxī yào xuéle, gǎnjué shíjiān yǒudiǎn bùgòu yòng.). Below the input text is its pinyin transcription: 'Wǒ juéde shēndù xuéxí hěn kù, dànshì yǒu tài duō dōngxī yào xuéle, gǎnjué shíjiān yǒudiǎn bùgòu yòng.'. The bottom section shows the translated text in English: 'I think deep learning is cool, but there are so many things to learn, and I feel that time is not enough.' The interface includes various icons for voice input, voice output, and a star icon for saving the translation.

DETECT LANGUAGE CHINESE ENGLISH SPANISH ▼

我觉得深度学习很酷，但是有太多东西要学了，感觉时间有点不够用。|

Wǒ juéde shēndù xuéxí hěn kù, dànshì yǒu tài duō dōngxī yào xuéle, gǎnjué shíjiān yǒudiǎn bùgòu yòng.

31/5000 拼

↔ ENGLISH CHINESE (SIMPLIFIED) SPANISH ▼

I think deep learning is cool, but there are so many things to learn, and I feel that time is not enough. ☆

🔊 📄 ✎ 🔗

Chinese couplet generation

<http://couplet.msra.cn/app/couplet.aspx>

上联 青 | 山 | 不 | 墨 | 千 | 秋 | 画

下联

横批

在输入框内输入部分下联，点击刷新候选，系统会根据规定生成完整下联

刷新候选

- ☐ 绿水无弦万古琴
- ☐ 碧水多情万古诗
- ☐ 流水无弦万古琴
- ☐ 白雪红梅迎春来
- ☐ 绿水无声鸟作歌
- ☐ 流水有声万古琴
- ☐ 流水无声万古琴
- ☐ 碧水无弦万古琴
- ☐ 绿水无言万古诗
- ☐ 澧水无弦万古琴

上联 云 | 影 | 波 | 光 | 天 | 上 | 下

下联

横批

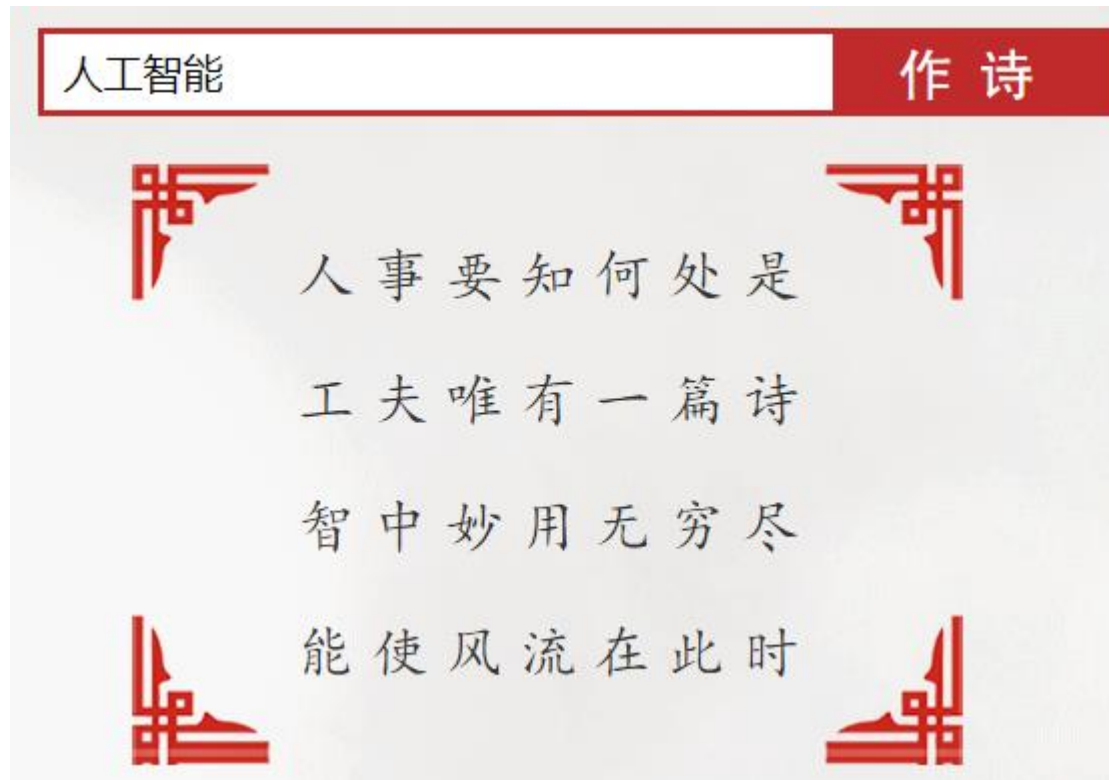
在输入框内输入部分下联，点击刷新候选，系统会根据规定生成完整下联

刷新候选

- ☐ 松涛竹韵水中央
- ☐ 涛声音籁海中透
- ☐ 涛音松籁水中央
- ☐ 松涛竹韵水中流
- ☐ 月辉银霞世春秋
- ☐ 风香月色眼前生
- ☐ 花香柳色水边生
- ☐ 花容月色水中生
- ☐ 风香月色水中间
- ☐ 风声月色眼里出

Chinese ancient poetry generation

<http://jiuge.thunlp.org/>



Outline

1. Typical tasks
2. Word representation
3. Text classification using NNs
4. Machine translation using NNs
5. Summary

Word representation

- One-hot representation

Flower: [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]

Phone: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]

Dimensionality: the vocabulary size $|\mathbf{V}|$ (could be millions)

- Problems:

- Dimensionality is high
- Does not represent the relationship between words

dist(“kid”, “child”) has nothing to do with dist(“flower”, “car”)

Word representation

- **Motivation:** You can get a lot of value by representing a word by means of its neighbors

“You shall know a word by the company it keeps”
(J. R. Firth 1957: 11)

...government debt problems turning into **banking**
crises as has happened in...
...saying that Europe needs unified **banking** regulation
to replace the hodgepodge...

These words will represent **banking**

One of the most successful ideas of modern statistical NLP!

How to make neighbors represent words

use a **co-occurrence matrix** !

- Two options: full document vs windows
 - **Word-document co-occurrence matrix** will give general topics (all sports terms will have similar entries) leading to “Latent Semantic Analysis”
 - **Window around each word** captures both syntactic and semantic information

Window based co-occurrence matrix

Slide from Richard Socher

- Window length 1 (more common: 5 - 10)
- Symmetric (irrelevant whether left or right context)

Example corpus

I like deep learning.
I like NLP.
I enjoy flying.

counts	I	like	deep	learning	NLP	enjoy	flying
I	0	2	0	0	0	1	0
like	2	0	1	0	1	0	0
deep	0	1	0	1	0	0	0
learning	0	0	1	0	0	0	0
NLP	0	1	0	0	0	0	0
enjoy	1	0	0	0	0	0	1
flying	0	0	0	0	0	1	0

What' s the problem with this co-occurrence matrix representation?

- ☒ A Dimension is high
- ☒ B Representation is sparse making models less robust
- ☐ C Does not represent the relationship between words
- ☒ D Need to recalculate the representation of all words by adding a word into the vocabulary

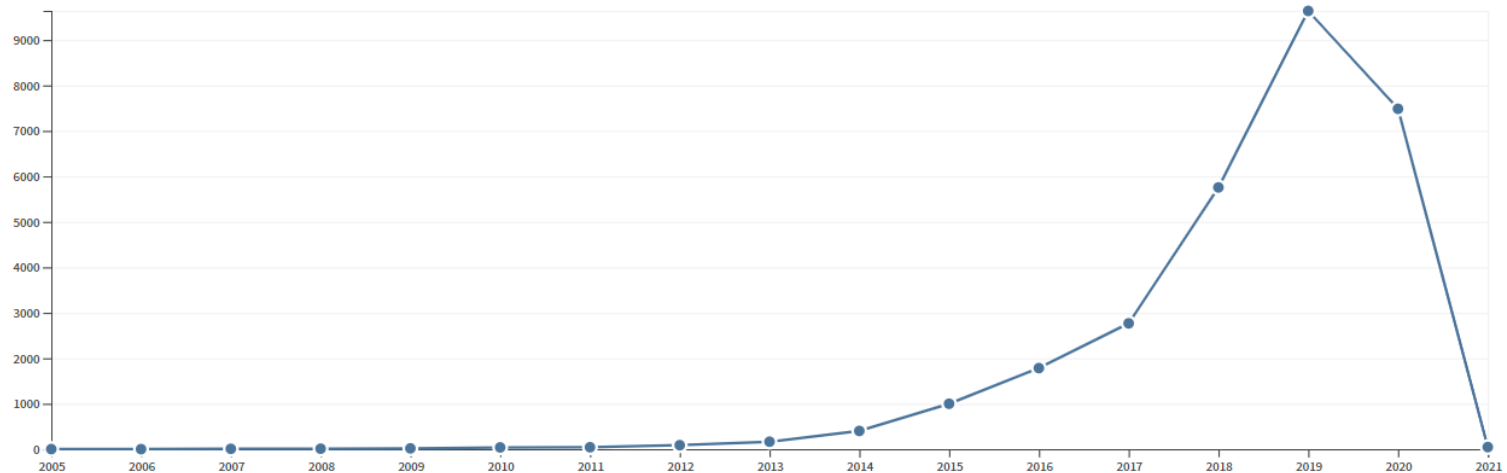
Submit

Represent words by low-dimensional vectors

- **Idea:** store “most” of the important information in a fixed, small number of dimensions: **a dense vector**
 - Usually around 25 – 1000 dimensions
 - It’s easy to perform tasks (classification, generation, etc.) based on this representation
- How to learn the word vectors?
 - A neural probabilistic language model (Bengio et al., JMLR, 2003)
 - A recent simpler and faster model (Mikolov et al., NeurIPS 2013): **word2vec** → **Intro here**

Influence of the seminal work

Bengio et al., A neural probabilistic language model.
Journal of Machine Learning Research, 2003



From Web of Science

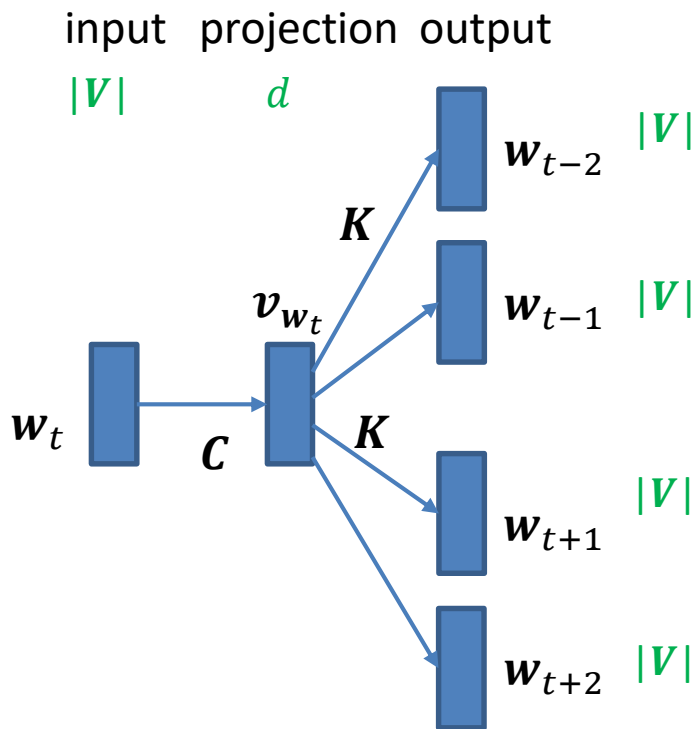


Yoshua Bengio
Turing Award 2018

...In 2000 he made a major contribution to natural language processing with the paper “A Neural Probabilistic Language Model.”

Main idea of word2vec

- Instead of capturing co-occurrence counts directly, **predict** surrounding words of every word

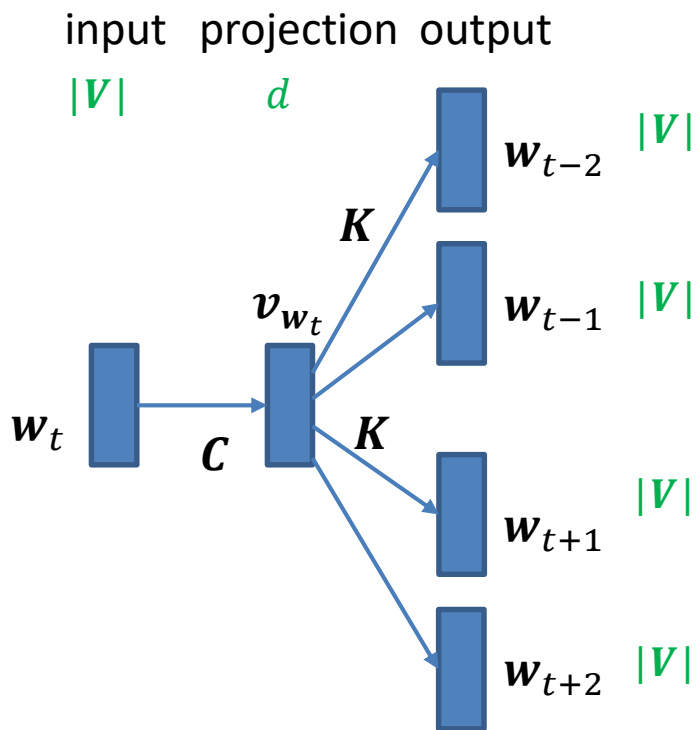


- $w_t \in R^{|V|}, v_{w_t} \in R^d$
- $C \in R^{d \times |V|}$: Each column is the vector of a word in the vocabulary

$|V|$ is about $10^5 \sim 10^7$
 d is about $50 \sim 1000$

Main idea of word2vec

- Instead of capturing co-occurrence counts directly, **predict** surrounding words of every word



$|V|$ is about $10^5 \sim 10^7$
 d is about $50 \sim 1000$

- $w_t \in R^{|V|}$, $v_{w_t} \in R^d$
- $C \in R^{d \times |V|}$: Each column is the vector of a word in the vocabulary

① Project w_t to v_{w_t} via C

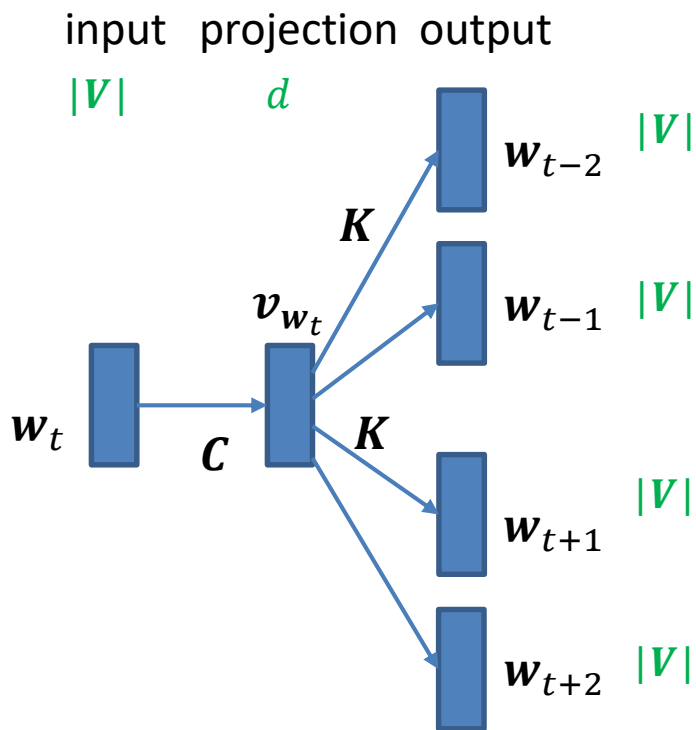
$$C \cdot w_t = v_{w_t}$$

d $|V|$

The one-hot vector w_t selects the corresponding column in C

Main idea of word2vec

- Instead of capturing co-occurrence counts directly, **predict** surrounding words of every word



$|V|$ is about $10^5 \sim 10^7$
 d is about $50 \sim 1000$

- $w_t \in R^{|V|}$, $v_{w_t} \in R^d$
 - $C \in R^{d \times |V|}$: Each column is the vector of a word in the vocabulary
- ② Project v_{w_t} to a word at location $t - 2$ via $K \in R^{|V| \times d}$

$$K \cdot v_{w_t} \xrightarrow{\text{softmax}} \text{GT } w_{t-2}$$

$$\begin{matrix} |V| \\ \left\{ \begin{array}{c} \text{---} \\ \text{---} \\ \vdots \\ \text{---} \end{array} \right. \end{matrix} \cdot \begin{matrix} d \\ \text{---} \end{matrix} = \begin{bmatrix} 1.1 \\ 0.0 \\ 0.1 \\ 2.8 \\ 3.4 \end{bmatrix} \xrightarrow{\text{softmax}} \begin{bmatrix} 0.1 \\ 0.0 \\ 0.0 \\ 0.2 \\ 0.6 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

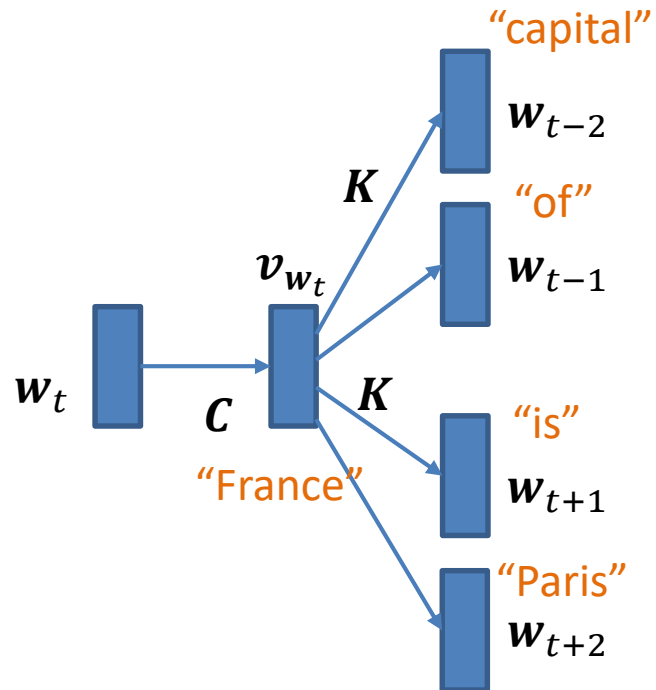
d

Input vector versus output vector

- Each **column** of \mathbf{C} corresponds to a word: “input vector”
- Each **row** of \mathbf{K} corresponds to a word: “output vector”
- Clearly, we need to predefine the correspondence between the columns of \mathbf{C} and the words in the corpus
- In order to do training, the same order of rows must be used in \mathbf{K}
- After learning, the i -th column of \mathbf{C} and the i -th row of \mathbf{K} can be averaged to represent the i -th word

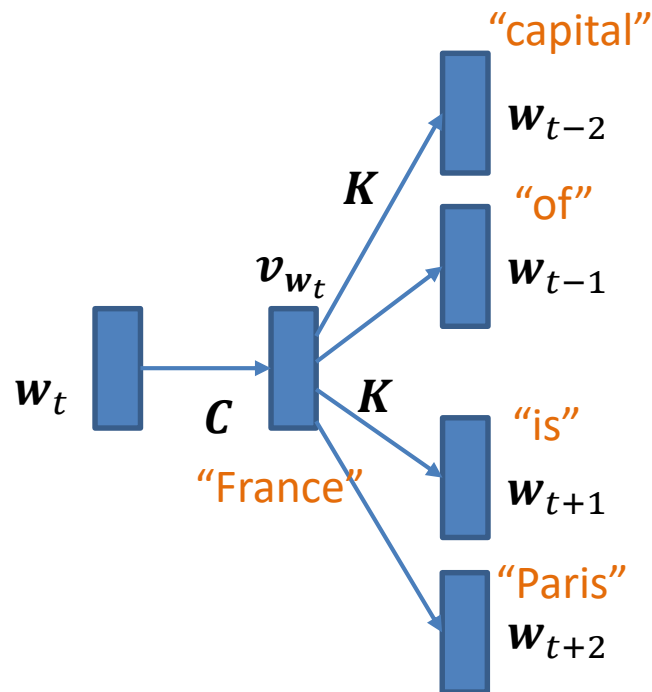
An example

“...capital of France is Paris...”



An example

“...capital of France is Paris...” • Project v_{w_t} to a word at location $t - 2$



$$K \cdot v_{w_t} \xrightarrow{\text{softmax}} \text{GT } w_{t-2}$$

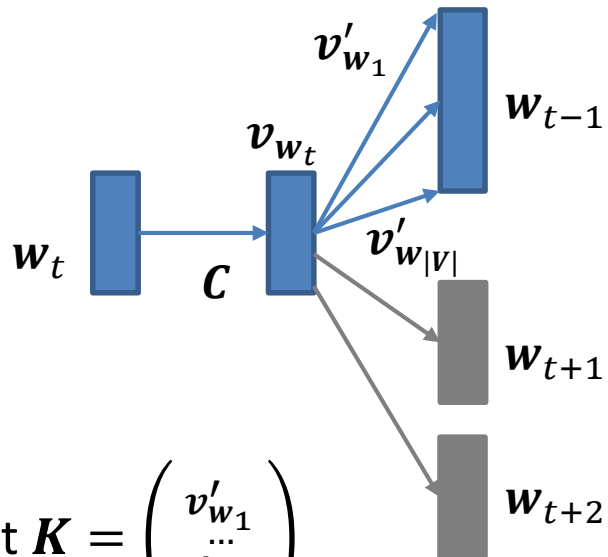
The diagram shows the calculation of the projection. A matrix K (with rows $v'_{\text{"capital"}}$, $v'_{\text{"of"}}$, $v'_{\text{"is"}}$, $v'_{\text{"Paris"}}$) is multiplied by the vector $v_{\text{"France"}}$. The result is a vector of scores $[1.1, 0.0, 2.8, 0.3, 3.4]$, which is then passed through a softmax function to produce the output vector $[0.1, 0.0, 0.3, 0.0, 0.6]$. The ground truth vector w_{t-2} is $[0, 0, 0, 0, 1]$.

- The row in K corresponding to “capital” should be similar to $v_{\text{“France”}}$
- Similarly, the rows corresponding to “of”, “is”, “Paris” should also be similar to $v_{\text{“France”}}$

Formulation

- **Objective:** given an input word, maximize the prob of surrounding words

skip-gram model

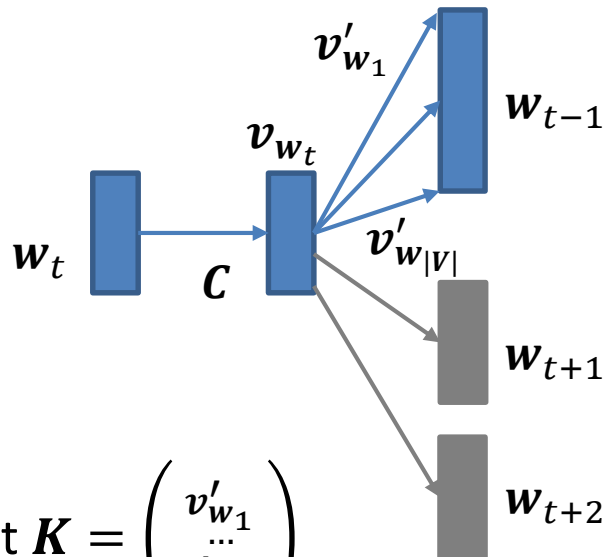


$$\text{Let } K = \begin{pmatrix} v'_{w_1} \\ \dots \\ v'_{w_{|V|}} \end{pmatrix}$$

Formulation

- **Objective:** given an input word, maximize the prob of surrounding words

skip-gram model



Let $K = \begin{pmatrix} v'_{w_1} \\ \dots \\ v'_{w_{|V|}} \end{pmatrix}$

- The prob of a word w_k at location $t + j$ ($j = -2, -1, 1, 2$) is

$$p(w_k | w_t) = \frac{\exp(v'_{w_k}^\top v_{w_t})}{\sum_{m=1}^{|V|} \exp(v'_{w_m}^\top v_{w_t})}$$

- During training, a word w_{t+j} , is given as the *desired* output word, the cross-entropy loss:

$$-\sum_{k=1}^{|V|} r_{w_k} \ln p(w_k | w_t) = -\ln p(w_{t+j} | w_t)$$

Objective function

- We have same requirement at other locations in the window.
Sum the losses over all locations

$$J^t(\boldsymbol{\theta}) = - \sum_{-c \leq j \leq c, j \neq 0} \ln p(\mathbf{w}_{t+j} | \mathbf{w}_t)$$

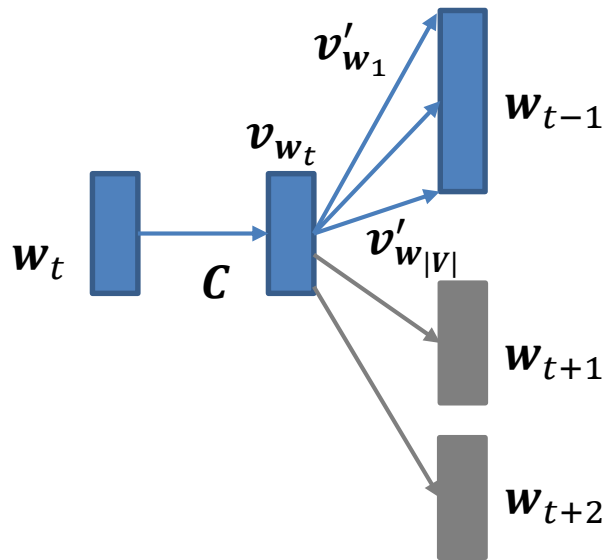
where c is the window size and $\boldsymbol{\theta}$ denote all parameters

- Average over all given input $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \dots, \mathbf{w}_T$,

$$J(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^T J^t(\boldsymbol{\theta}) = - \frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \ln p(\mathbf{w}_{t+j} | \mathbf{w}_t)$$

- It's equivalent to maximizing the average log probability
- BP algorithm and SGD are used to train the model

Alternatives to full softmax



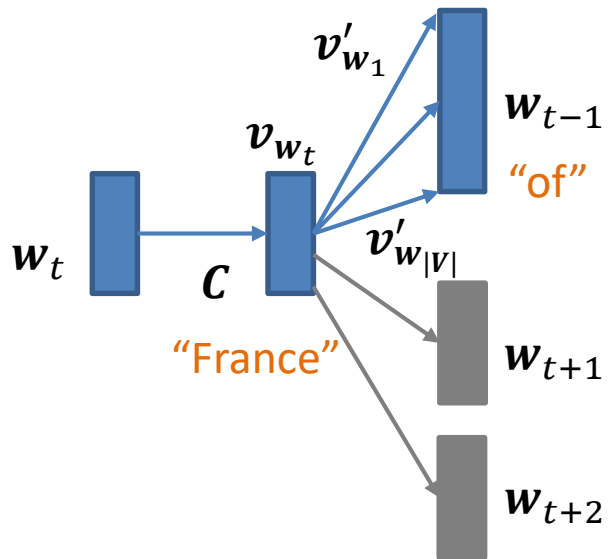
At *each location* in the window

$$p(\mathbf{w}_k = \text{"at this location"} | \mathbf{w}_t) = \frac{\exp(\mathbf{v}'_{\mathbf{w}_k}^\top \mathbf{v}_{\mathbf{w}_t})}{\sum_{m=1}^{|V|} \exp(\mathbf{v}'_{\mathbf{w}_m}^\top \mathbf{v}_{\mathbf{w}_t})}$$

where $k = 1, \dots, |V|$

- Any problem with this formulation?
 - The normalization term is computationally expensive
- Two alternatives
 - **Hierarchical softmax** (Morin, Bengio, 2005; Mikolov et al., 2013)
 - **Negative sampling** (Mikolov et al., 2013) -> [Intro here](#)

Motivation



Suppose $t - 1$ is in the window

At $t - 1$, Softmax calculates the probs:

$$p(\text{"of"} = \text{"at } t - 1" | \text{"France"})$$

$$p(\text{"Beijing"} = \text{"at } t - 1" | \text{"France"})$$

$$p(\text{"fish"} = \text{"at } t - 1" | \text{"France"})$$

...

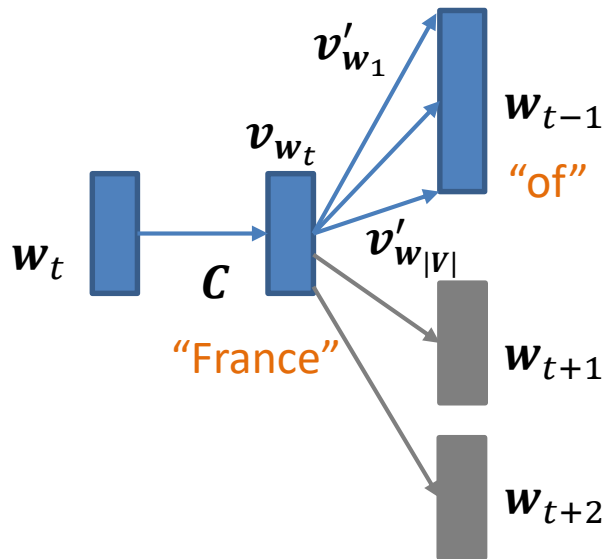
$p(\text{"of"} = \text{"at } t - 1" | \text{"France"})$ is larger than other probs because "of" appears at $t - 1$ in the training set

Construct a binary classification problem at $t - 1$:

- "of" is at $t - 1$
- other $|V| - 1$ words are not at $t - 1$

Logistic
regression!

Motivation



At *each location* in the window, calculate

$$p(\mathbf{w}_k = \text{"at this location"} | \mathbf{w}_t)$$

$$= \frac{1}{1 + \exp(-\mathbf{v}'_{\mathbf{w}_k}{}^\top \mathbf{v}_{\mathbf{w}_t})}$$

where $k = 1, \dots, |V|$

Sigmoid
function

Suppose $t - 1$ is in the window

At $t - 1$, Sigmoid calculates the probs:

$$p(\text{"of"} = \text{"at } t - 1" | \text{"France"}) > 0.5$$

$$p(\text{"Beijing"} = \text{"at } t - 1" | \text{"France"}) < 0.5$$

$$p(\text{"fish"} = \text{"at } t - 1" | \text{"France"}) < 0.5$$

...

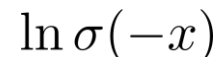
$|V| - 1$ Sample K terms,
terms where $K \ll |V| - 1$

Negative sampling

- We use **logistic regression** to approximate softmax
 - Classify $K + 1$ words into **two classes**: (1) appears in the window; (2) doesn't appear in the window
 - One **positive sample**, K **negative samples** from a distribution $P(\mathbf{w})$
 - K can be 5~20 for small training datasets, while 2~5 for large datasets

Recall: cross-entropy error for logistic regression (for one input sample)

$$E(\boldsymbol{\theta}) = -t \ln \sigma(\mathbf{x}) - (1 - t) \ln(1 - \sigma(\mathbf{x}))$$


$$\ln \sigma(-x)$$

- σ is the logistic sigmoid function
- t is the ground-truth, 1 or 0

Negative sampling

$$p(\mathbf{w}_k | \mathbf{w}_t) = \sigma(\mathbf{v}'_{\mathbf{w}_k}{}^\top \mathbf{v}_{\mathbf{w}_t}) = \frac{1}{1 + \exp(-\mathbf{v}'_{\mathbf{w}_k}{}^\top \mathbf{v}_{\mathbf{w}_t})}$$

$k = 1, \dots, K + 1$

- The objective function for every word in the window

$$J^c = -\ln \sigma(\mathbf{v}'_0{}^\top \mathbf{v}_t) - \sum_{i=1}^K \mathbb{E}_{\mathbf{w}_i \sim P(\mathbf{w})} \ln \sigma(-\mathbf{v}'_i{}^\top \mathbf{v}_t)$$

- \mathbf{v}_t is the vector of the *input word* \mathbf{w}_t
- \mathbf{v}'_0 and \mathbf{v}'_i are the vectors of the positive and negative *output words* \mathbf{w}_0 and \mathbf{w}_i , respectively
- Max. prob that real words appear in the window
- Min. prob that random words appear in the window

Overall loss

For softmax (recall):

$$J(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^T J^t(\boldsymbol{\theta}) = -\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \ln p(\mathbf{w}_{t+j} | \mathbf{w}_t)$$

For negative sampling:

$$J(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^T J^t(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} J^c$$

where $J^c = -\ln \sigma(\mathbf{v}'_0{}^\top \mathbf{v}_t) - \sum_{i=1}^K \mathbb{E}_{\mathbf{w}_i \sim P(\mathbf{w})} \ln \sigma(-\mathbf{v}'_i{}^\top \mathbf{v}_t)$

How to define $P(\mathbf{w})$?

Subsampling of frequent words

- In very large corpora, the most frequent words can easily occur hundreds of millions of times (e.g., “in”, “the”, and “a”)
- Such words usually provide less information value than the rare words (e.g., “France”, “lavender”)
- During training,
 - The vectors of frequent words **do not change significantly** after training on several million examples
 - The vectors of rare words **change significantly** after training on a small number of examples
- We need to counter the **imbalance** between the rare and frequent words

Subsampling of frequent words

- Each word \mathbf{w}_i in the training set is **discarded** with probability (Mikolov et al., 2013)

$$P(\mathbf{w}_i) = 1 - \sqrt{t/f(\mathbf{w}_i)}$$

where $f(\mathbf{w}_i)$ is the frequency of \mathbf{w}_i in the corpus and t is a threshold, typically around 10^{-5}

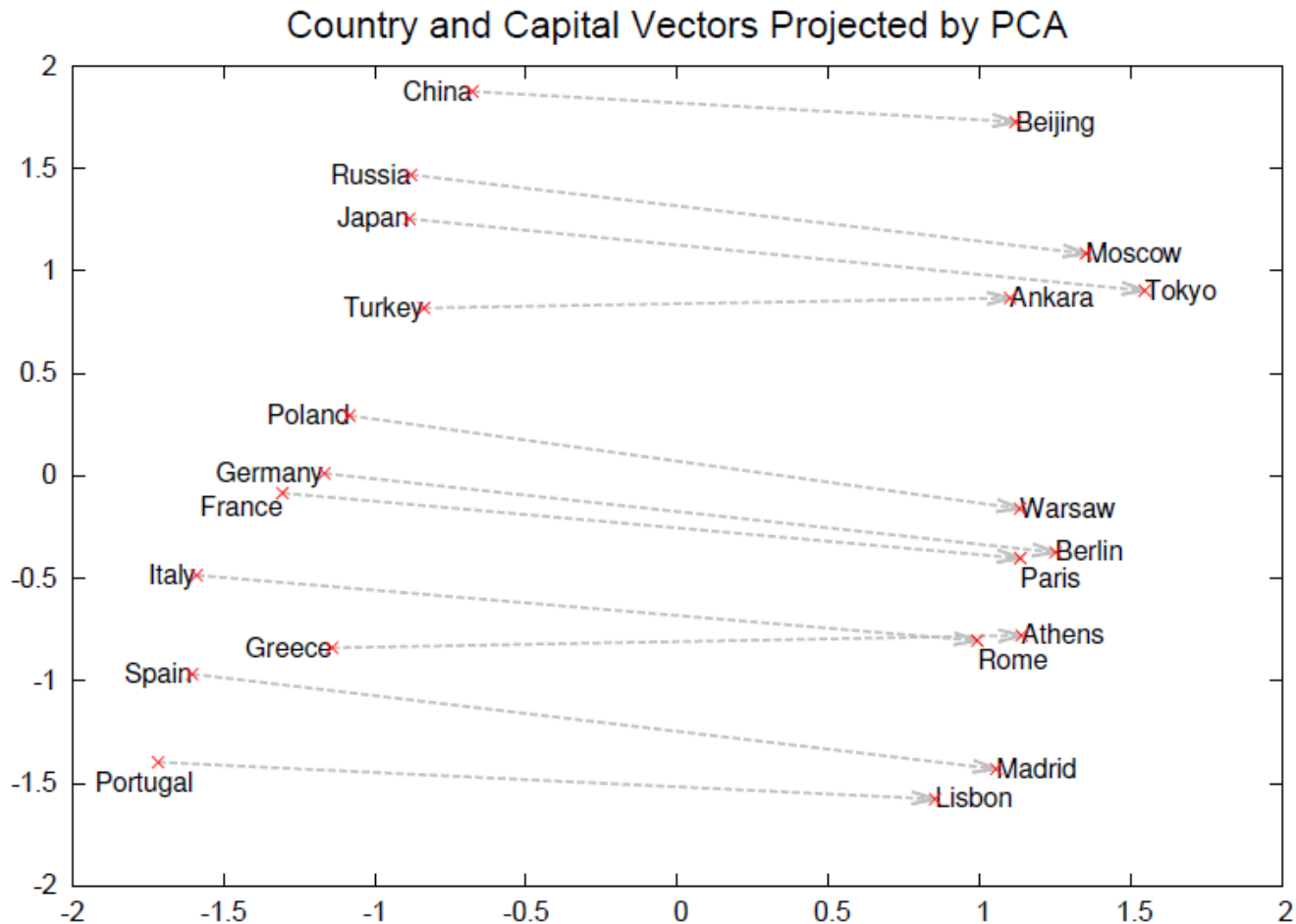
- But in the released codes, a different prob is used

$$P(\mathbf{w}_i) = \left(\sqrt{\frac{z(\mathbf{w}_i)}{0.001}} + 1 \right) \cdot \frac{0.001}{z(\mathbf{w}_i)}$$

where $z(\mathbf{w}_i)$ is the fraction of the total words in the corpus that are \mathbf{w}_i

Results

Mikolov et al., 2013



2D PCA projection of the 1000-dimensional Skip-gram vectors

Results

Mikolov et al., 2013

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

Extension*

- Skip gram is a direct prediction method
 - Scales with corpus size
 - Inefficient usage of statistics
- Historically, there are methods which are based on counts of words
 - LSA, HAL (Lund & Burgess), COALS (Rohde et al), Hellinger-PCA (Lebret & Collobert)
 - Fast training
 - Efficient usage of statistics
 - Primarily used to capture word similarity
- **Glove** combines the two kinds of methods
([Pennington et al. EMNLP 2014](#))

Outline

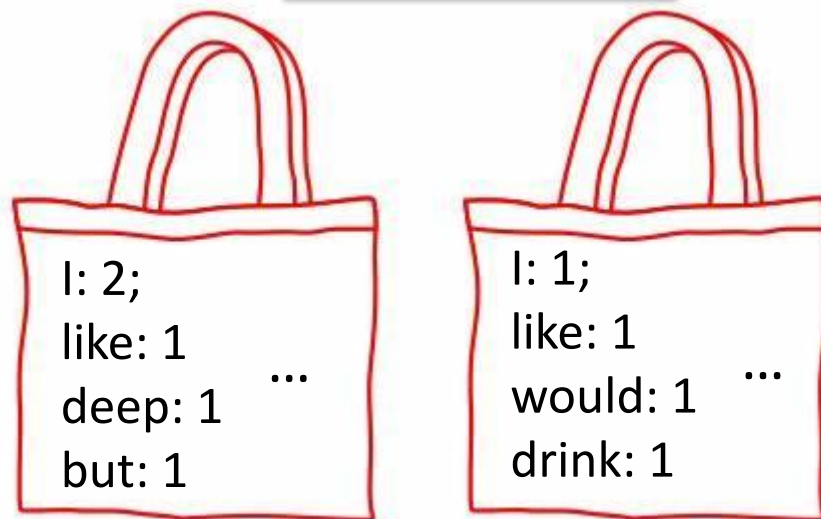
1. Typical tasks
2. Word representation
3. Text classification using NNs
4. Machine translation using NNs
5. Summary

Without word embedding, how do people represent sentences/documents?

Bag of words

“I like deep learning, but I do not have much time for doing homework.”

“I would like to drink a cup of coffee.”



Bag 1

Bag 2

Disadvantage: lack of order of the words

“white blood cells destroying an infection”

“an infection destroying white blood cells”

With word embedding, how do we classify sentences/documents?

“I like deep learning, but I do not have much time for doing homework.”



Use CNN?

“I would like to drink a cup of coffee.”



Use RNN?

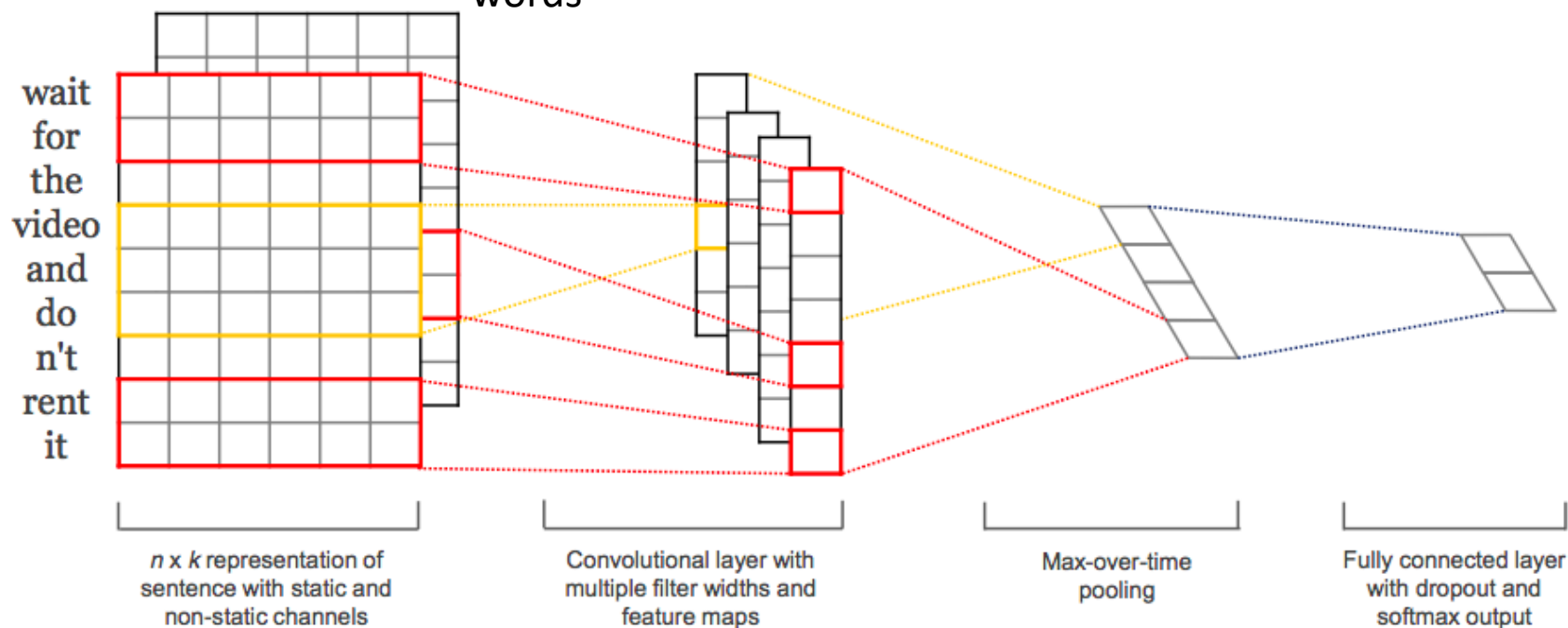


How?

Text classification using CNNs

Kim, 2014

Each kernel has dim. $h \times k$, where h is a window of words



Each word has a k -D real vector, and n words are concatenated together

Each feature vector is the result corresponding to a different h

Simple idea, great influence



Yoon Kim

MIT-IBM Watson AI Lab

Verified email at seas.harvard.edu - [Homepage](#)

[Machine Learning](#) [Natural Language Processing](#) [Deep Learning](#)

 FOLLOW

TITLE	CITED BY	YEAR
Convolutional Neural Networks for Sentence Classification Y Kim EMNLP 2014	9129	2014
Character-Aware Neural Language Models Y Kim, Y Jernite, D Sontag, AM Rush AAAI 2016	1426	2015
OpenNMT: Open-Source Toolkit for Neural Machine Translation G Klein, Y Kim, Y Deng, J Senellart, AM Rush ACL 2017 (System Demonstrations)	1135	2017
Sequence-Level Knowledge Distillation Y Kim, AM Rush EMNLP 2016	349	2016

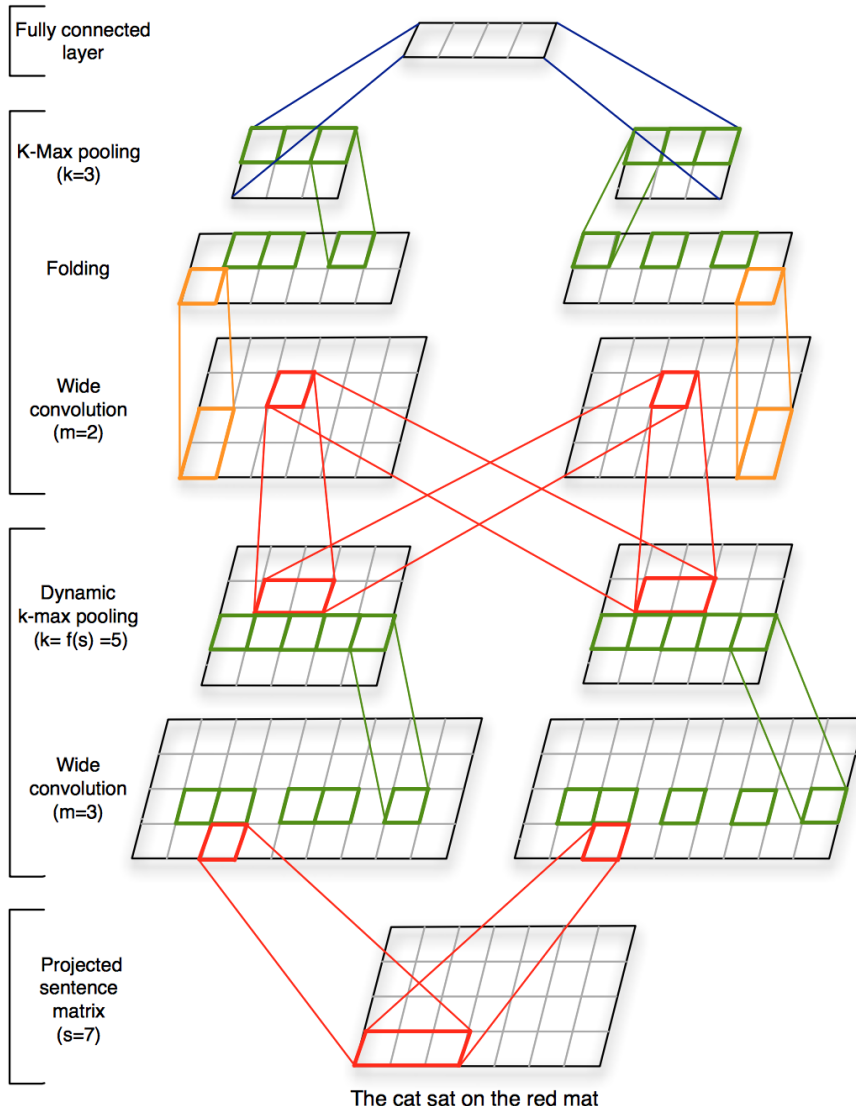
From Google Scholar, 2020/11/21

Any problem with this model?

- It's not deep enough
 - One conv layer and one pooling layer
- Features are not diverse enough
 - Each convolutional kernel results in a 1D feature map, i.e., a feature vector
 - The global max pooling on one feature vector, i.e., max pooling over all time, results in a scalar

A deeper model

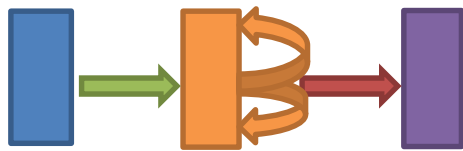
Kalchbrenner et al., 2014



- 1-D convolution along the time axis
- K-max pooling over time
 - What's the purpose?
- Folding: elementwise summation of two rows
 - A special kind of average pooling

Text classification using RNNs

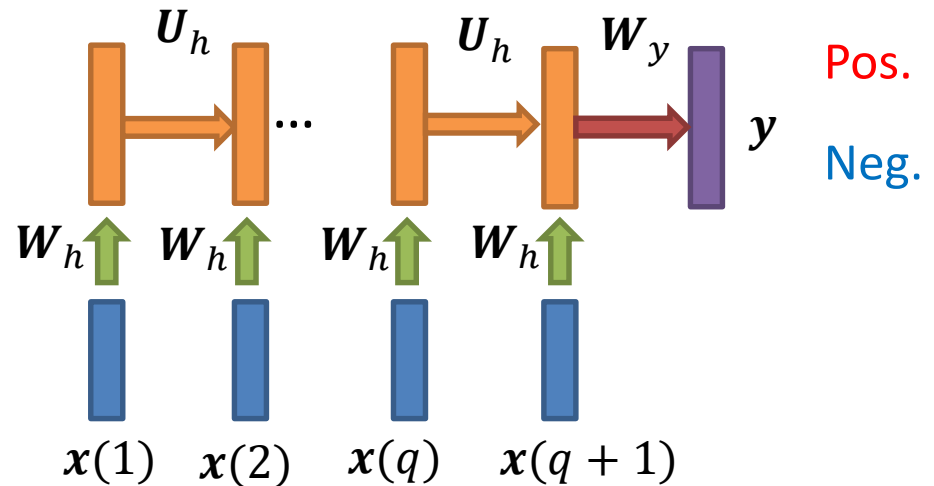
Unfold for the Elman network



Input x Hidden h Output y

$$\begin{aligned} h(t) &= \sigma_h(W_h x(t) + U_h h(t-1) + b_h) \\ y(t) &= \sigma_y(W_y h(t) + b_y) \end{aligned}$$

- x is time-varying
- Label r is only present at the last step



- LSTM or GRU can be used
- Bidirectional RNN can be used
- See homework

Deep learning is so cool.

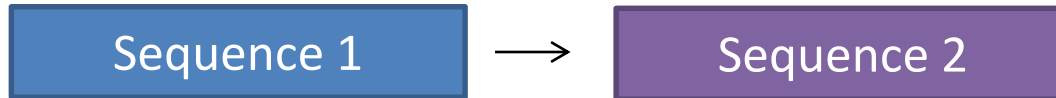
How noisy the teacher's English is!

Outline

- Typical tasks
- Word representation
- Text classification using NNs
- Machine translation using NNs
- Summary

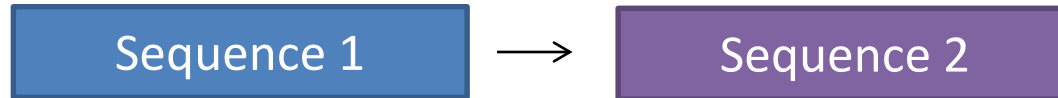
From Abigail See's slides
(Stanford University)

Sequence-to-sequence learning



- Usually it involves two RNNs: **encoder** and **decoder**
- Many NLP tasks can be phrased as sequence-to-sequence:
 - **Machine translation** (French → English)

Sequence-to-sequence learning

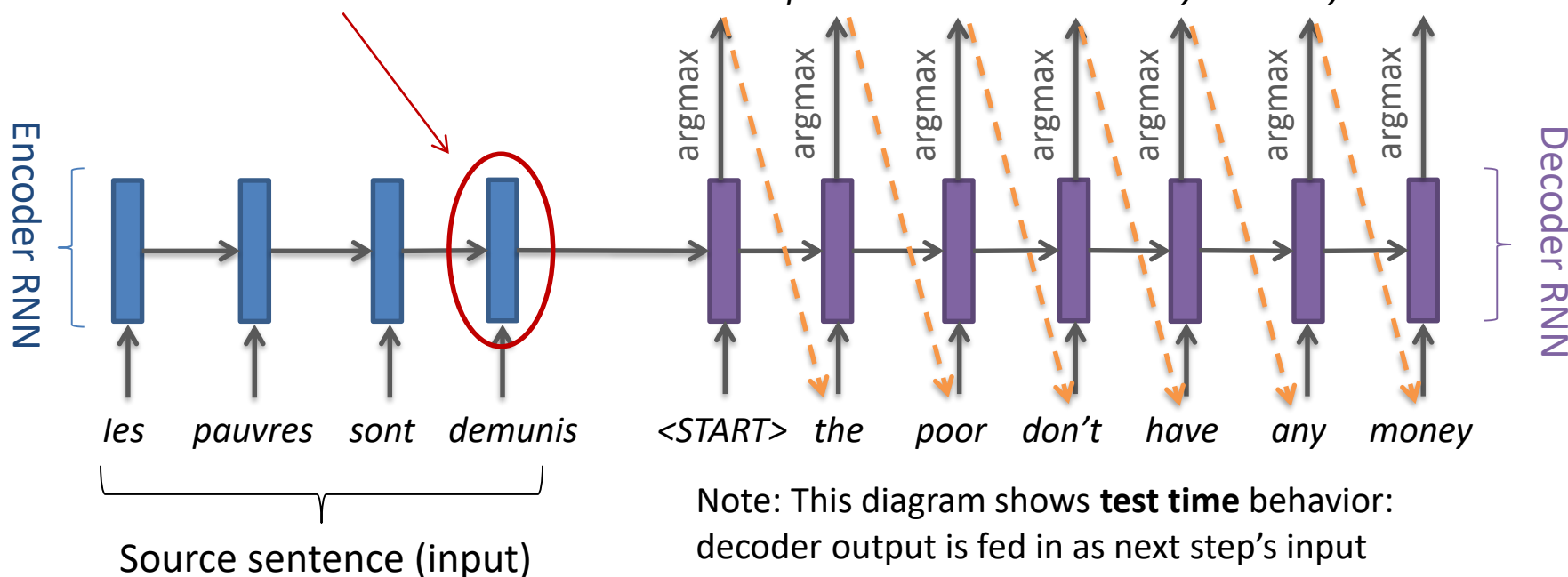


- Usually it involves two RNNs: **encoder** and **decoder**
- Many tasks can be phrased as sequence-to-sequence:
 - **Machine translation** (French → English)
 - **Summarization** (long text → short text)
 - **Dialogue** (previous utterances → next utterance)
 - **Parsing** (input text → output parse as sequence)
 - **Code generation** (natural language → Python code)
 - **Melody generation** (one musical phrase → next phrase)
 - **Speech recognition** (sound → text)

Neural machine translation (NMT)

Sutskever et al., 2014

Encoding of the source sentence.
Provides **initial hidden state** for
Decoder RNN



- Encoder RNN produces an encoding of the source sentence
- Decoder RNN is a Language Model that generates target sentence conditioned on encoding.

If you use the CE loss to train the model, which of the following should NOT be used to represent words?

- ☐ A One-hot representation input to the encoder
- ☐ B Pretrained word embeddings input to the encoder
- ☐ C One-hot representation output by the decoder
- ☒ D Pretrained word embeddings output by the decoder

Submit

Encoder and decoder

- For the encoder, one can use either pretrained word embedding, e.g., **word2vec**, or **one-hot** representation
- For the decoder, the **one-hot** representation is used
 - We need a prob for each reference word to define the objective function
 - **Softmax** function is usually used as the output
 - What if the dictionary is very large?
- The encoder RNN and decoder RNN are often **different**, and **deep RNNs** can be used

Can the dictionaries for the encoder and the decoder be different?

☒ A Yes

☐ B No

Submit

In which tasks, can the dictionaries for the encoder and the decoder be the same?

- ☐ A Machine translation
- ☒ B Text summarization
- ☒ C Story generation
- ☒ D Dialog

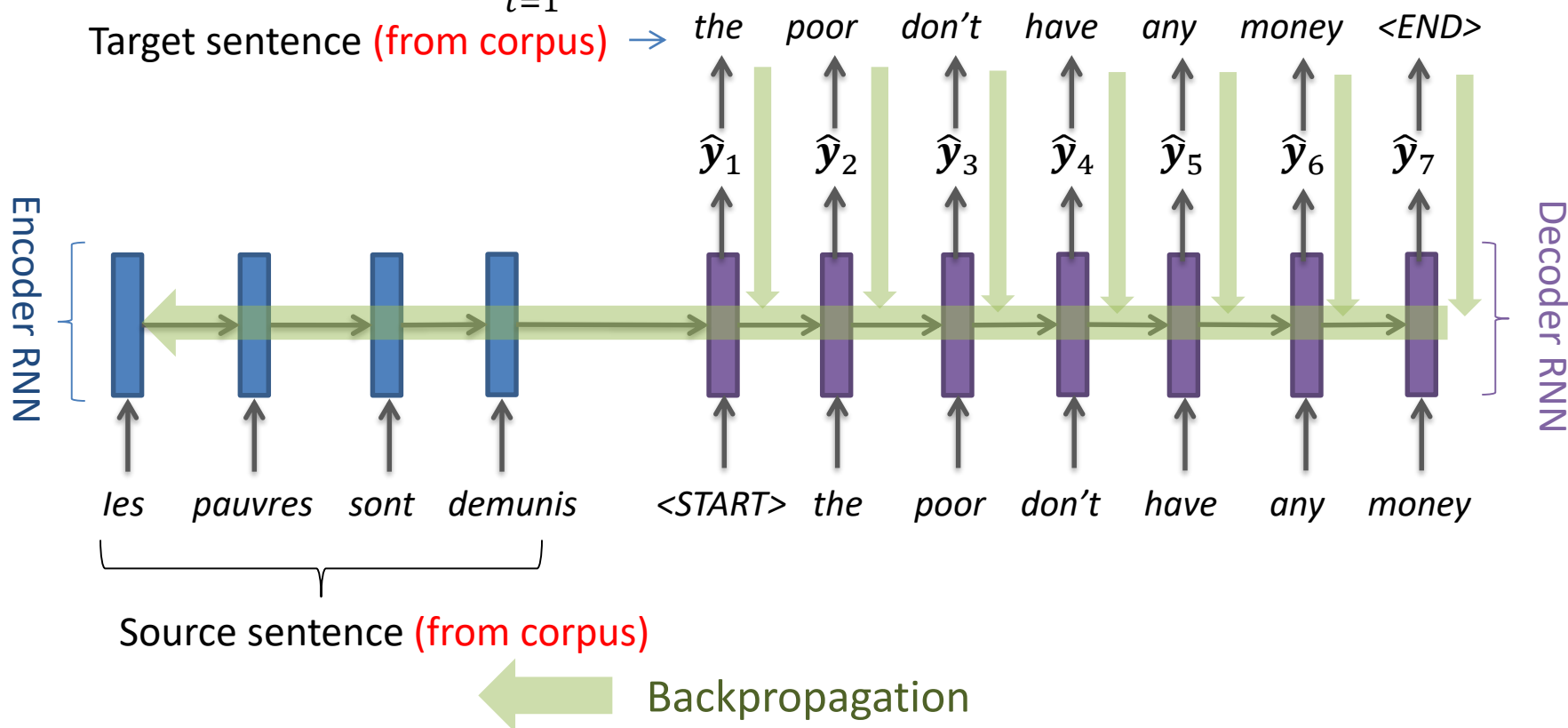
Submit

Conditional Language Model

- The sequence-to-sequence model is an example of a Conditional Language Model
 - **Language Model** because the decoder is predicting the next word of the target sentence y
 - **Conditional** because its predictions are *also* conditioned on the source sentence x
- NMT directly calculates $P(Y|X)$, where $Y = [y_1, \dots, y_T]$, $X = [x_1, \dots, x_{T'}]$
$$P(Y|X) = P(y_1|X)P(y_2|y_1, X)P(y_3|y_1, y_2, X) \cdots \underbrace{P(y_T|y_1, \dots, y_{T-1}, X)}_{\text{Probability of next target word, given target words so far and source sentence } X}$$
- How to train an NMT?
 - You need a big parallel corpus...

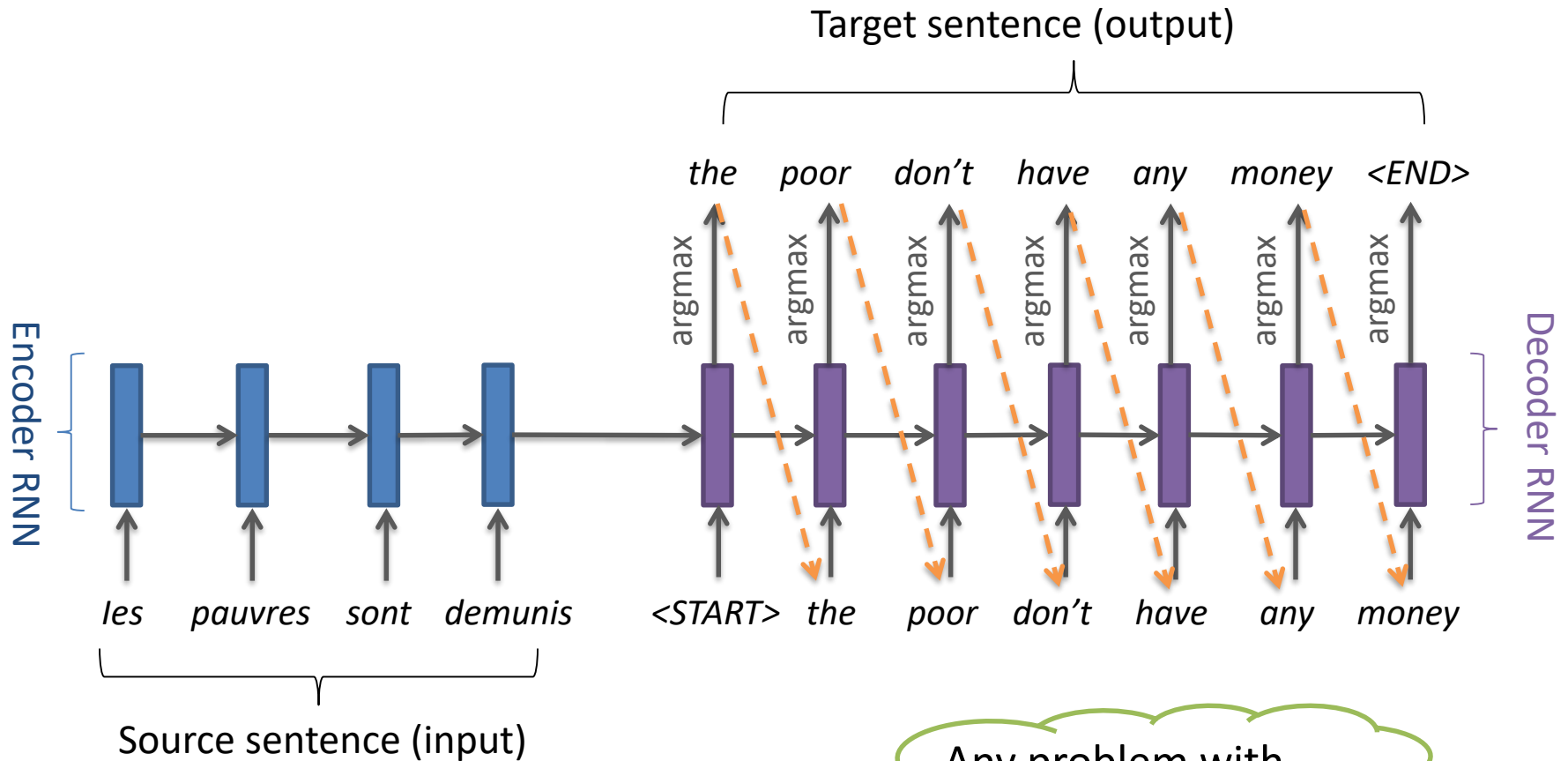
Training an NMT system

$$J = \frac{1}{T} \sum_{t=1}^T J_t = \underbrace{J_1}_{\text{= negative log prob of "the"}} + J_2 + J_3 + \underbrace{J_4}_{\text{= negative log prob of "have"}} + J_5 + J_6 + \underbrace{J_7}_{\text{= negative log prob of <END>}}$$

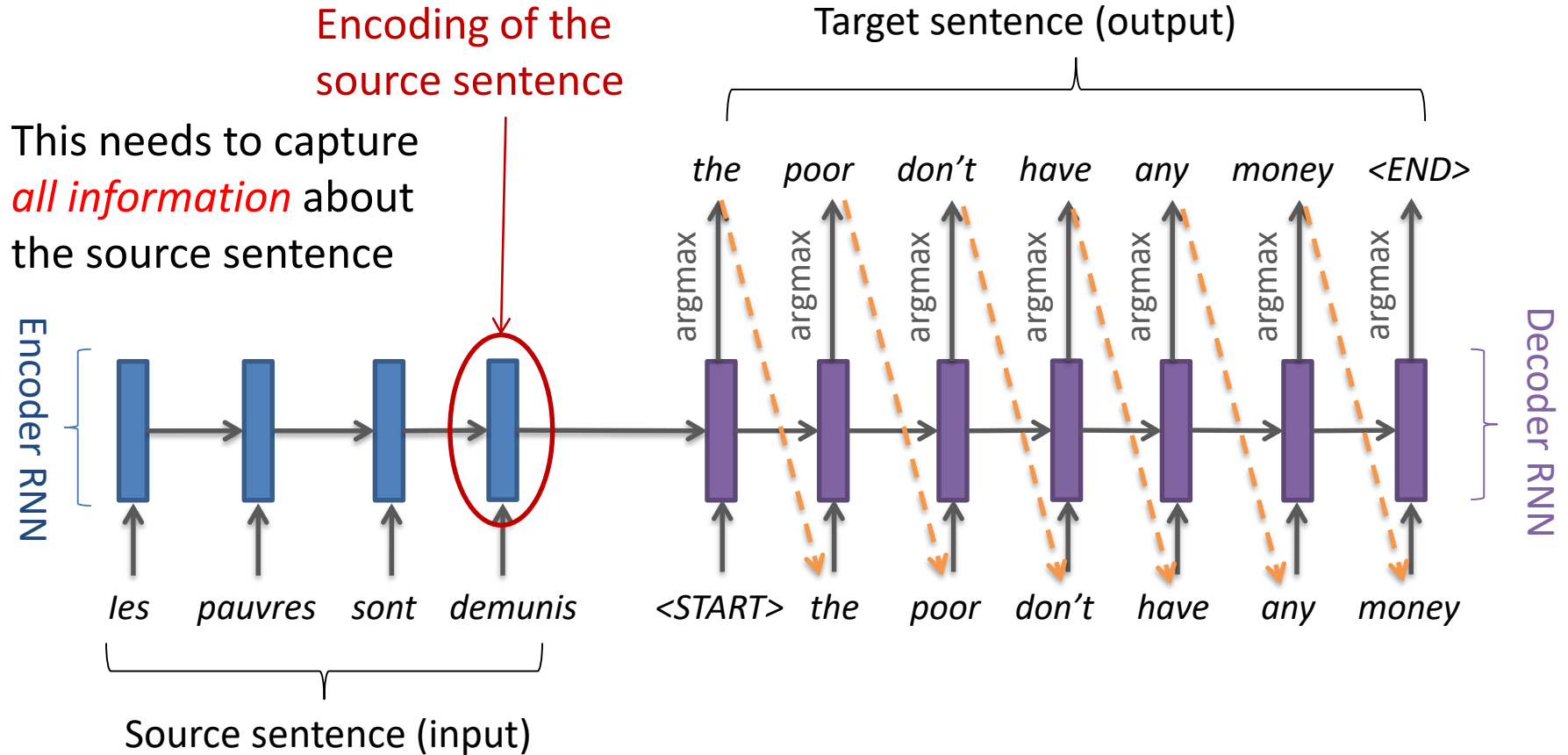


Seq2seq is optimized as a **single system**

The bottleneck problem



The bottleneck problem



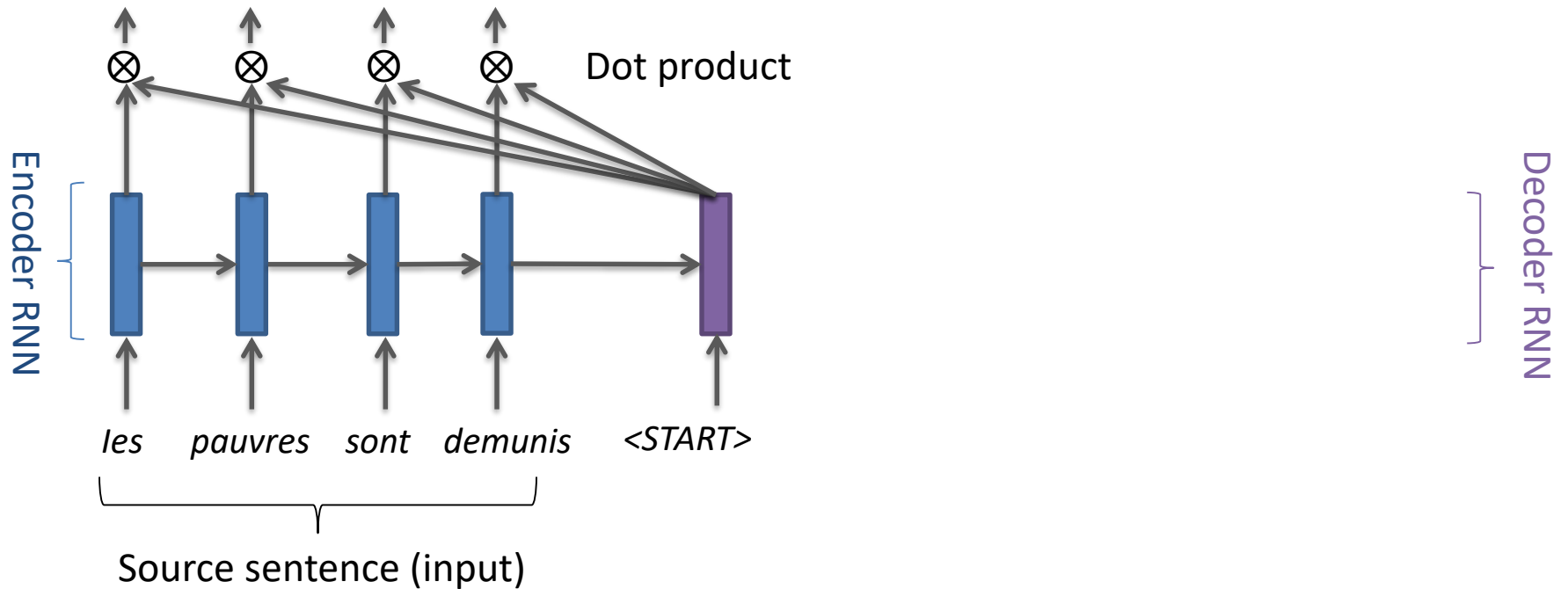
Attention

- **Attention** provides a solution to the bottleneck problem.
- Core idea: on each step of the decoder, *focus on a particular part* of the source sequence



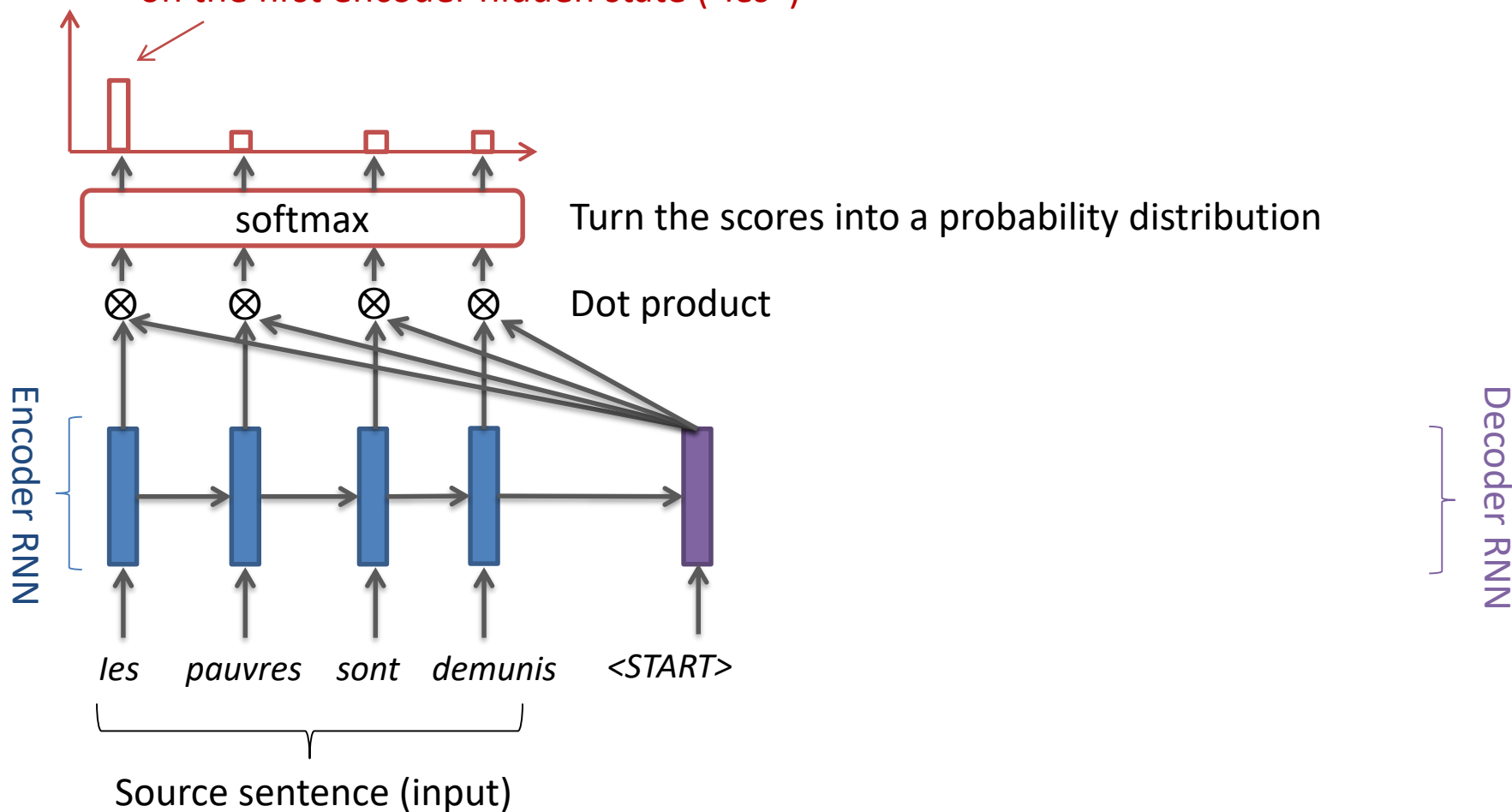
- There are many forms of attention
- I will show a simple example

Seq2seq with attention

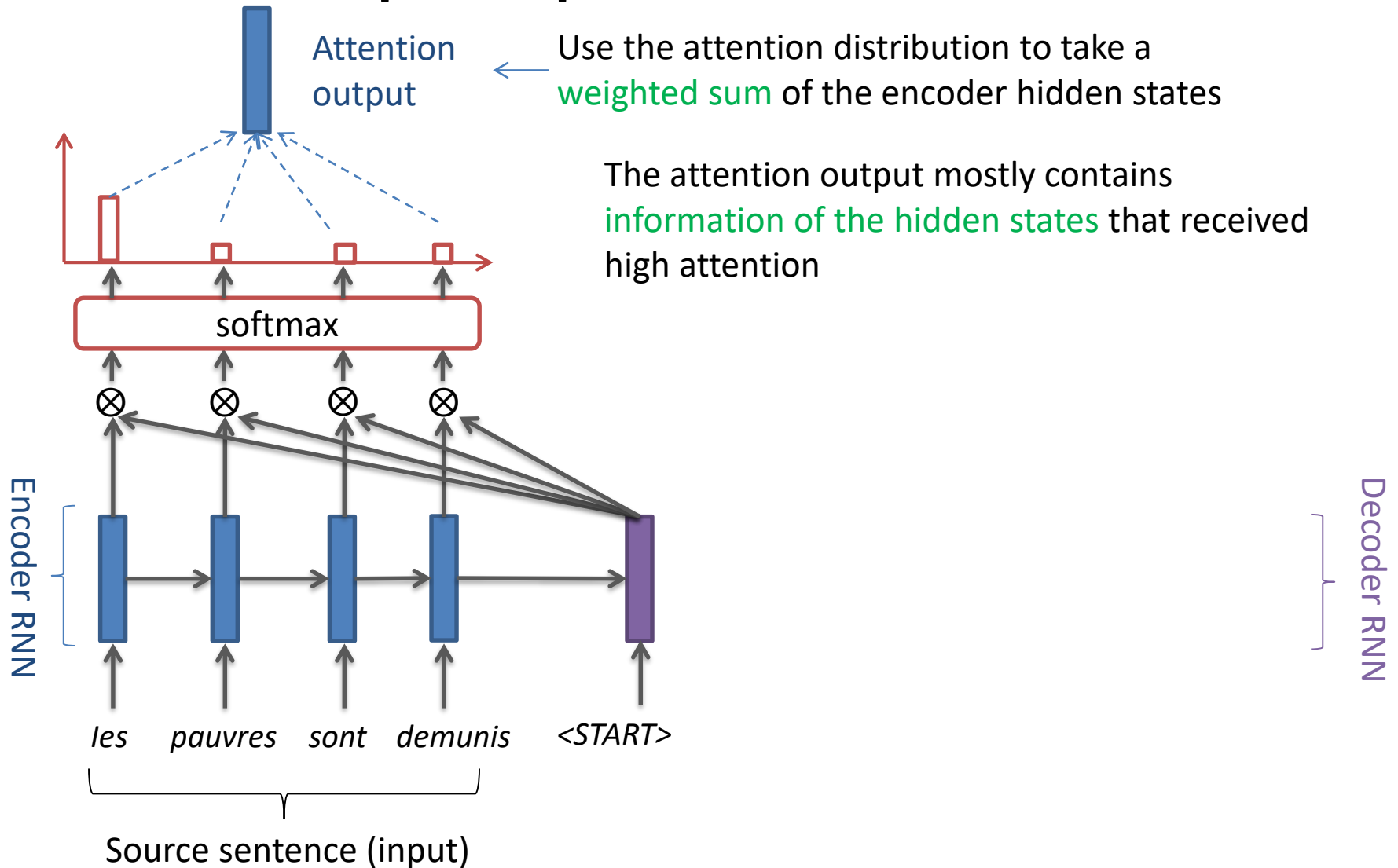


Seq2seq with attention

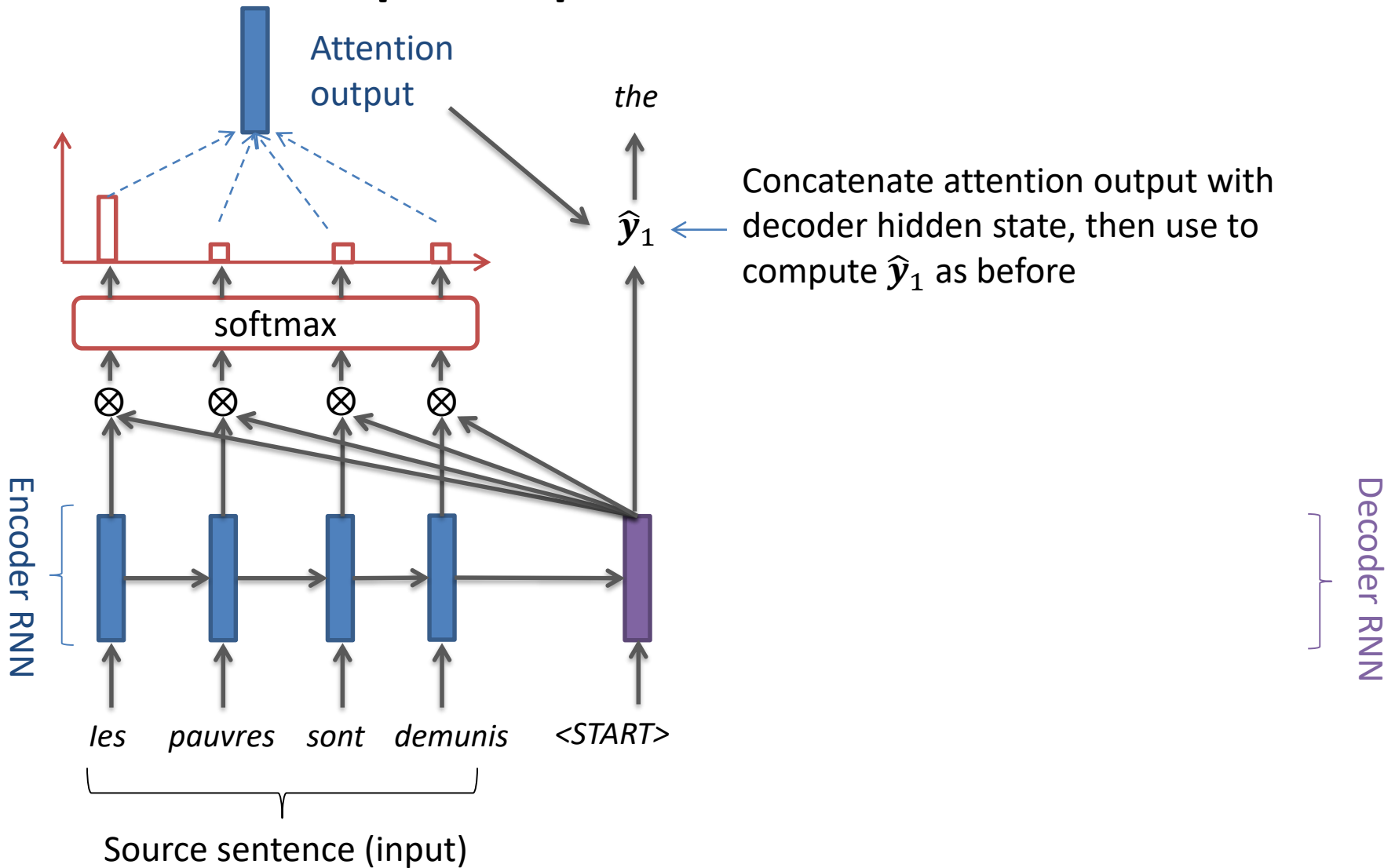
On this decoder timestep, we're mostly focusing on the first encoder hidden state ("les")



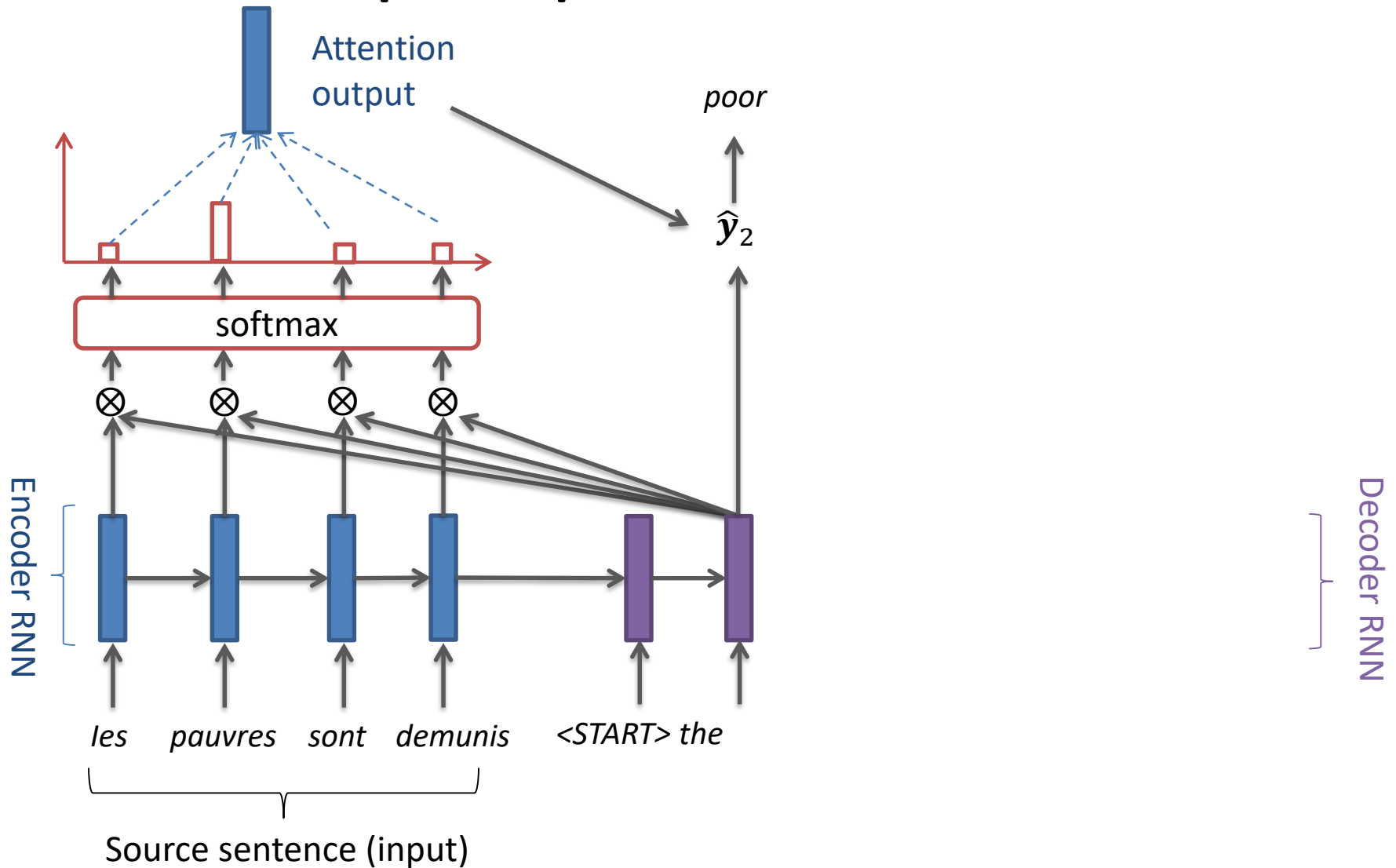
Seq2seq with attention



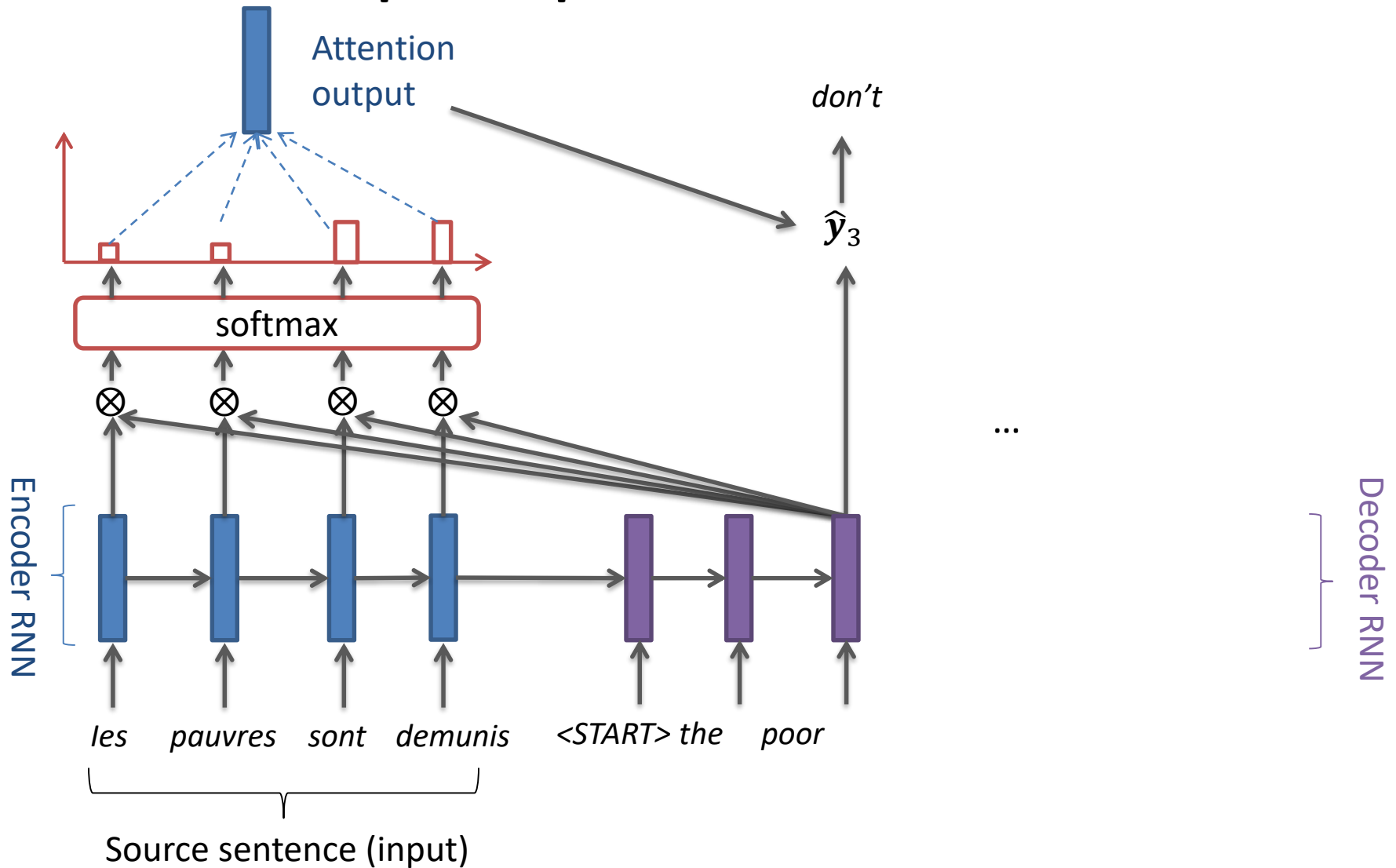
Seq2seq with attention



Seq2seq with attention



Seq2seq with attention



Formulation

The encoder's hidden states: $\mathbf{h}_1, \dots, \mathbf{h}_N \in R^h$

At time step t , the decoder's hidden state $\mathbf{s}_t \in R^h$

- We get the attention scores \mathbf{e}^t for this step:

$$\mathbf{e}^t = [\mathbf{s}_t^\top \mathbf{h}_1, \dots, \mathbf{s}_t^\top \mathbf{h}_N] \in R^N$$

- Take softmax to get the attention distribution for this step

$$\boldsymbol{\alpha}^t = \text{softmax}(\mathbf{e}^t) \in R^N$$

- Calculate the attention output

$$\mathbf{a}_t = \sum_{i=1}^N \alpha_i^t \mathbf{h}_i \in R^h$$

- Finally, concatenate the attention output with the decoder hidden state and proceed as in the non-attention seq2seq model

$$[\mathbf{a}_t; \mathbf{s}_t] \in R^{2h}$$

Outline

1. Typical tasks
2. Word representation
3. Text classification using NNs
4. Machine translation using NNs
5. Summary

Summary of this lecture

Knowledge

1. Typical tasks

Tagging and parsing

Text generation

Text/document classification

Sentiment analysis

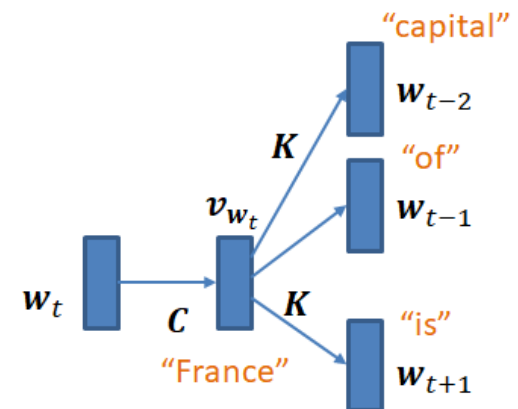
Machine translation

Question answering

2. Word representation

counts	I	like	deep	learning	NLP	enjoy	flying
I	0	2	0	0	0	1	0
like	2	0	1	0	1	0	0
deep	0	1	0	1	0	0	0
learning	0	0	1	0	0	0	0
NLP	0	1	0	0	0	0	0
enjoy	1	0	0	0	0	0	1
flying	0	0	0	0	0	1	0

Co-occurrence matrix

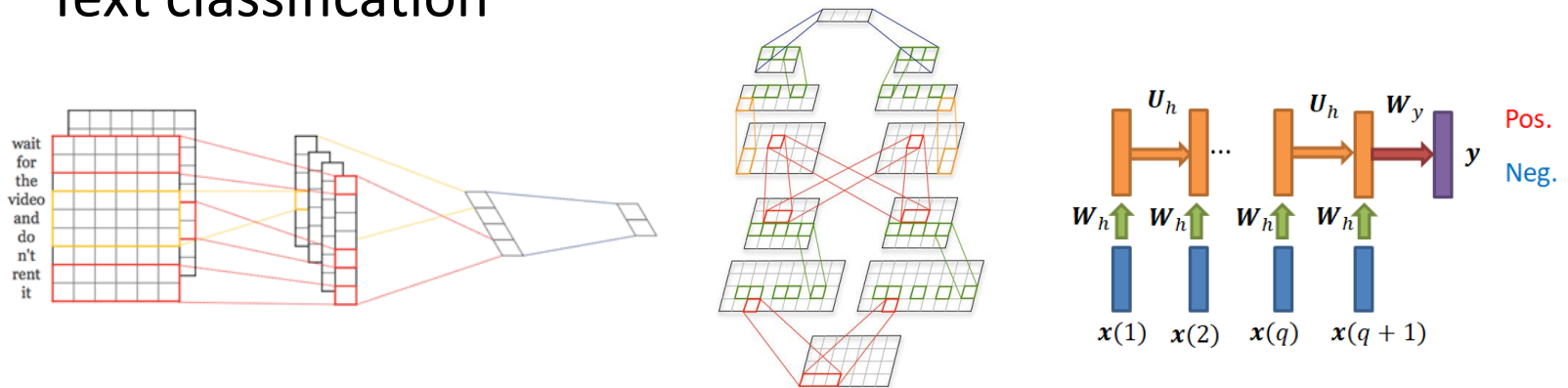


Dense vectors

Summary of this lecture

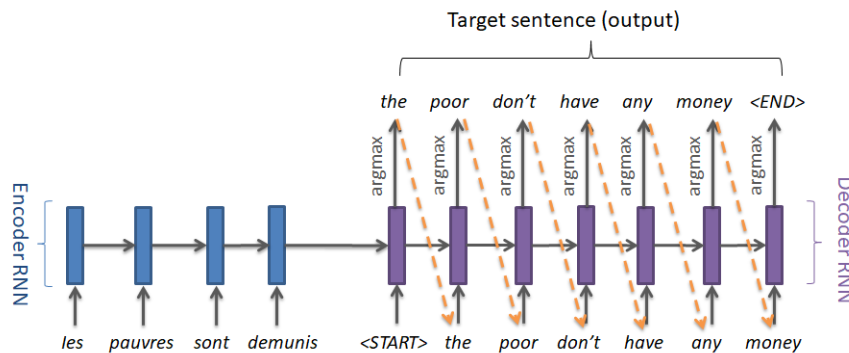
Knowledge

3. Text classification

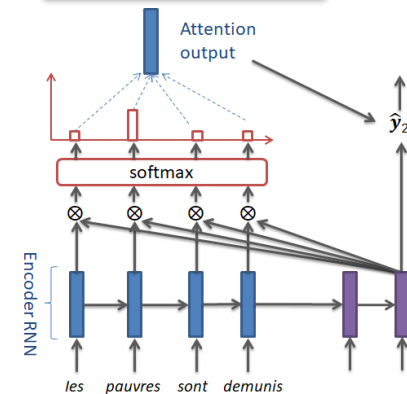


4. Machine translation

Encoder-decoder



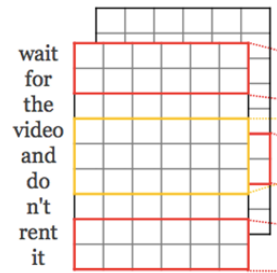
Attention



Summary of this lecture

Capability and value

- Comparative thinking



- Simple idea, great influence

Convolutional Neural Networks for Sentence Classification

Y Kim
EMNLP 2014

9129

2014

Recommended reading

- Kim (2014)
Convolutional neural networks for sentence classification
[arXiv preprint arXiv:1408.5882](#)
- Bengio, Ducharme, Vincent, Jauvin (2003)
A neural probabilistic language model
[Journal of Machine Learning Research](#)
- Mikolov, et al. (2013)
Distributed representations of words and phrases and their compositionality
[NeurIPS](#)
- Sutskever, Vinyals, Le (2014)
Sequence to Sequence Learning with Neural Networks
[NeurIPS](#)