Course number: 80240743

Deep Learning

Xiaolin Hu (胡晓林) & Jun Zhu (朱军)
Dept. of Computer Science and Technology
Tsinghua University

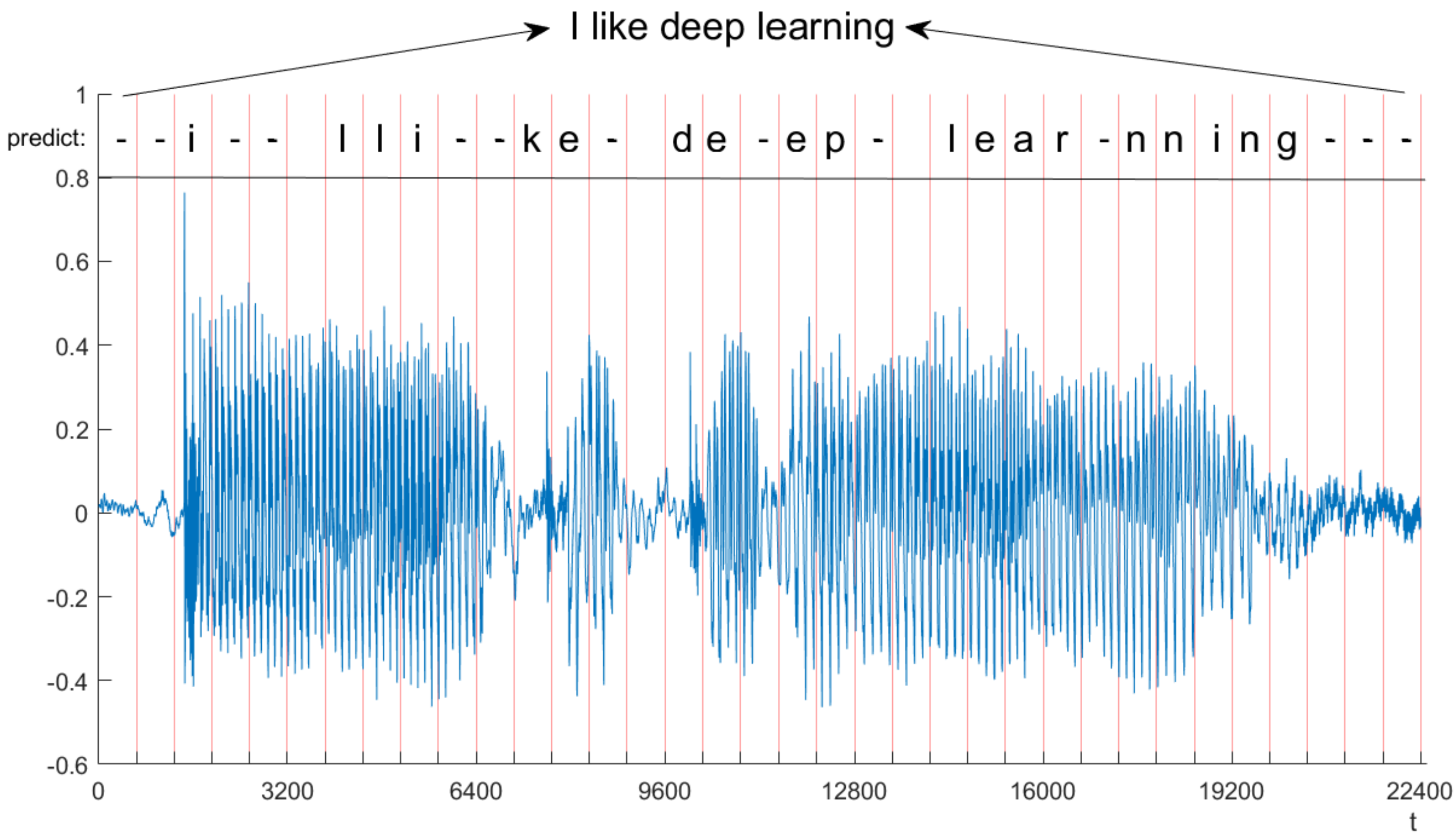
Topic 7: NNs for sequential data processing

Xiaolin Hu
Dept. of Computer Science and Technology
Tsinghua University

Outline

- Speech recognition
- Natural language processing
  – Typical tasks
  – Word representation
  – Text classification using NNs
  – Machine translation using RNNs

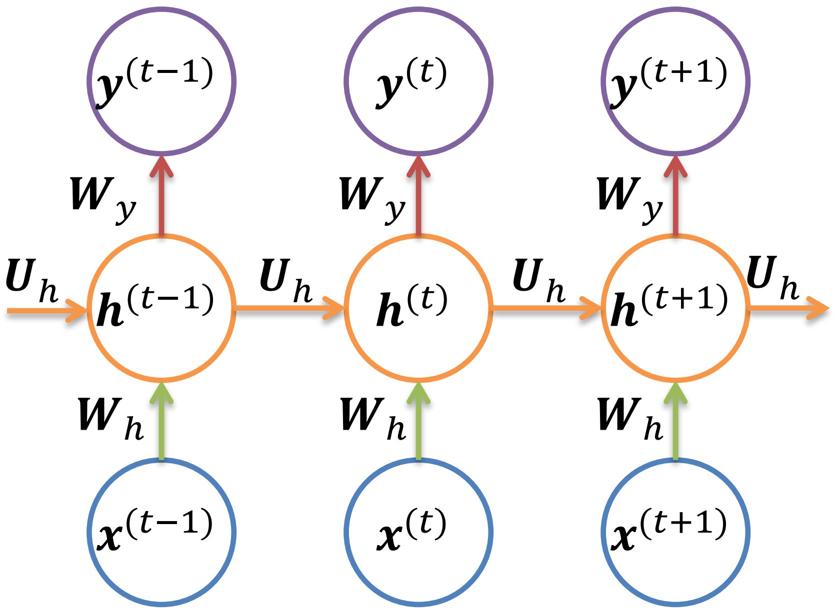
Speech recognition



A demo from Microsoft

RNN setting

There are reference $\boldsymbol{r}^{(t)}$ at each time step



- Suppose there is no act fun
  $$\boldsymbol{y}^{(t)} = \boldsymbol{W}_y \boldsymbol{h}^{(t)} + \boldsymbol{b}_{\boldsymbol{y}}$$
- At every time step, use a softmax fun to predict an output, e.g., a phoneme or a blank
  - There are $K + 1$ classes at time t, where K is the number of phonemes, usually < 100

$$P\left(\mathrm{r}_k^{(t)} = 1 | \boldsymbol{h}^{(t)}\right) = \frac{\exp\left(y_k^{(t)}\right)}{\sum_{k=1}^{K+1} \exp\left(y_k^{(t)}\right)}$$

A random variable, not reference value

Objective function

- Maximize the prob of reference class at all time steps

$$\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} \ln P\left(\mathrm{r}_k^{(t)} = r_k^{(t)} | \boldsymbol{h}^{(t)}\right)$$

Reference value which is 1

- This is equivalent to minimizing the cross-entropy error
  - The cross-entropy error at time t

$$-\sum_{k=1}^{K+1} r_k^{(t)} \ln p\left(\mathrm{r}_k^{(t)} = 1 \middle| \boldsymbol{h}^{(t)}\right) = -\ln p\left(\mathrm{r}_k^{(t)} = 1 \middle| \boldsymbol{h}^{(t)}\right)$$

where k satisfies $r_k^{(t)} = 1$ because other elements of $\boldsymbol{r}^{(t)}$ are zeros

  - Sum the cross-entropy error over time

$$-\sum_{t=1}^{T} \ln p\left(\mathrm{r}_k^{(t)} = 1 \middle| \boldsymbol{h}^{(t)}\right)$$

Objective function

- Optimizing the objective function in the previous slide will result in T outputs

"learning"

ϕ ϕ /l/ /ə/ /ə/ /ə/ ϕ ϕ /n/ /i/ /i/ /ŋ/ ϕ

ϕ is blank

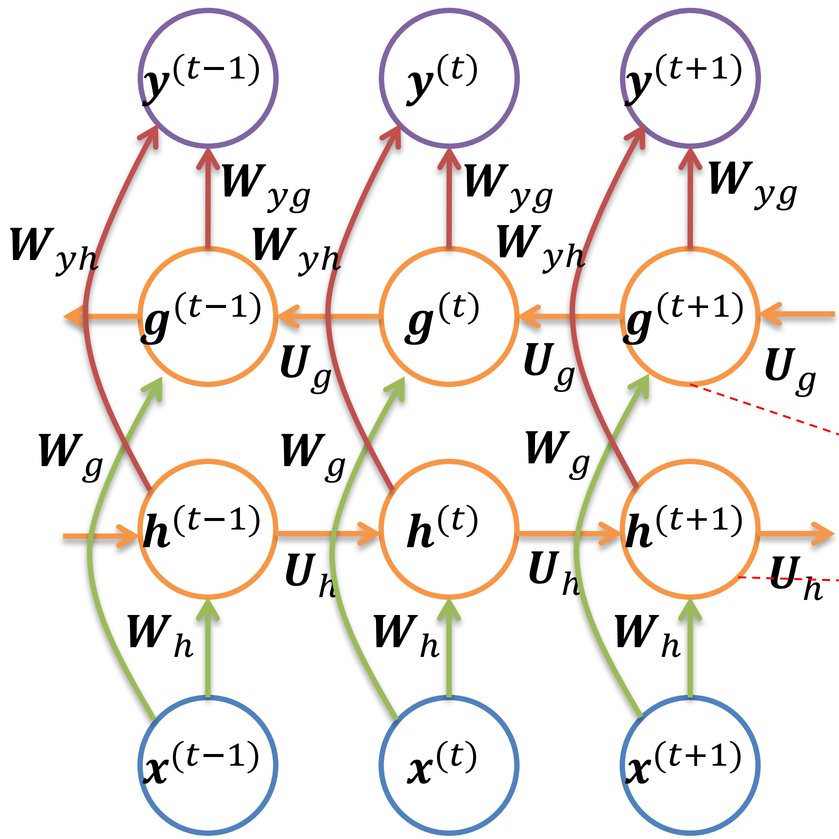
/l/ /ə/ /r/ /n/ /i/ /ŋ/

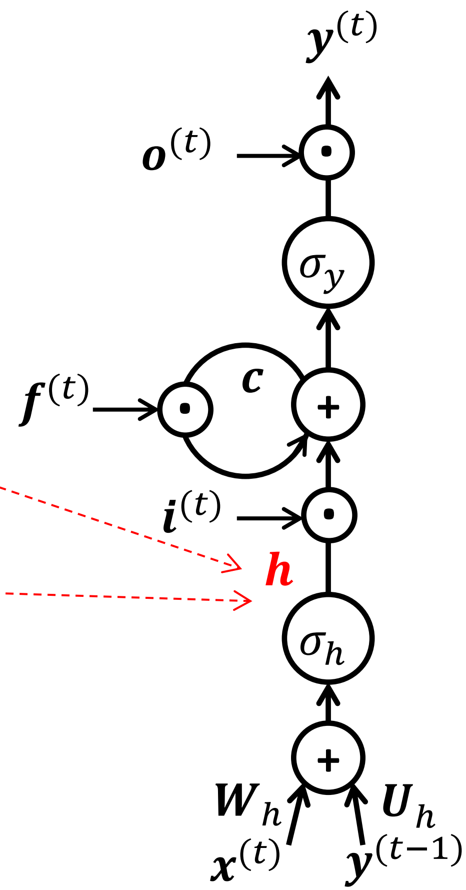
- But the reference sequence may be shorter or longer than it
- How to measure the difference and minimize the difference?
  - "edit-distance" is introduced: the minimum number of insertions, substitutions and deletions required to change seq p into seq q
  - A method called "Connectionist Temporal Classification (CTC)" is usually used (Graves et al., 2006)

Use bidirectional LSTM

Graves et al., 2013

Each of the two RNNs are LSTMs

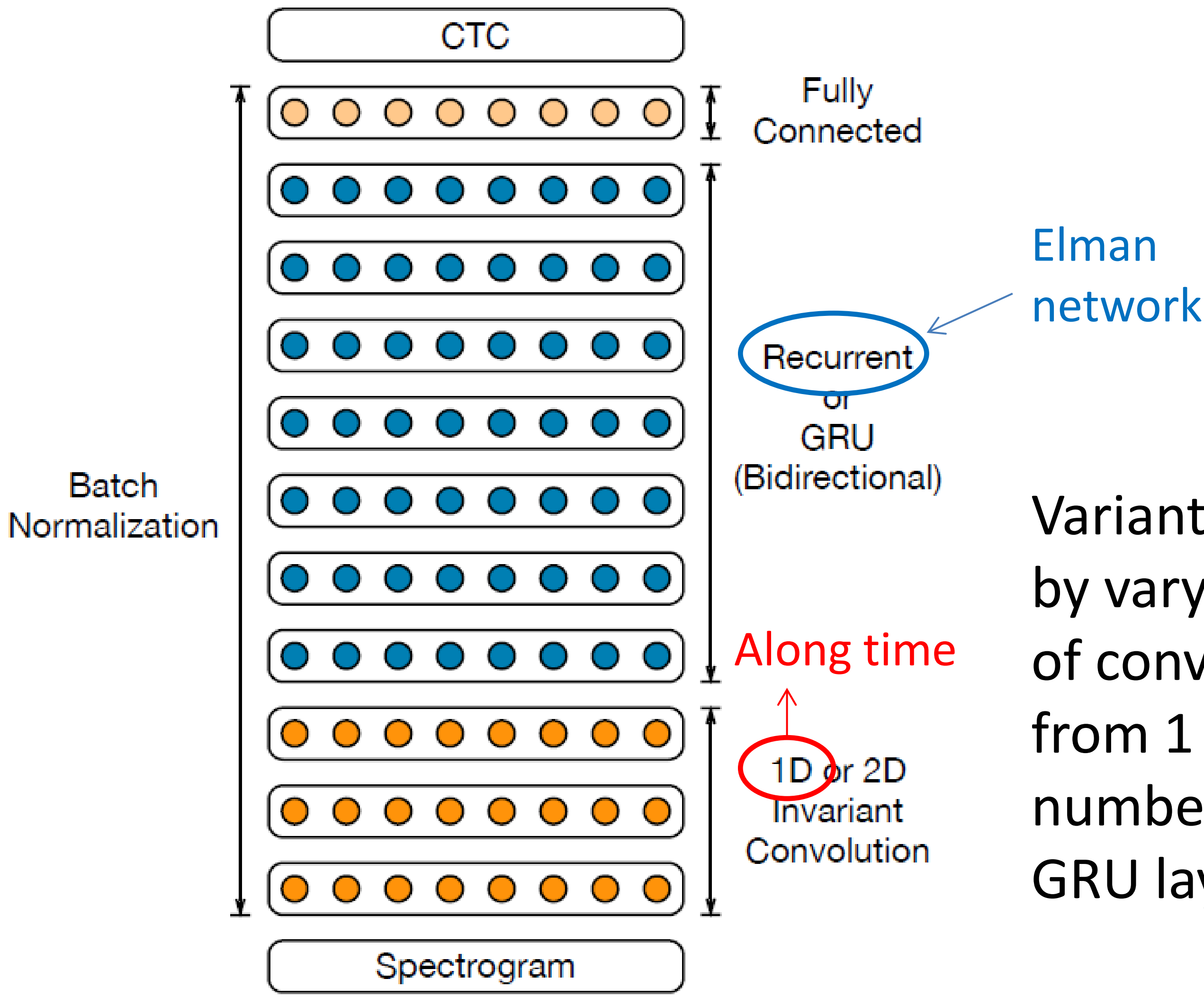




Define the obj fun the same as before based on: $y^{(t)} = W_{yh} h^{(t)} + W_{yg} g^{(t)} + b_y$

Deep RNN – Deep Speech 2 by Baidu

Amodei et al., 2015



Variants were explored by varying the number of convolutional layers from 1 to 3 and the number of recurrent or GRU layers from 1 to 7

Data preprocessing

- Usually, the input to the model is not raw wav signal, but the spectral-temporal signal
  - In (Graves et al., 2013), the audio data was encoded using a Fourier-transform-based filter-bank with 40 coefficients (plus energy) distributed on a mel-scale, together with their first and second temporal derivatives
  - Each input vector was therefore size 123
  - The data were normalized so that every element of the input vectors had zero mean and unit variance over the training set

Benchmark datasets

- Benchmark datasets
  - TIMIT, small
  - Switchborad, 260 hours
  - LibriSpeech, 1000 hours
  - CHiME, with various enviroment noises
- Many benchmark datasets are only used for testing, and you need to use your own training set
  - Deep speech 2 the English system was trained on 11,940 hours of English speech, while the Mandarin system was trained on 9,400 hours. Data synthesis was used to further augment the data.
- Researchers tend to opensource their models but do not release the training set
  - This makes the evaluation of different models difficult

State of the art

- In 2017, Microsoft announced that their speech recognition system has achieved 5.1% word error rate (WER) on Switchboard
  - This is average level of professional transcribers
- However, all current models perform poorly on noisy data
  - The lowest WER in *CHiME 2018 Challenge* is about 50%. See results here: http://spandh.dcs.shef.ac.uk/chime_challenge/results.html

Outline

- Speech recognition
- Natural language processing
  - Typical tasks
  - Word representation
  - Text classification using NNs
  - Machine translation using RNNs

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

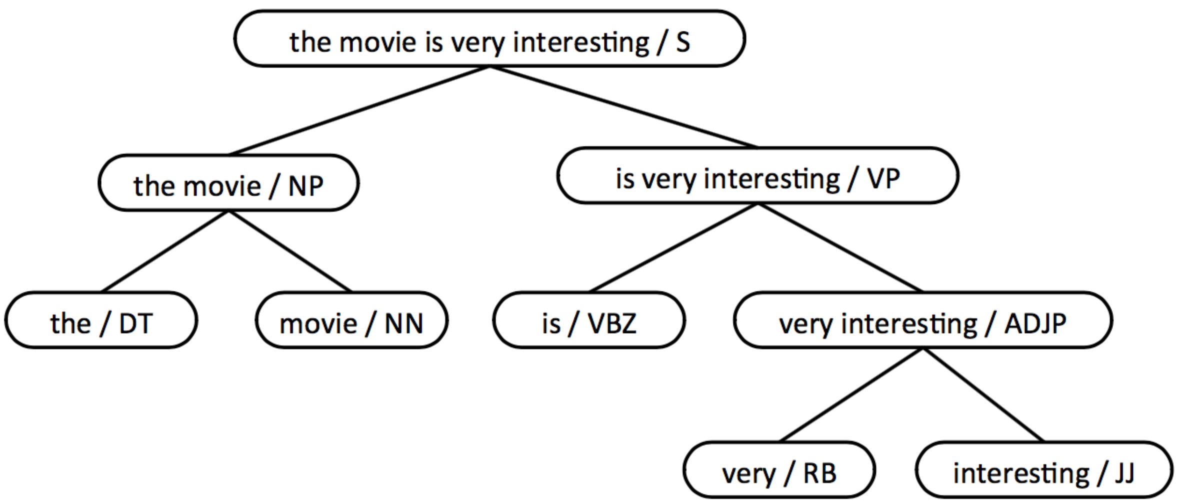
Typical tasks

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

Tagging and parsing

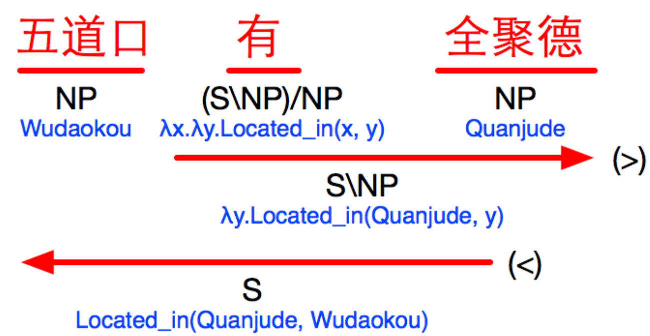
Part-of-speech (POS) Tagging

The	movie	is	very	interesting
DT	NN	VBZ	RB	JJ



Syntactic Parsing

Semantic Parsing



Slides from Prof. Minlie Huang

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

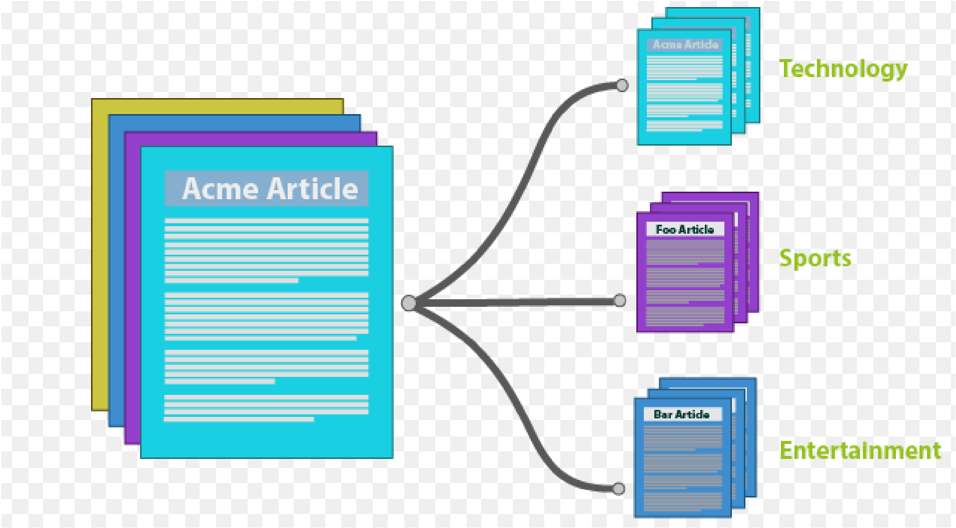
Slides from Prof. Minlie Huang

Text/document classification

- Sentiment classification

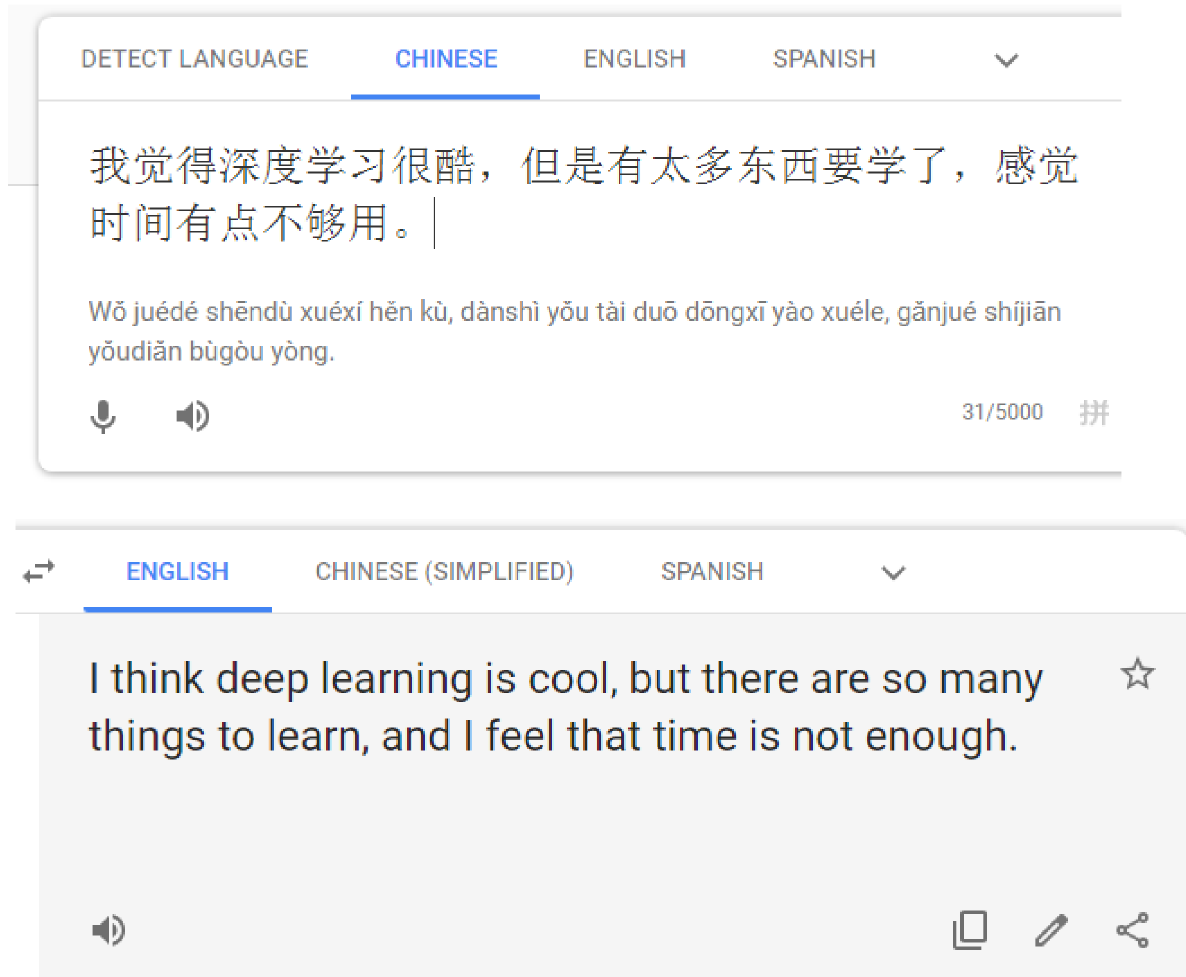
Deep learning is so cool.	Positive
How noisy the teacher's English is!	Negative
I attended the lecture today.	Neutral

- Document classification



Machine translation

https://translate.google.com

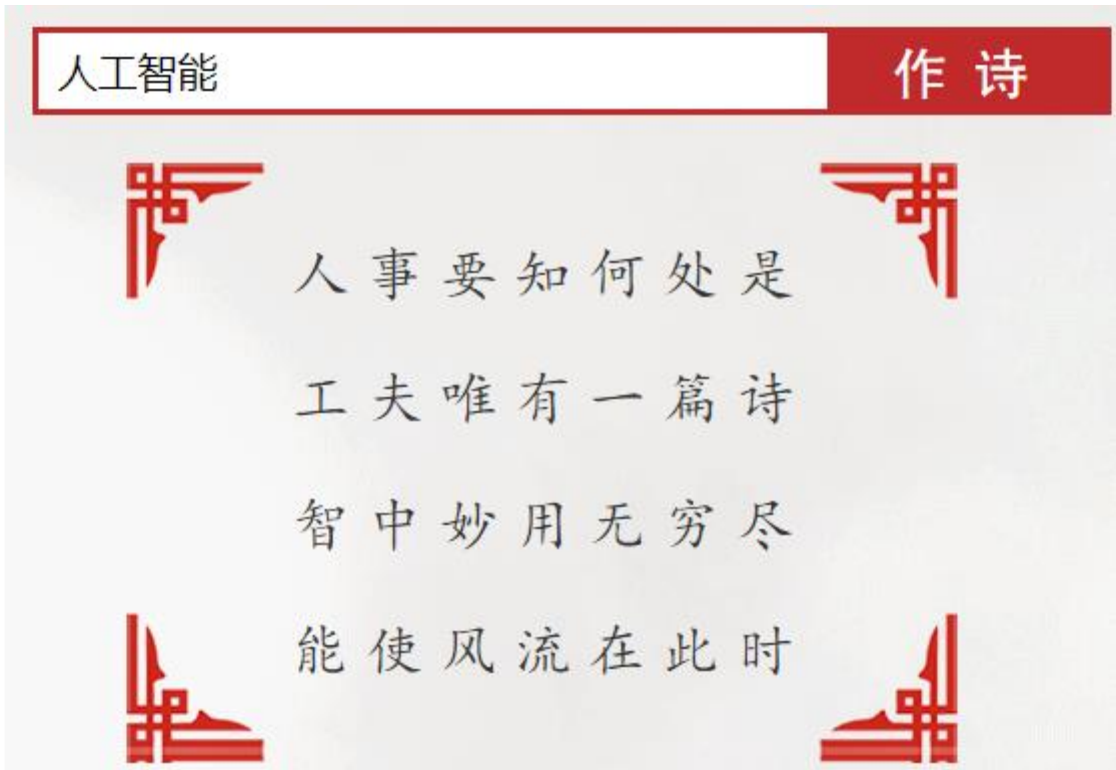


Chinese couplet generation

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

Chinese ancient poetry generation

http://jiuge.thunlp.org/



Outline

- Speech recognition
- Natural language processing
  - Typical tasks
  - Word representation
  - Text classification using NNs
  - Machine translation using RNNs

Word representation

- One-hot representation

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

  Dimensionality: the vocabulary size $|\boldsymbol{V}|$ (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

- Use cooccurrence of the word with other words to represent that word
- 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!

Slide from Richard Socher

How to make neighbors represent words

- Answer: use a cooccurrence matrix
- Two options: full document vs windows
  - Word-document cooccurrence 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

Slide from Richard Socher

Window based cooccurence 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	1	0	1	0
like	2	0	1	0	1	0	0
deep	0	1	0	0	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

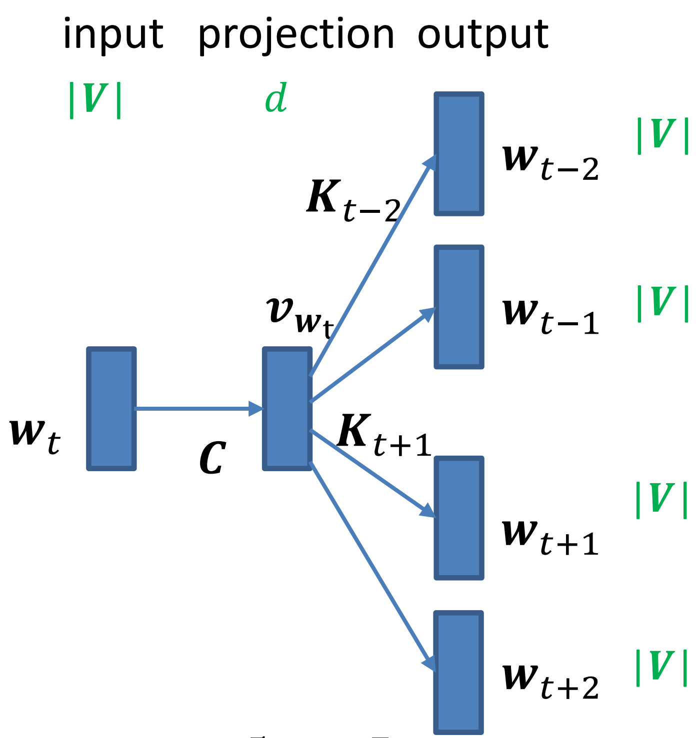
Problems: 1. very high dimensional! 2. sparse data->models are less robust

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., 2003)
  - A recent, even simpler and faster model: word2vec (Mikolov et al. 2013a) -> intro now

Main idea of word2vec

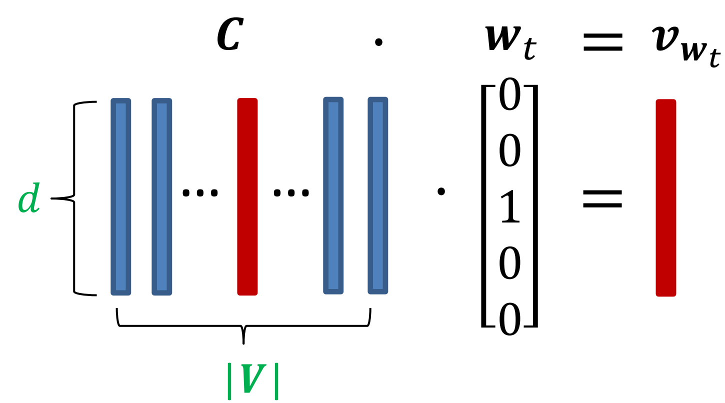
- Instead of capturing cooccurrence counts directly, predict surrounding words of every word



$|\boldsymbol{V}|$ is about $10^5 \sim 10^7$

d is about $50 \sim 1000$

- $\boldsymbol{w}_t \in R^{|\boldsymbol{V}|}, \boldsymbol{v}_{\boldsymbol{w}_t} \in R^d$
- $\boldsymbol{C} \in R^{d \times |\boldsymbol{V}|}$: Each column is the vector of a word in the vocabulary

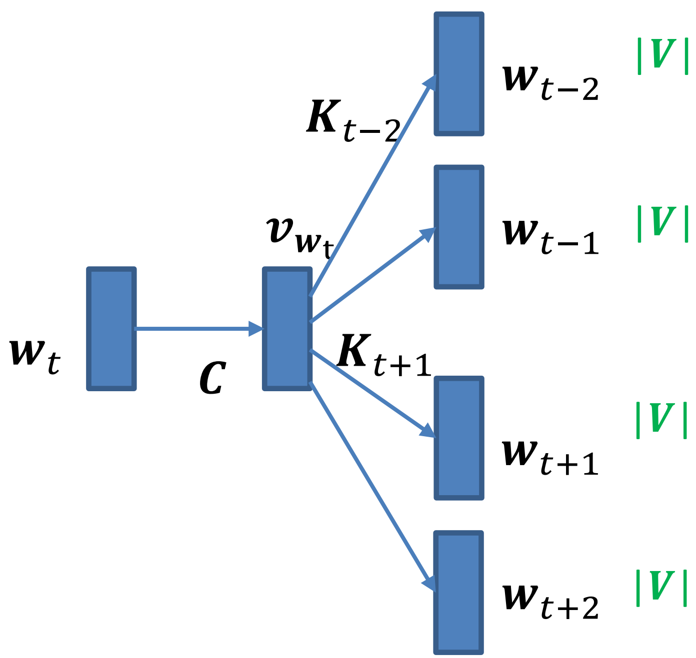


The one-hot vector $\boldsymbol{w}_t$ selects the corresponding column in $\boldsymbol{C}$

- $\boldsymbol{K}_t \in R^{|\boldsymbol{V}| \times d}$

Formulation

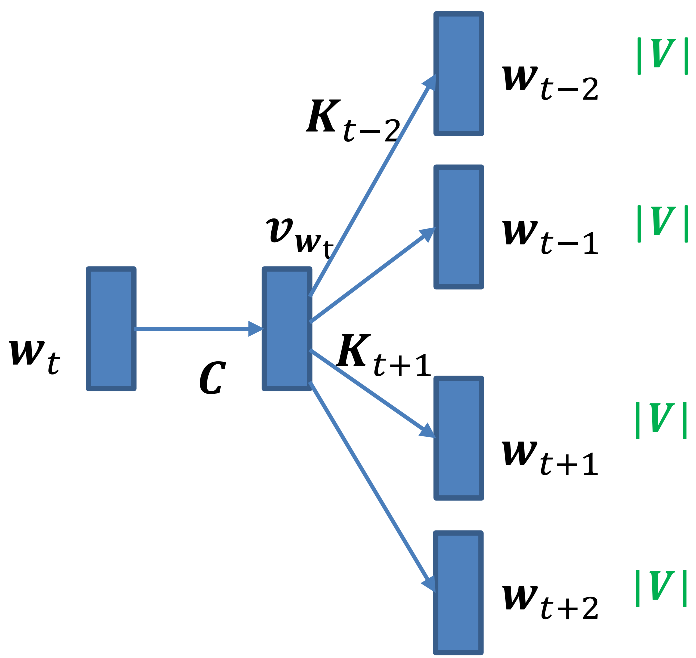
- Objective: given an input word, maximize the prob of surrounding words
- This is called skip-gram model



- $K_t \in R^{|V| \times d}$ is also a collection of vectors for all words: each row is a vector for a word
  - Every word has an "input" vector and an "output" vector
  - After learning, the two vectors can be averaged to represent a word
- Before softmax, the input to each output unit at location $t+j$ is ${v'_{w_k}}^\top v_{w_t}$, where $k = 1, \ldots, |V|$
  - v_{w_t} and v'_{w_k} denote the "input" vector and "output" vector

Formulation

- Objective: given an input word, maximize the prob of surrounding words
- This is called skip-gram model



- The prob of a word $\boldsymbol{w}_k$ at location $t+j$ is

$$p(\boldsymbol{w}_k|\boldsymbol{w}_t) = \frac{\exp\left(\boldsymbol{v}'_{\boldsymbol{w}_k}{}^{\top}\boldsymbol{v}_{\boldsymbol{w}_t}\right)}{\sum_{m=1}^{|\boldsymbol{V}|}\exp\left(\boldsymbol{v}'_{\boldsymbol{w}_m}{}^{\top}\boldsymbol{v}_{\boldsymbol{w}_t}\right)}$$

- During training, a word $\boldsymbol{w}_{t+j}$, say “learning”, is given as the output word
  - In the desired output $\boldsymbol{r}$, only the element corresponding to $\boldsymbol{w}_{t+j}$ is 1 and others are 0
  - The cross-entropy loss:

$$-\sum_{k=1}^{|\boldsymbol{V}|} r_{\boldsymbol{w}_k}\ln p(\boldsymbol{w}_k|\boldsymbol{w}_t) = -\ln p\left(\boldsymbol{w}_{t+j}\middle|\boldsymbol{w}_t\right)$$

Objective function

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

$$J^t(\boldsymbol{\theta}) = -\sum_{-c\le j\le c, j\ne 0} \ln p\left(\boldsymbol{w}_{t+j} \middle| \boldsymbol{w}_t\right)$$

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

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

$$J(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T} J^t(\boldsymbol{\theta}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0} \ln p\left(\boldsymbol{w}_{t+j} \middle| \boldsymbol{w}_t\right)$$

- It's equivalent to maximizing the average log probability
- BP algorithm and SGD are used to train the model: update parameters after each window

Alternatives to full softmax

- At *each location* in the window, the full softmax output is

$$p(\boldsymbol{w}_k|\boldsymbol{w}_t) = \frac{\exp\left({\boldsymbol{v}'_{\boldsymbol{w}_k}}^\top \boldsymbol{v}_{\boldsymbol{w}_t}\right)}{\sum_{m=1}^{|\boldsymbol{V}|} \exp\left({\boldsymbol{v}'_{\boldsymbol{w}_m}}^\top \boldsymbol{v}_{\boldsymbol{w}_t}\right)}$$

  where $k = 1, \ldots, |\boldsymbol{V}|$
- Any problem with this formulation?
  – The normalization term is computationally expensive
- Two alternatives
  – Hierarchical softmax (Morin, Bengio, 2005; Mikolov et al., 2013b)
  – Negative sampling (Mikolov et al., 2013b) -> Intro here

Negative sampling

- We use $K+1$ logistic regression models to approximate the full softmax model
  - These models are used to 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 the noise distribution $P(\boldsymbol{w})$
  - K can be 5~20 for small training datasets, while 2~5 for large datasets
- Overall objective function $J(\boldsymbol{\theta}) = \frac{1}{T} J^t(\boldsymbol{\theta})$, where

$$J^t(\boldsymbol{\theta}) = -\ln\sigma\left(\boldsymbol{v}_O'^{\top}\boldsymbol{v}_t\right) - \sum_{i=1}^{K} \mathbb{E}_{\boldsymbol{w}_i \sim P(\boldsymbol{w})} \ln\sigma\left(-\boldsymbol{v}_i'^{\top}\boldsymbol{v}_t\right)$$

  - σ is the logistic sigmoid function, $\boldsymbol{v}_t$ is the vector of the input word $\boldsymbol{w}_t$, $\boldsymbol{v}_O'$ and $\boldsymbol{v}_i'$ are the vectors of the positive and negative output words $\boldsymbol{w}_O$ and $\boldsymbol{w}_i$, respectively

Negative sampling

$$J^t(\boldsymbol{\theta}) = -\ln\sigma\left({\boldsymbol{v}'_O}^\top \boldsymbol{v}_t\right) - \sum_{i=1}^{K} \mathbb{E}_{\boldsymbol{w}_i \sim P(\boldsymbol{w})} \ln\sigma\left(-{\boldsymbol{v}'_i}^\top \boldsymbol{v}_t\right)$$

- It's easy to show that every term corresponds to a cross-entropy loss of a logistic regression model
  - Please derive by yourself (hint: the labels for positive and negative samples are 1 and 0, respectively)
- Intuitively
  - Max. prob that real outside word appears
  - Min. prob that random words appear around center word
- How to define $P(\boldsymbol{w})$?
  - It is recommended (Mikolov et al., 2013b)

$$P(\boldsymbol{w}_i) = \frac{f(\boldsymbol{w}_i)^{\frac{3}{4}}}{\sum_j f(\boldsymbol{w}_j)^{\frac{3}{4}}}$$

where $f(\boldsymbol{w}_i)$ denotes the frequency of $\boldsymbol{w}_i$ in the corpus

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 $\boldsymbol{w}_i$ in the training set is discarded with probability (Mikolov et al., 2013b)

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

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

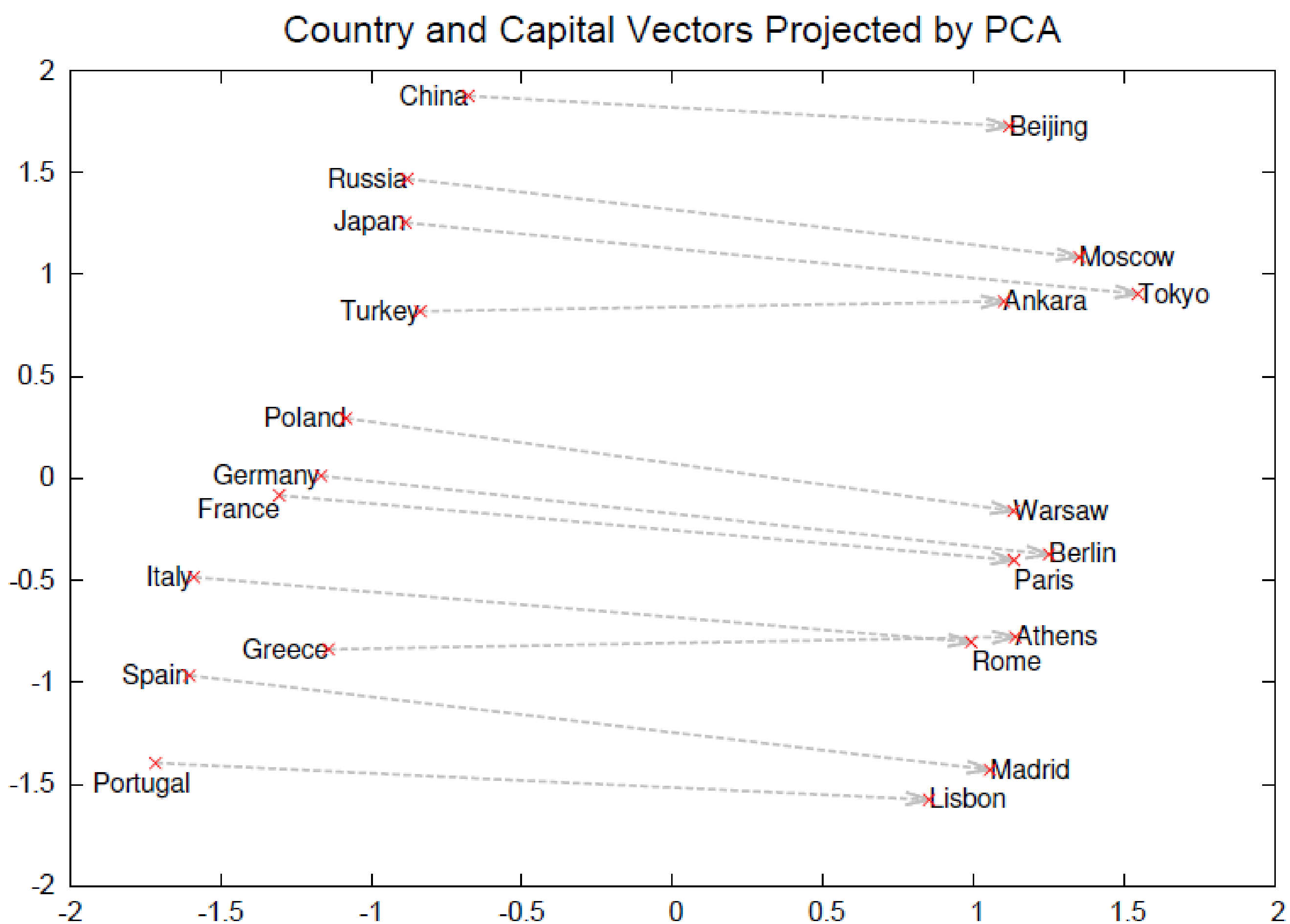
- But in the released codes, a different prob is used

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

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

Results

(Mikolov et al., 2013b)



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

Results

(Mikolov et al., 2013b)

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 is a method combining the two kinds of methods (2014)
  - Pennington et al. (2014) Glove: Global Vectors for Word Representation, EMNLP

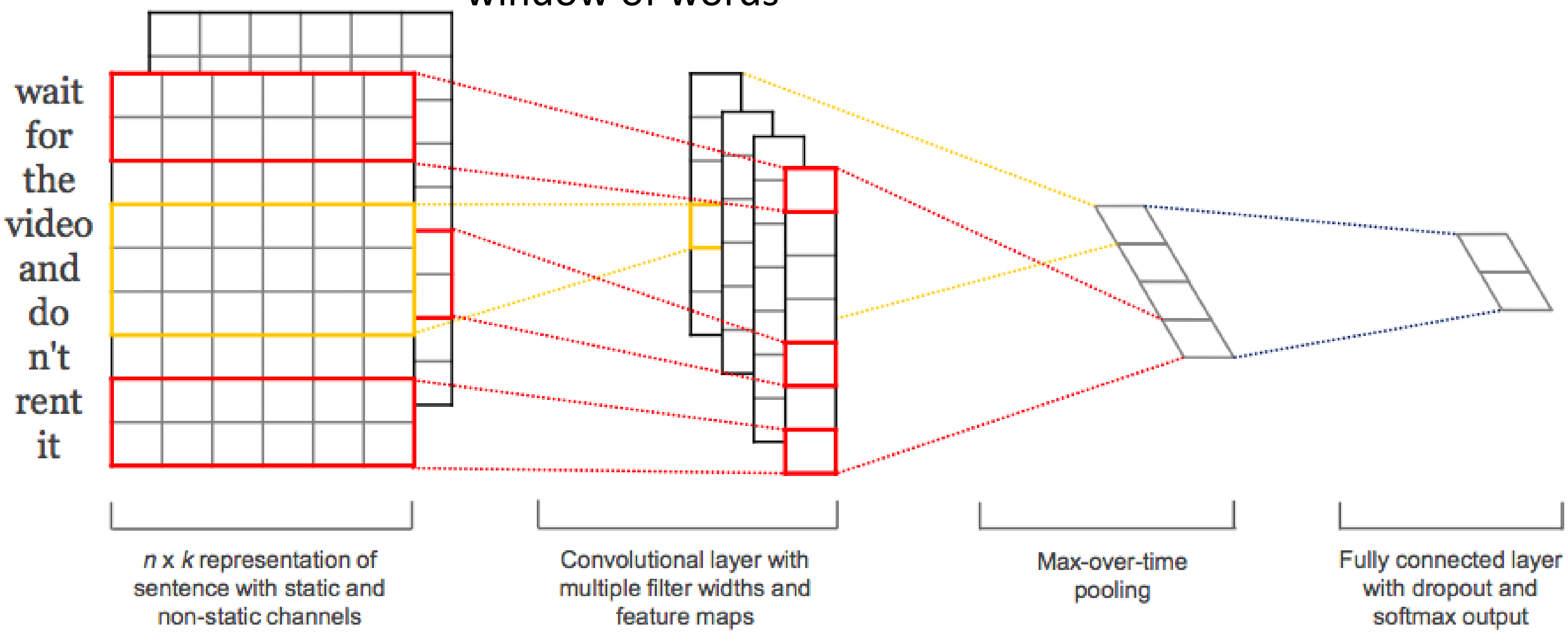
Outline

- Speech recognition
- Natural language processing
  - Typical tasks
  - Word representation
  - Text classification using NNs
  - Machine translation using RNNs

Text classification using CNNs

Kim, 2014

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



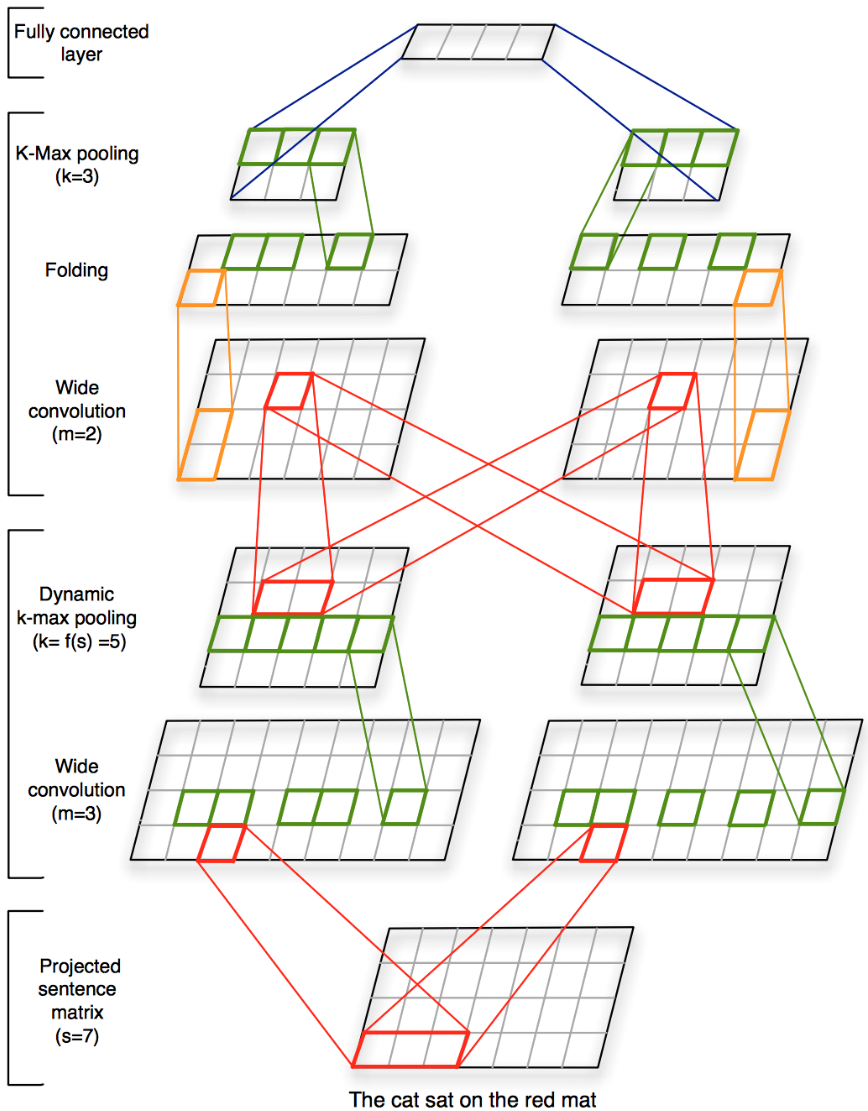
Each word has a k-D real vector, and n words are concatenated together

Each feature vector is the result corresponding to a different h

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

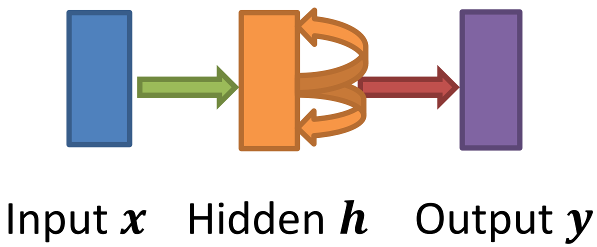


Kalchbrenner et al., 2014

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

Text classification using RNNs

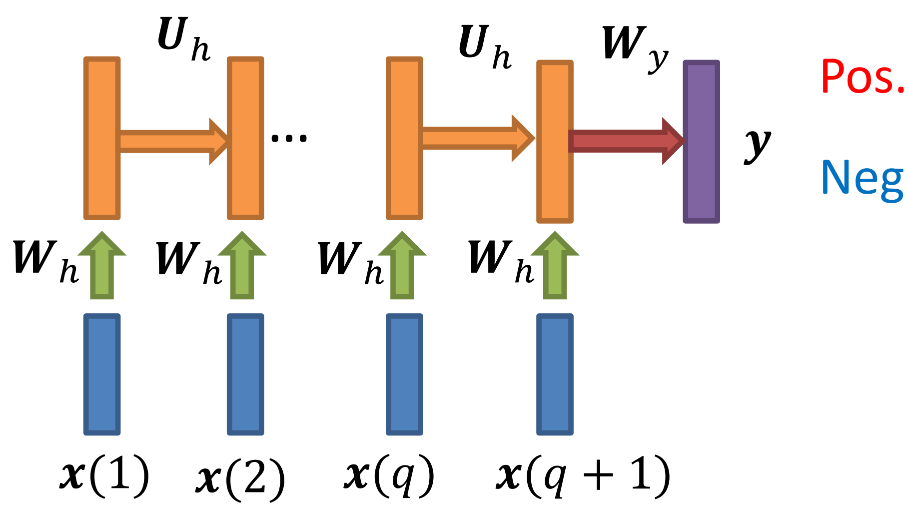
Unfold for the Elman network



$$\boldsymbol{h}(t) = \sigma_h(\boldsymbol{W}_h\boldsymbol{x}(t) + \boldsymbol{U}_h\boldsymbol{h}(t-1) + \boldsymbol{b}_h)$$

$$\boldsymbol{y}(t) = \sigma_y(\boldsymbol{W}_y\boldsymbol{h}(t) + \boldsymbol{b}_y)$$

- $\boldsymbol{x}$ is time-varying
- Label $\boldsymbol{r}$ is only present at the last step



Deep learning is so cool.

How noisy the teacher’s English is!

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

Outline

- Speech recognition
- Natural language processing
  - Typical tasks
  - Word representation
  - Text classification using CNNs
  - Machine translation using RNNs

From Abigail See‘s slides (Stanford University)

Sequence-to-sequence learning

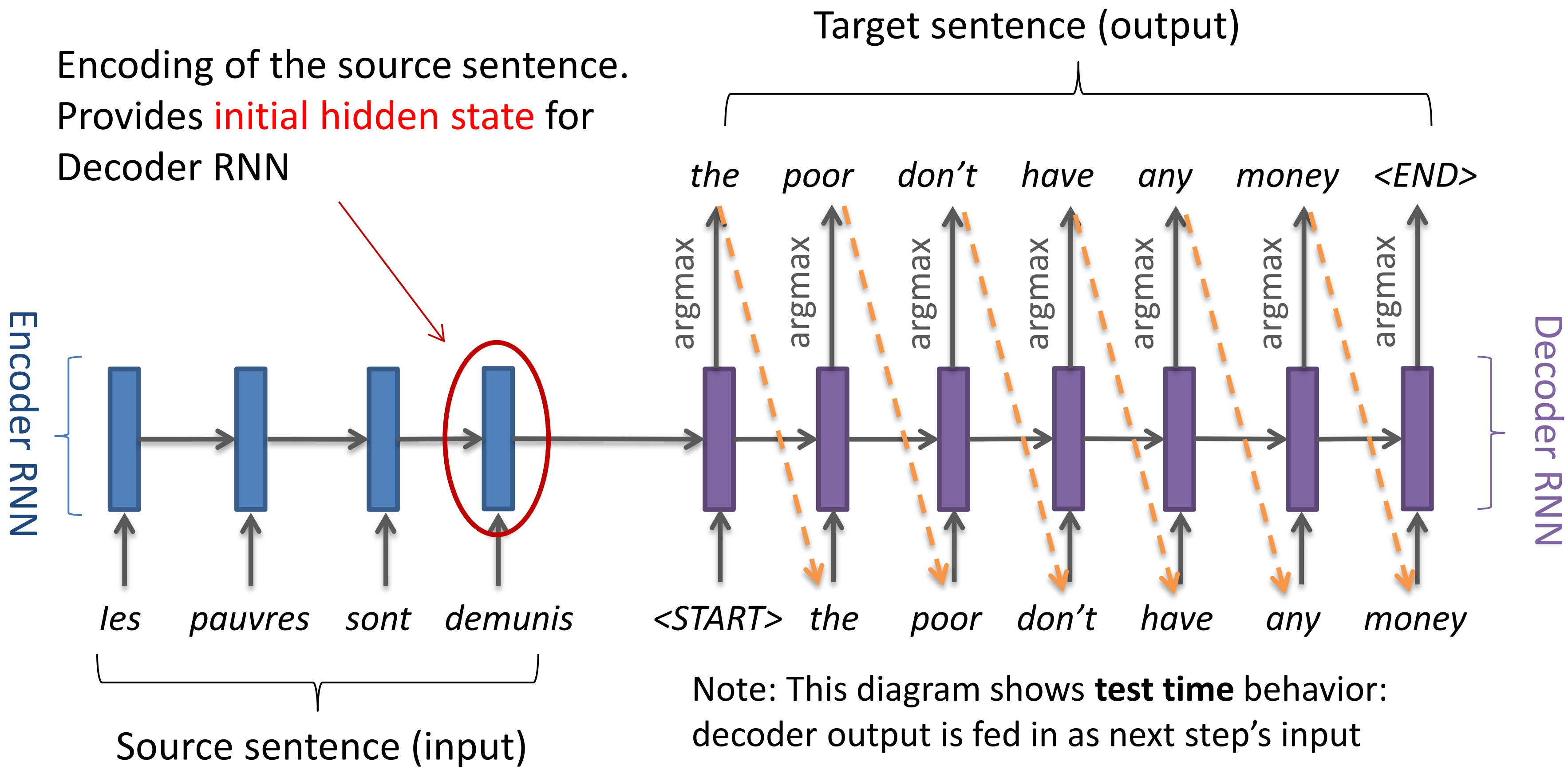
Sequence 1 → Sequence 2

- Usually it involves two RNNs: encoder and decoder
- Many NLP 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



Note: This diagram shows **test time** behavior: decoder output is fed in as next step’s input

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

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 dictionaries for the encoder and decoder
  - For some tasks, e.g., machine translation, they are different
  - For other tasks, e.g., summarization and dialog, they are the same
- The encoder RNN and decoder RNN are often different, and deep RNNs can be used

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 $P(\boldsymbol{Y}|\boldsymbol{X})$, where $\boldsymbol{Y} = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_T]$, $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_{T'}]$

$$P(\boldsymbol{Y}|\boldsymbol{X}) = P(\boldsymbol{y}_1|\boldsymbol{X})P(\boldsymbol{y}_2|\boldsymbol{y}_1, \boldsymbol{X})P(\boldsymbol{y}_3|\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{X}) \cdots \underbrace{P(\boldsymbol{y}_T|\boldsymbol{y}_1, \dots, \boldsymbol{y}_{T-1}, \boldsymbol{X})}$$

Probability of next target word, given target words so far and source sentence $\boldsymbol{X}$

- How to train an NMT?
  - You need a big parallel corpus…

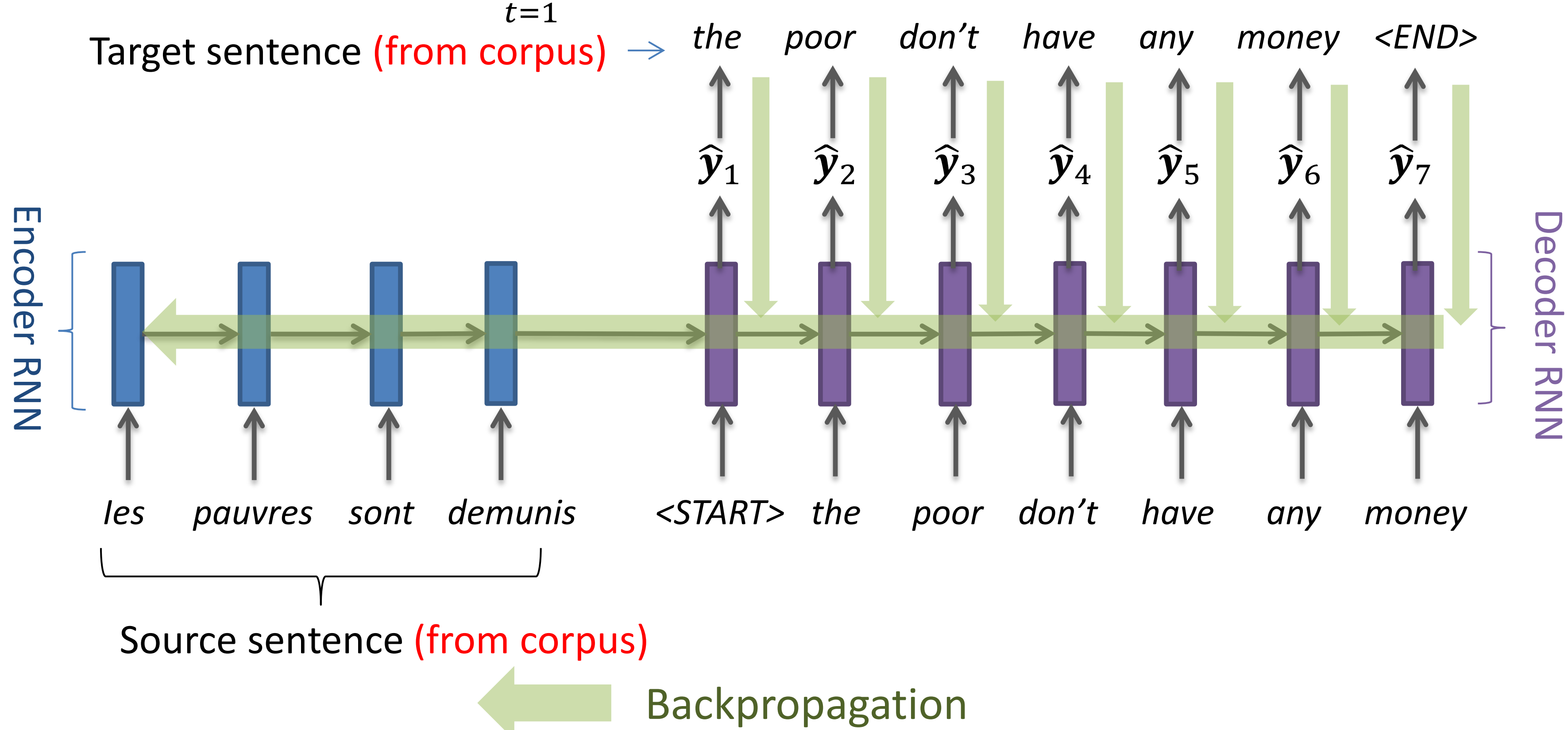
Training an NMT system

= negative log prob of “the”

= negative log prob of “have”

= negative log prob of <END>

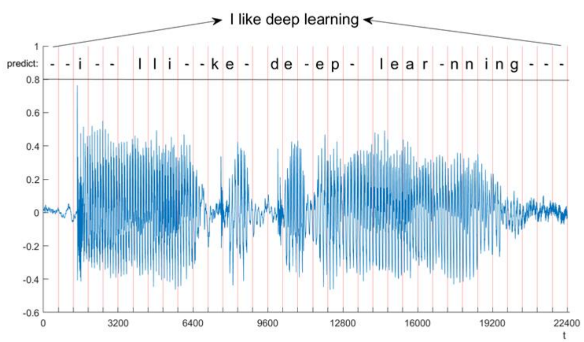
$$J = \frac{1}{T}\sum_{t=1}^{T} J_t = J_1 + J_2 + J_3 + J_4 + J_5 + J_6 + J_7$$

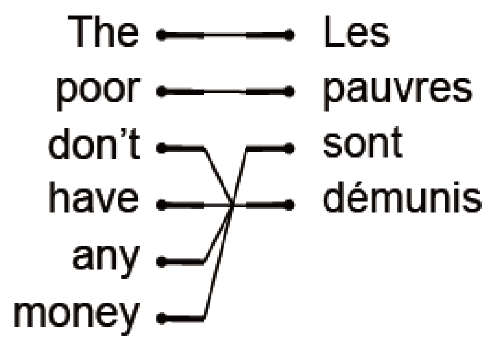


Seq2seq is optimized as a **single system**

CTC versus seq2seq

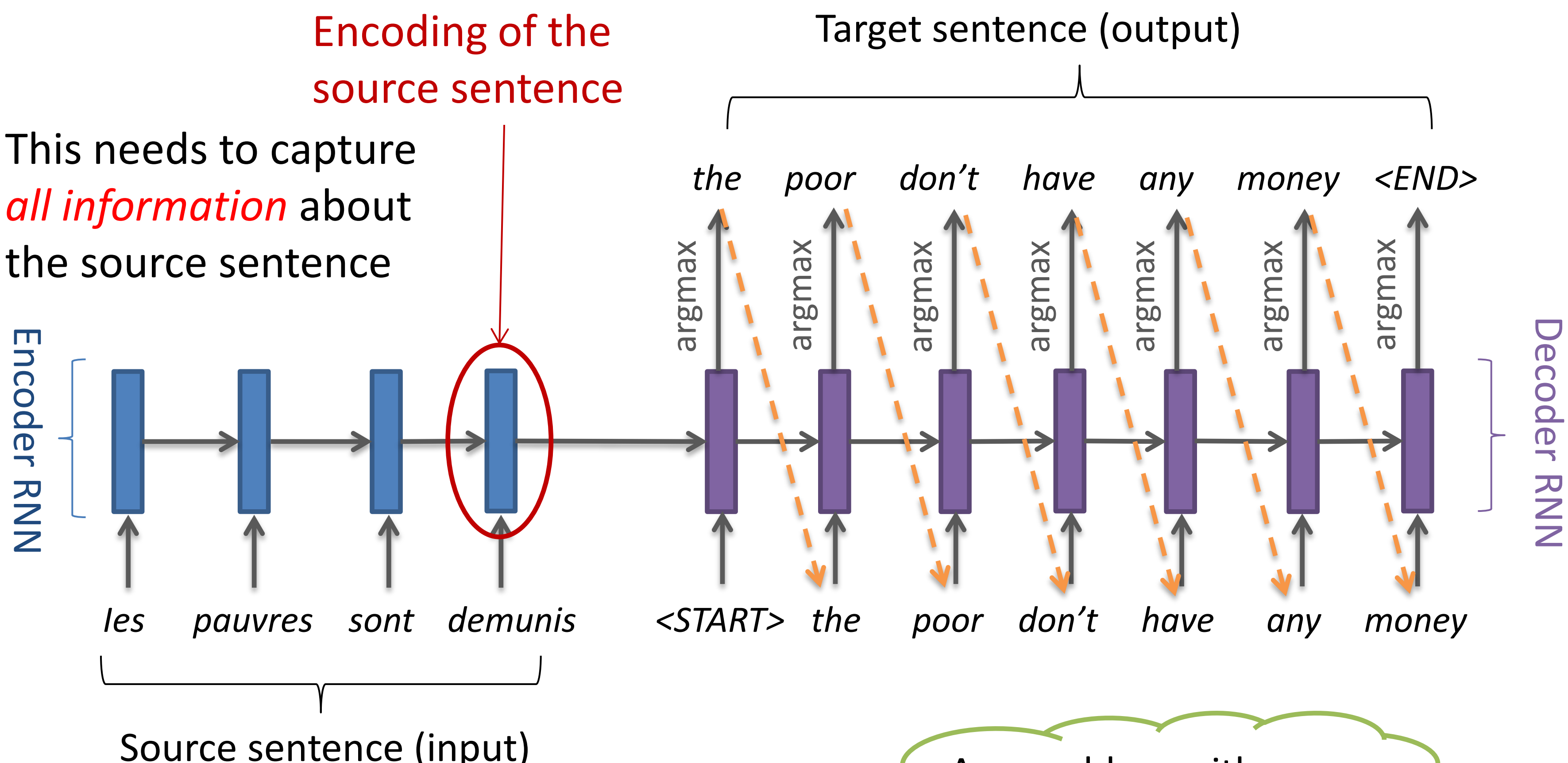
- What are in common between the two models?
  - Both are used to map one sequence to another sequence
- What are the major differences between them?
  - CTC: monotonic alignment between input and output
  - Seq2seq: many-to-many alignment





- Can we use seq2seq to do speech recognition?
  - See (Chorowski et al., NIPS 2015; Bahdanau et al., ICASSP 2016)

The bottleneck problem



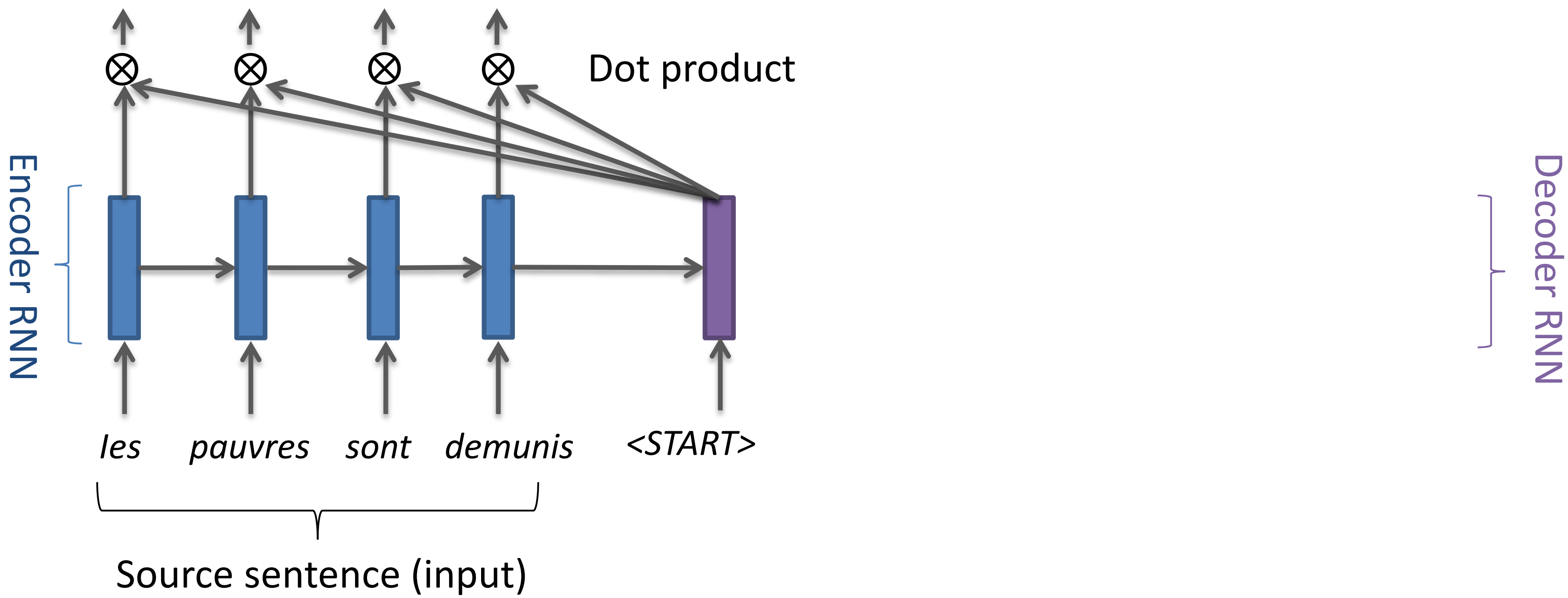
Any problem with this architecture?

Attention

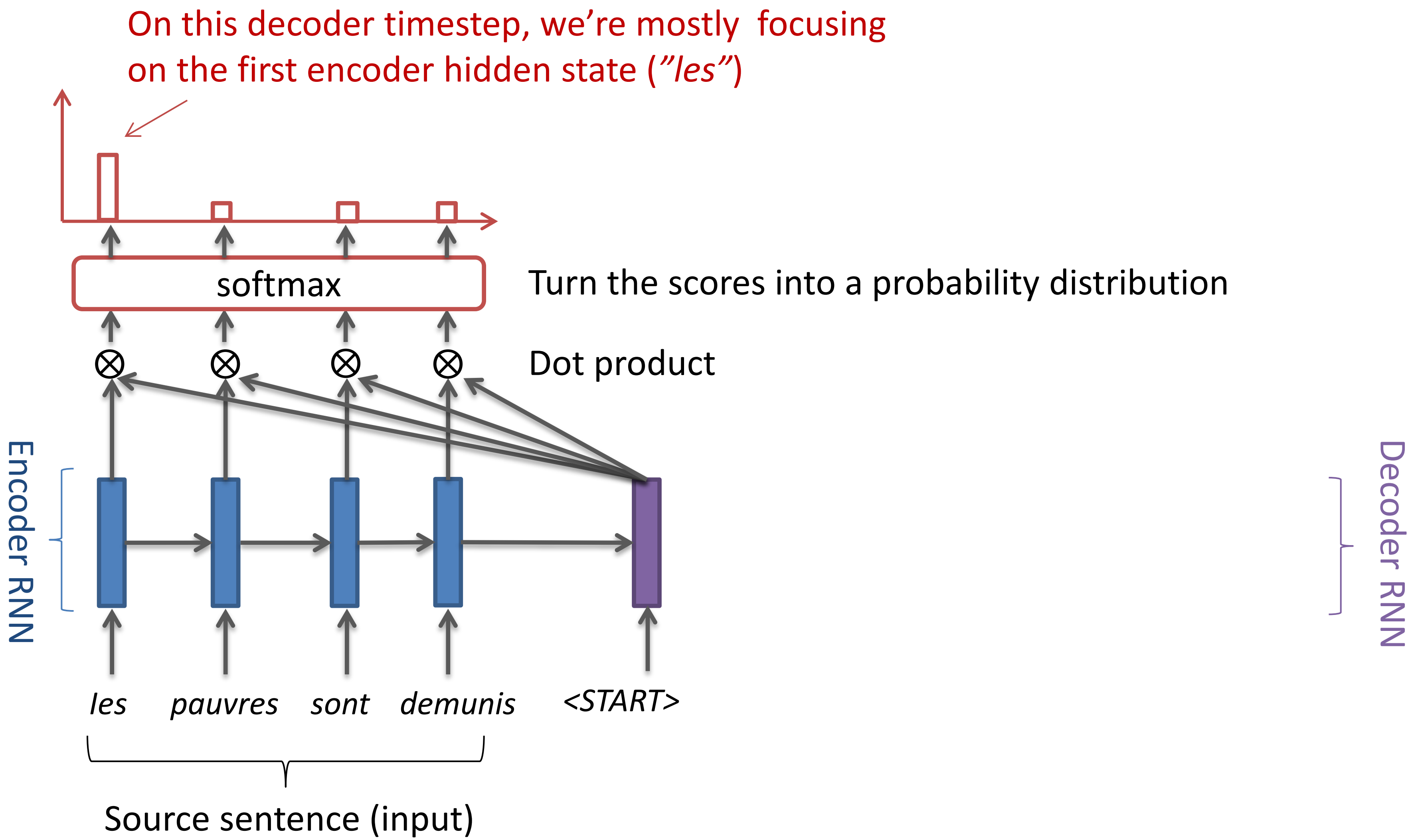
- Attention provides a solution to the bottleneck problem.
- Core idea: on each step of the decoder, *focus on a particular part* of the source sequence

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

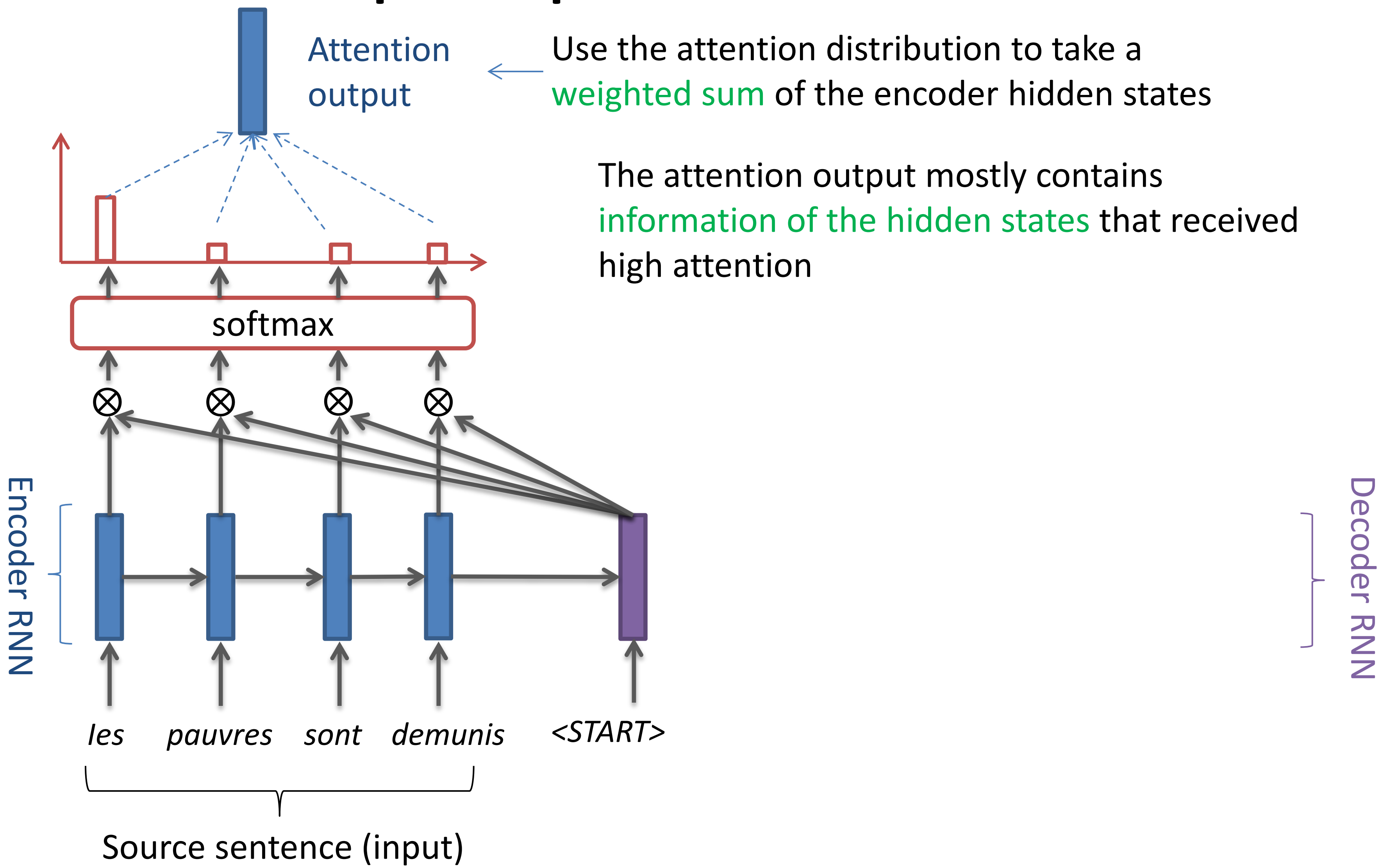
Seq2seq with attention



Seq2seq with attention



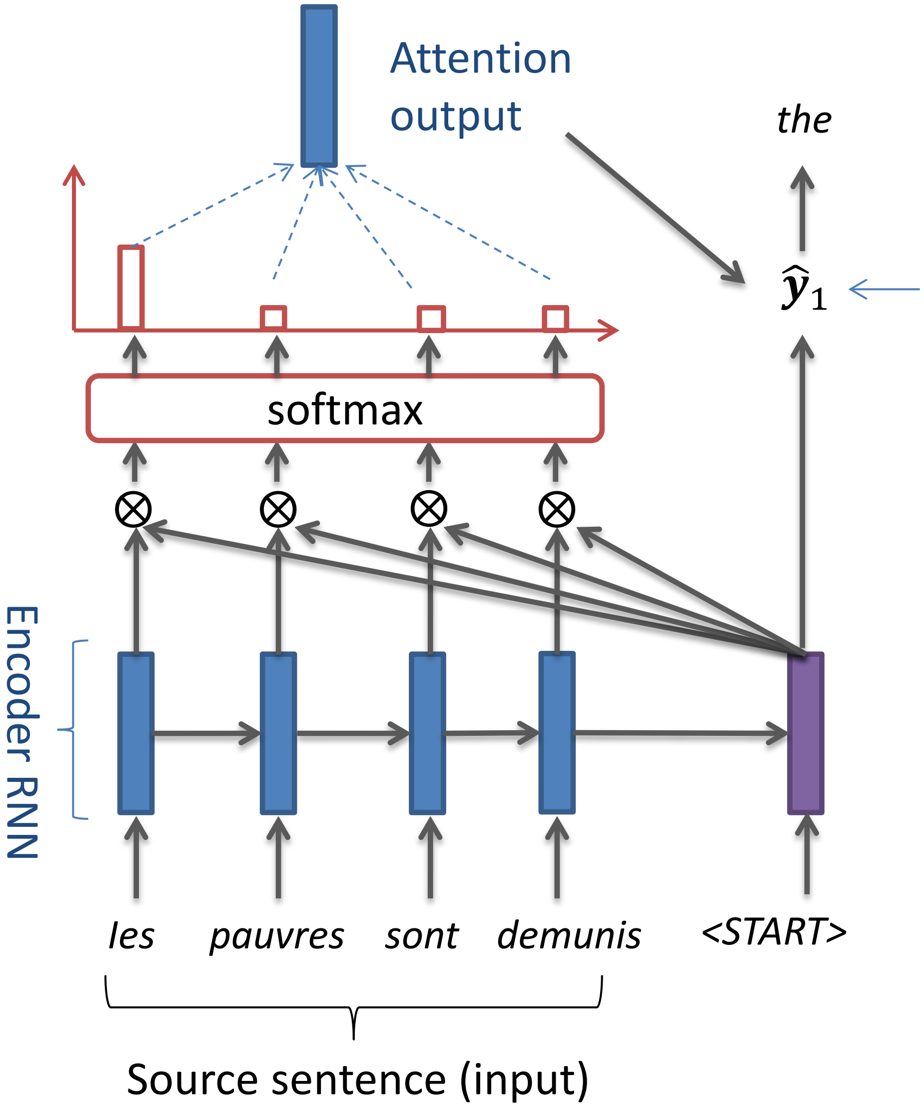
Seq2seq with attention



Use the attention distribution to take a weighted sum of the encoder hidden states

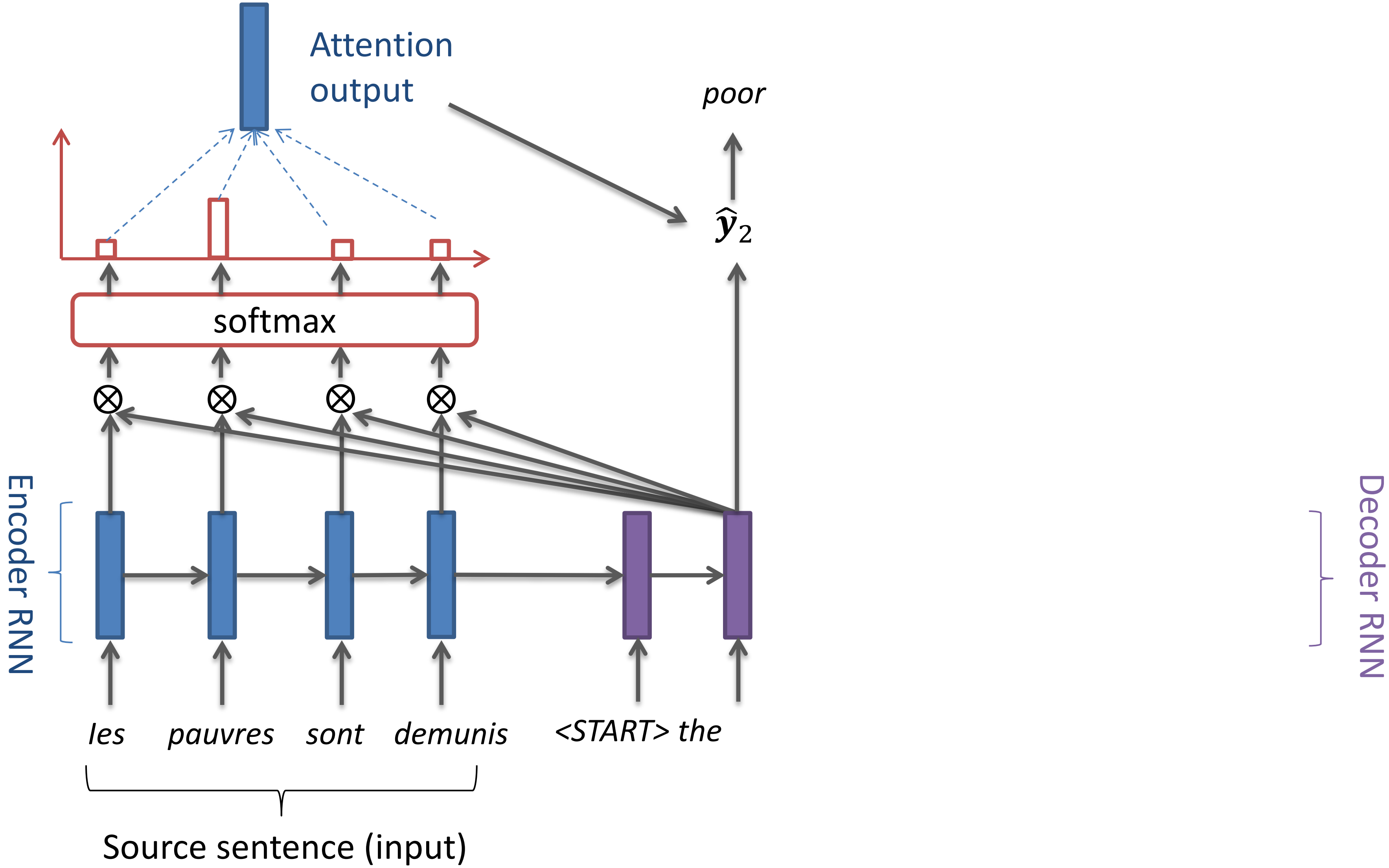
The attention output mostly contains information of the hidden states that received high attention

Seq2seq with attention

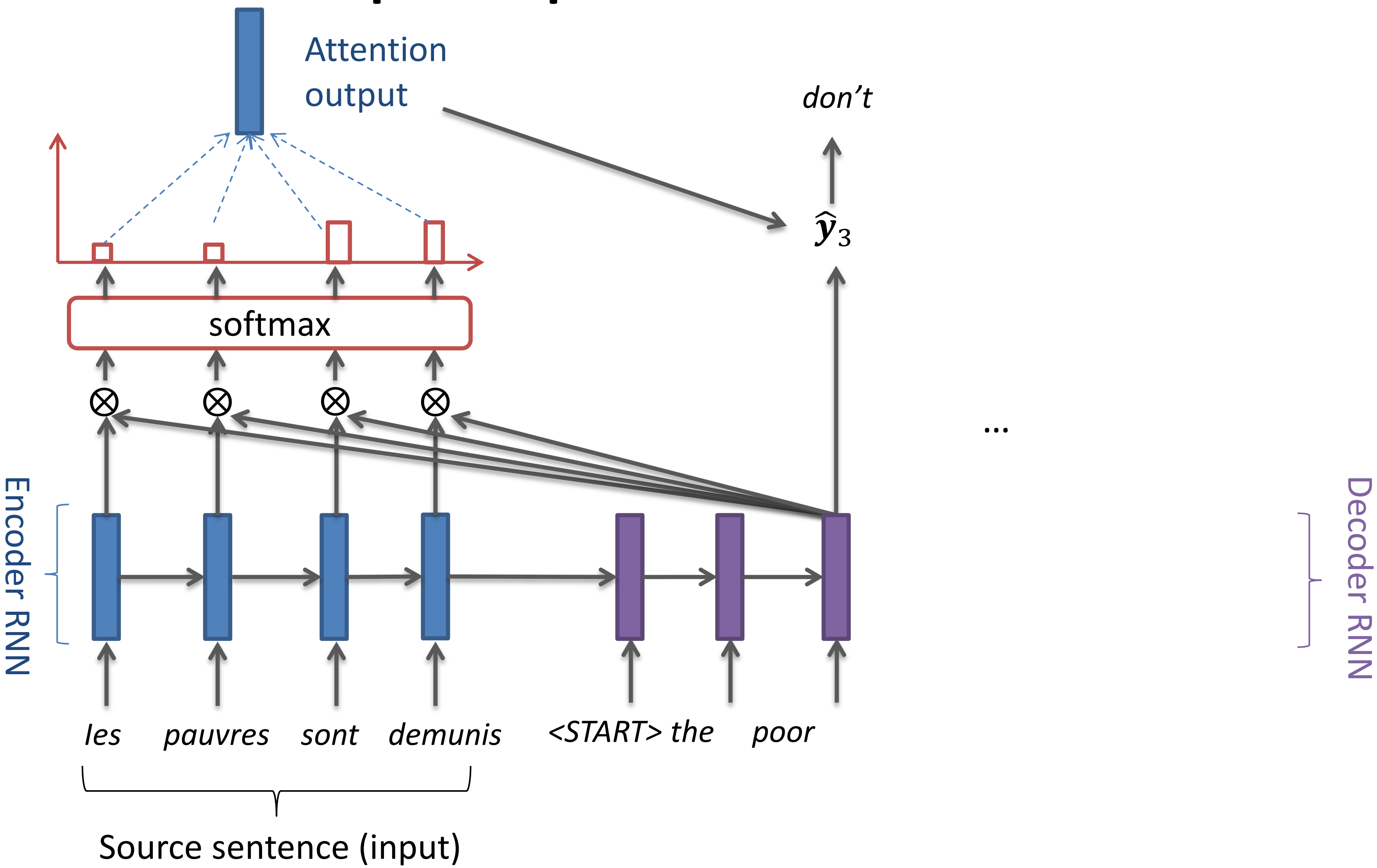


Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Seq2seq with attention



Seq2seq with attention



Formulation

The encoder's hidden states: $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in R^h$
At time step t, the decoder's hidden state $\boldsymbol{s}_t \in R^h$

- We get the attention scores $\boldsymbol{e}^t$ for this step:
$$\boldsymbol{e}^t = [\boldsymbol{s}_t^\top \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^\top \boldsymbol{h}_N] \in R^N$$
- Take softmax to get the attention distribution for this step
$$\boldsymbol{\alpha}^t = \mathrm{softmax}(\boldsymbol{e}^t) \in R^N$$
- Calculate the attention output
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in R^h$$
- Finally, concatenate the attention output with the decoder hidden state and proceed as in the non-attention seq2seq model
$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in R^{2h}$$

Summary

- Speech recognition
  - RNN+CTC
- Natural language processing
  - Typical tasks
  - Word representation
    - word2vec
  - Text classification using CNNs
  - Machine translation using RNNs
    - encoder-decoder, attention

Further reading

- Sutskever, Fernandez, Gomez, Schmidhuber (2006)
  Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks
  ICML
- Graves, Mohamed, Hinton (2013)
  Speech recognition with deep recurrent neural networks
  IEEE ICASSP
- 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

Further reading

- Mikolov, et al. (2013a)

  Efficient estimation of word representations in vector space

  arXiv preprint arXiv:1301.3781
- Mikolov, et al. (2013b)

  Distributed representations of words and phrases and their compositionality

  NIPS
- Sutskever, Vinyals, Le (2015)

  Sequence to Sequence Learning with Neural Networks

  NIPS