Lecture 8

Tian Han

Outline

- Data processing
- Word Embedding
- RNN basics

Data Processing Basics

Processing Categorical Features

Numeric Features and Categorical Features

Age	Gender	Nationality
35	Male	US
31	Male	China
29	Female	India
27	Male	US

Numeric Features and Categorical Features

Age	Gender	Nationality
35	Male	US
31	Male	China
29	Female	India
27	Male	US

- Age is a numeric feature because it is ordered.
- 35-year-old is older than 31-year-old.

Numeric Features and Categorical Features

Age	Gender	Nationality
35	**Male**	US
31	**Male**	China
29	**Female**	India
27	**Male**	US

- Gender is a binary feature: female or male. (In most people's opinion.)
- Represent ``female'' by 0.
- Represent ``male'' by 1.

Numeric Features and Categorical Features

Age	Gender	Nationality
35	1	US
31	1	China
29	0	India
27	1	US

- Gender is a binary feature: female or male. (In most people's opinion.)
- Represent ``female'' by 0.
- Represent ``male'' by 1.

Numeric Features and Categorical Features

Age	Gender	Nationality
35	1	US
31	1	China
29	0	India
27	1	US

- Nationality is a categorical feature.
- There are 200 countries (arguably.)
- We need to represent countries by numeric vectors.

Numeric Features and Categorical Features

Age	Gender	Nationality
35	1	US
31	1	China
29	0	India
27	1	US

Build a dictionary that maps countries to indices

- E.g., US→1, China→2, India→3, Japan→4, Germany→5, …
- Count from “1” (instead of “0”).

Numeric Features and Categorical Features

Age	Gender	Nationality
35	1	1
31	1	2
29	0	3
27	1	1

Build a dictionary that maps countries to indices

- E.g., US→1, China→2, India→3, Japan→4, Germany→5, …
- Count from “1” (instead of “0”).

Numeric Features and Categorical Features

Age	Gender	Nationality
35	1	1
31	1	2
29	0	3
27	1	1

Build a dictionary that maps countries to indices

- The indices are not numeric features; they cannot be compared.

Numeric Features and Categorical Features

Age	Gender	Nationality
35	1	1
31	1	2
29	0	3
27	1	1

One-hot encoding (count from 1)

- US → 1 → $[1, 0, 0, 0, \cdots, 0]$.

Numeric Features and Categorical Features

Age	Gender	Nationality
35	1	1
31	1	2
29	0	3
27	1	1

One-hot encoding (count from 1)

- US → 1 → $[1, 0, 0, 0, \cdots, 0]$.
- China → 2 → $[0, 1, 0, 0, \cdots, 0]$.

Numeric Features and Categorical Features

Age	Gender	Nationality
35	1	$[1, 0, 0, 0, \cdots, 0]$
31	1	$[0, 1, 0, 0, \cdots, 0]$
29	0	$[0, 0, 1, 0, \cdots, 0]$
27	1	$[1, 0, 0, 0, \cdots, 0]$

One-hot encoding (count from 1)

- US → 1 → $[1, 0, 0, 0, \cdots, 0]$.
- China → 2 → $[0, 1, 0, 0, \cdots, 0]$.
- ⋮

Numeric Features and Categorical Features

Age	Gender	Nationality
35	1	$[1, 0, 0, 0, \cdots, 0]$
31	1	$[0, 1, 0, 0, \cdots, 0]$
29	0	$[0, 0, 1, 0, \cdots, 0]$
27	1	$[1, 0, 0, 0, \cdots, 0]$

- Why the indices start from "1" (the US) rather than "0"?
- Reserve "0" (whose one-hot encode is $[0, 0, \cdots, 0]$) for unknown or missing nationalities.

Data Processing

- Represent a person's feature (age, gender, nationality) using a 202-dim numeric vector.

Data Processing

- Represent a person's feature (age, gender, nationality) using a 202-dim numeric vector.
- For example, convert (28, Female, China) to vector

$$[28,\ 0,\ 0, 1, 0, 0, \cdots, 0].$$

a 200-dim vector for nationality.

Data Processing

- Represent a person's feature (age, gender, nationality) using a 202-dim numeric vector.
- For example, convert $(28, \text{Female}, \text{China})$ to vector

$$[28,\ 0,\ \underbrace{0, 1, 0, 0, \cdots, 0}_{\text{a 200-dim vector for nationality.}}].$$

- For example, convert $(36, \text{Male}, \text{unknown})$ to vector

$$[36,\ 1,\ 0, 0, 0, 0, \cdots, 0].$$

Why using one-hot vectors?

- We represent nationalities using one hot vectors:
  - US: $[1, 0, 0, 0, \cdots, 0]$
  - China: $[0, 1, 0, 0, \cdots, 0]$
  - India: $[0, 0, 1, 0, \cdots, 0]$

- Why not representing nationalities using scalars?
  - 1 for “US”, 2 for “China”, and 3 for “India”.
  - This saves 200x space and computation.

Why using one-hot vectors?

- What if we use 1 for “US”, 2 for “China”, and 3 for “India”?
- Then “US”+ “China” $=$ 3 $=$ “India”.

Why using one-hot vectors?

- What if we use 1 for “US”, 2 for “China”, and 3 for “India”?
- Then “US”+ “China” $=3=$ “India”.

- What if we represent nationalities using one hot vectors?
  - US: $[1,0,0,0,\cdots,0]$.
  - China: $[0,1,0,0,\cdots,0]$.
  - India: $[0,0,1,0,\cdots,0]$.
- Then “US”+ “China” $=[1,1,0,0,\cdots,0]$.

Processing Text Data

Step 1: Tokenization (Text to Words)

- We are given a piece of text (string), e.g.,

  `S = "… to be or not to be…".`

- Break the string (string) into a list of words:

  `L = […, to, be, or, not, to, be, …],`

Step 2: Count Word Frequencies

- Build a dictionary (e.g., hash table) to count words' frequencies.
- Initially, the dictionary is empty.

Key (word)	Value (frequency)

Step 2: Count Word Frequencies

- Update the dictionary in this way:
  - If word w is **not** in the dictionary, add $(w, 1)$ to the dictionary.
  - If word w is in the dictionary, increase its frequency counter.

Key (word)	Value (frequency)
a	219
to	398
hamlet	5
be	131
not	499
prince	12
kill	31

Step 2: Count Word Frequencies

- Update the dictionary in this way:
  - If word w is **not** in the dictionary, add $(w, 1)$ to the dictionary.
  - If word w is in the dictionary, increase its frequency counter.

- Example:

...	to	be	or	not	to	be	...

Key (word)	Value (frequency)
a	219
to	398
hamlet	5
be	131
not	499
prince	12
kill	31

Step 2: Count Word Frequencies

- Update the dictionary in this way:
  - If word w is **not** in the dictionary, add $(w, 1)$ to the dictionary.
  - If word w is in the dictionary, increase its frequency counter.

- Example:

...	to	be	or	not	to	be	...

  - Word "to" is in the dictionary.

Key (word)	Value (frequency)
a	219
to	398
hamlet	5
be	131
not	499
prince	12
kill	31

Step 2: Count Word Frequencies

- Update the dictionary in this way:
  - If word w is **not** in the dictionary, add $(w, 1)$ to the dictionary.
  - If word w is in the dictionary, increase its frequency counter.

- Example:

...	to	be	or	not	to	be	...

  - Word "to" is in the dictionary.
  - Increase its counter.

Key (word)	Value (frequency)
a	219
to	399
hamlet	5
be	131
not	499
prince	12
kill	31

Step 2: Count Word Frequencies

- Update the dictionary in this way:
  - If word w is **not** in the dictionary, add $(w, 1)$ to the dictionary.
  - If word w is in the dictionary, increase its frequency counter.

- Example:

...	to	be	or	not	to	be	...

  - Word "be" is in the dictionary.

Key (word)	Value (frequency)
a	219
to	399
hamlet	5
be	131
not	499
prince	12
kill	31

Step 2: Count Word Frequencies

- Update the dictionary in this way:
  - If word w is **not** in the dictionary, add $(w, 1)$ to the dictionary.
  - If word w is in the dictionary, increase its frequency counter.

- Example:

...	to	be	or	not	to	be	...

  - Word "be" is in the dictionary.
  - Increase its counter.

Key (word)	Value (frequency)
a	219
to	399
hamlet	5
be	132
not	499
prince	12
kill	31

Step 2: Count Word Frequencies

- Update the dictionary in this way:
  - If word w is **not** in the dictionary, add $(w, 1)$ to the dictionary.
  - If word w is in the dictionary, increase its frequency counter.

- Example:

...	to	be	or	not	to	be	...

  - Word "or" is not in the dictionary.

Key (word)	Value (frequency)
a	219
to	399
hamlet	5
be	132
not	499
prince	12
kill	31

Step 2: Count Word Frequencies

- Update the dictionary in this way:
  - If word w is **not** in the dictionary, add $(w, 1)$ to the dictionary.
  - If word w is in the dictionary, increase its frequency counter.

- Example:

...	to	be	or	not	to	be	...

  - Word "or" is not in the dictionary.
  - Add ("or", 1) to the dictionary.

Key (word)	Value (frequency)
a	219
to	399
hamlet	5
or	1
be	132
not	499
prince	12
kill	31

Step 2: Count Word Frequencies

- Sort the table so that the frequency is in the descending order.

Key (word)	Value (frequency)
a	219
to	399
hamlet	5
or	1
be	132
not	499
prince	12
kill	31

Step 2: Count Word Frequencies

- Sort the table so that the frequency is in the descending order.

Key (word)	Value (frequency)
not	499
to	399
a	219
be	132
kill	31
prince	12
hamlet	5
or	1

Step 2: Count Word Frequencies

- Sort the table so that the frequency is in the descending order.
- Replace “frequency” by “index” (starting from 1.)

Key (word)	Value (frequency)
not	499
to	399
a	219
be	131
kill	31
prince	12
hamlet	5
or	1

Step 2: Count Word Frequencies

- Sort the table so that the frequency is in the descending order.
- Replace “frequency” by “index” (starting from 1.)
- The number of unique words is called “vocabulary”.

Key (word)	Value (index)
not	1
to	2
a	3
be	4
kill	5
prince	6
hamlet	7
or	8

Step 2: Count Word Frequencies

- If the vocabulary is too big, e.g., greater than 10K, then keep only the 10K most frequent words.
- Why removing infrequent words?

Key (word)	Value (index)
not	1
to	2
a	3
be	4
kill	5
prince	6
hamlet	7
or	8

Step 2: Count Word Frequencies

- If the vocabulary is too big, e.g., greater than 10K, then keep only the 10K most frequent words.
- Why removing infrequent words?

1. Infrequent words are usually meaningless, e.g.,
   - Name entities, e.g., “Tian”.
   - Typos, e.g., “prinse” and “hemlat”.
2. Bigger vocabulary ➔ higher-dim one-hot vectors.
   - Slower computation.
   - More parameters in word-embedding layer.

Key (word)	Value (index)
not	1
to	2
a	3
be	4
kill	5
prince	6
hamlet	7
or	8

Step 3: One-Hot Encoding

- Map every word to its index.
- For example,

```
Words: [to, be, or, not, to, be]
```

```
Indices: [2,  4,  8,  1,  2,  4]
```

Key (word)	Value (index)
not	1
to	2
a	3
be	4
kill	5
prince	6
hamlet	7
or	8

Step 3: One-Hot Encoding

- Map every word to its index.
- For example,

```
Words: [to, be, or, not, to, be]
```

```
Indices: [2,  4,  8,  1,  2,  4]
```

- If necessary, convert every index to a one-hot vector.
  - The one-hot vector' dimension is the vocabulary.
  - Vocabulary means # of unique words in the dictionary.

Key (word)	Value (index)
not	1
to	2
a	3
be	4
kill	5
prince	6
hamlet	7
or	8

Step 3: One-Hot Encoding

- If a word (e.g., typo) cannot be found in the dictionary, then simply ignore it, or encode it as 0.
- Example:

a typo:

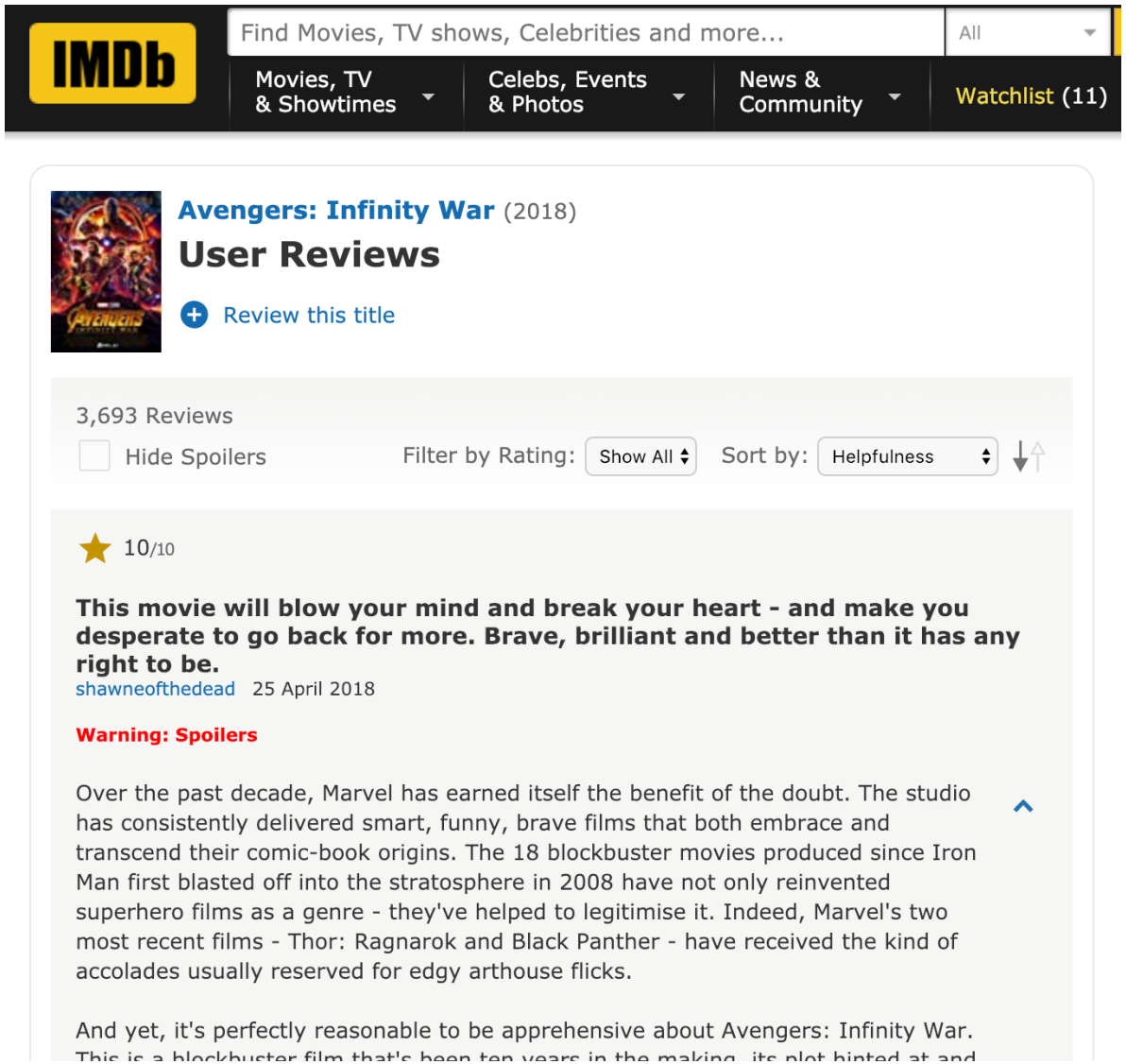
```
Words: [to, bi, or]
```

```
Indices: [2,    8]
```

Key (word)	Value (index)
not	1
to	2
a	3
be	4
kill	5
prince	6
hamlet	7
or	8

Text Processing and Word Embedding

The IMDB Movie Review Dataset

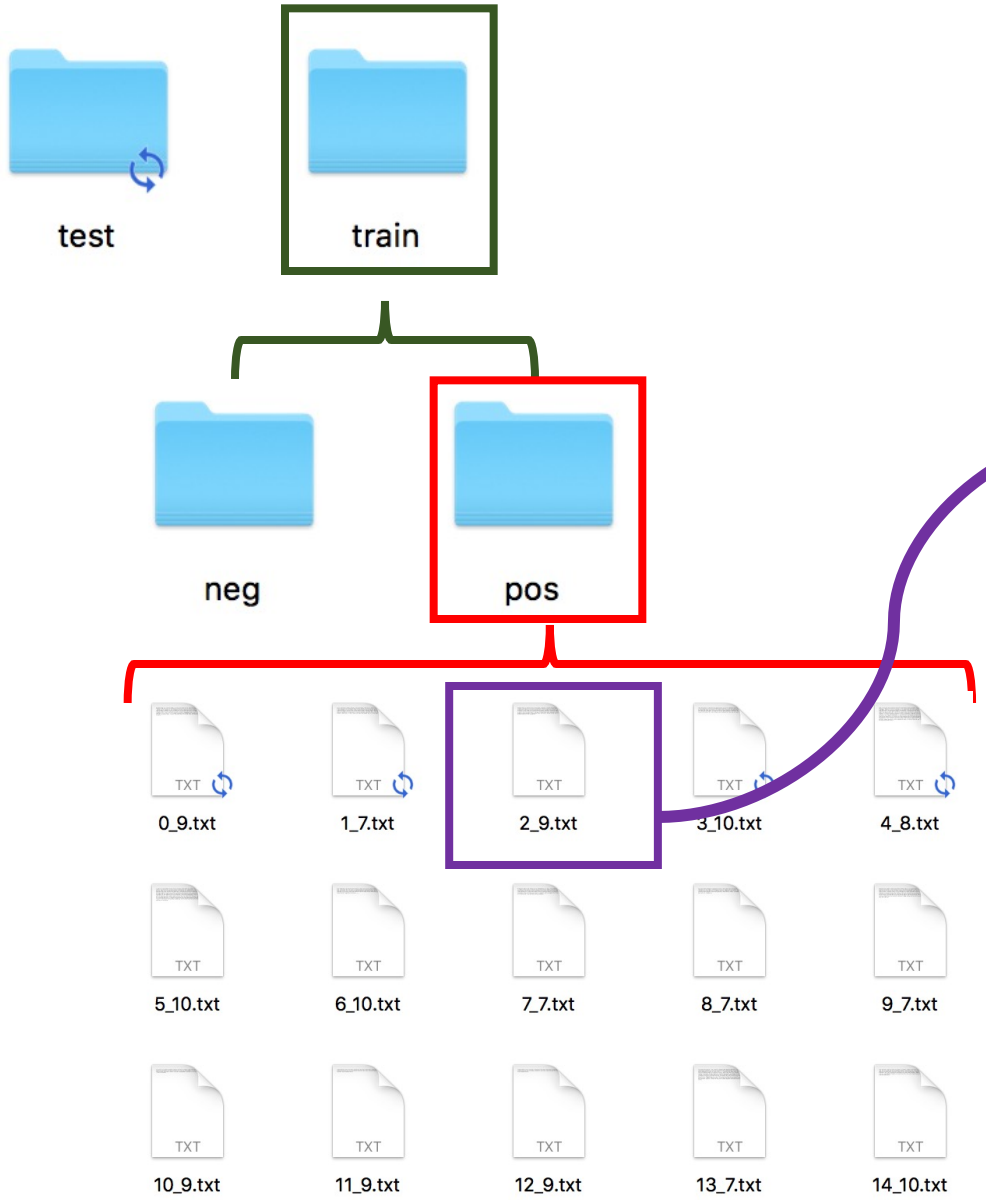


The IMDB Movie Review Dataset

- 50K movie reviews (text).
- Each review is labeled with either “positive” or “negative”.
- It is a binary classification problem.
- 25K for training and 25K for test.

- Download from
  - http://ai.stanford.edu/~amaas/data/sentiment/
  - http://s3.amazonaws.com/text-datasets/aclImdb.zip

The IMDB Movie Review Dataset



2_9.txt

Bromwell High is nothing short of brilliant. Expertly scripted and perfectly delivered, this searing parody of a students and teachers at a South London Public School leaves you literally rolling with laughter. It's vulgar, provocative, witty and sharp. The characters are a superbly caricatured cross section of British society (or to be more accurate, of any society). Following the escapades of Keisha, Latrina and Natella, our three "protagonists" for want of a better term, the show doesn't shy away from parodying every imaginable subject. Political correctness flies out the window in every episode. If you enjoy shows that aren't afraid to poke fun of every taboo subject imaginable, then Bromwell High will not disappoint!

Text to Sequence

Step 1: Tokenization

- Tokenization breaks a piece of text down into a list of tokens.
- Here, a token is a word. (A token can be a character in some applications.)

`texts[i]` = “the cat sat on the mat.”

`tokens[i]` = [“the”, “cat”, “sat”, “on”, “the”, “mat”]

Step 1: Tokenization

- Tokenization breaks a piece of text down into a list of tokens.
- Here, a token is a word. (A token can be a character in some applications.)

`texts[i]` = "the cat sat on the mat."

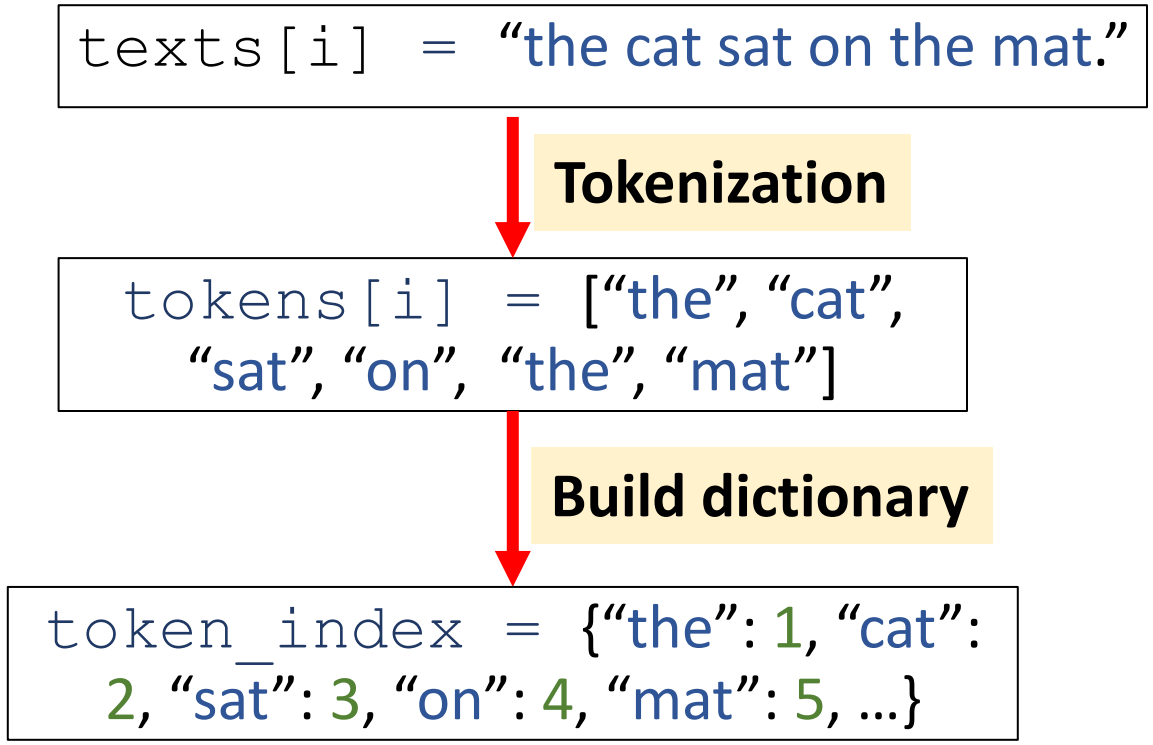
Tokenization

`tokens[i]` = ["the", "cat", "sat", "on", "the", "mat"]

- Considerations in tokenization:
  - Upper case to lower case. ("Apple" to "apple"?)
  - Remove stop words, e.g., "the", "a", "of", etc.
  - Typo correction. ("goood" to "good".)

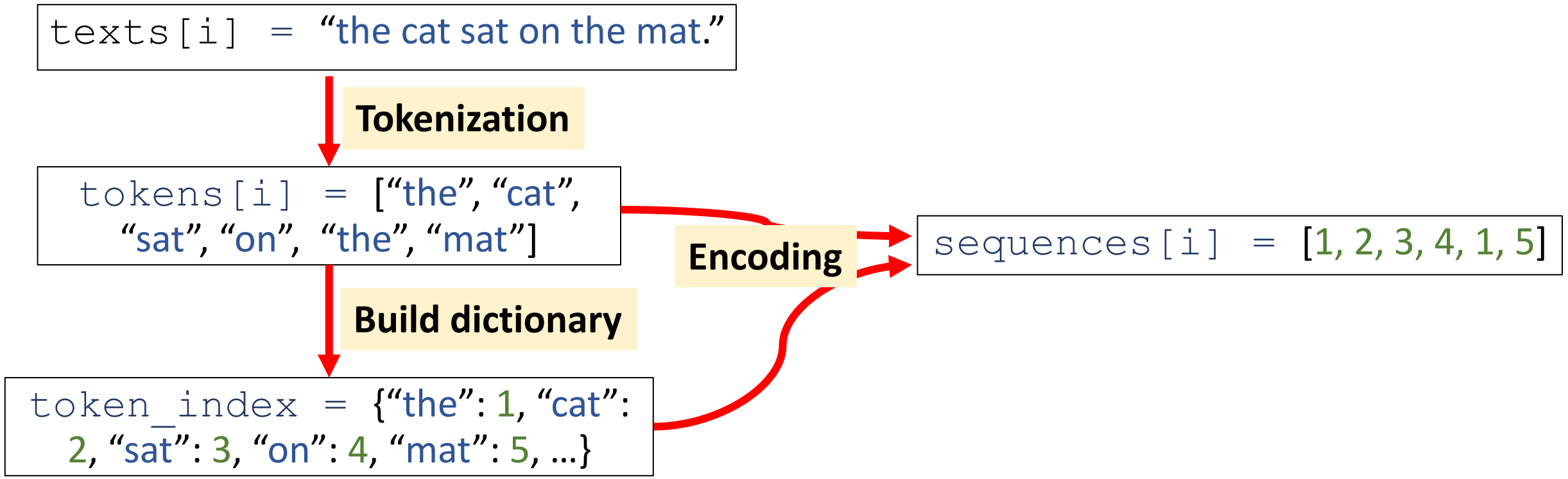
Step 2: Build Dictionary

- Use a dictionary (hash table) to count word frequencies.
- The dictionary maps word to index.



Step 3: One-Hot Encoding

- Use the dictionary to map words to indices (integers).
- A list of indices is called a sequence.



Step 3: One-Hot Encoding

texts[0]

For a movie that gets no respect there sure are a lot of memorable quotes listed for this gem. Imagine a movie where Joe Piscopo is actually funny! Maureen Stapleton is a scene stealer. The Moroni character is an absolute scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt.

sequences[0]

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 11, 12, 13,
14, 15, 1, 16, 17, 18, 2, 3, 19, 20, 21, 22,
23, 24, 25, 26, 22, 2, 27, 28, 29, 30, 31, 22,
32, 33, 34, 35, 1, 36, 29, 37, 38, 39, 40, 2,
41, 42]
```

texts[5]

I saw the movie with two grown children. Although it was not as clever as Shrek, I thought it was rather good. In a movie theatre surrounded by children who were on spring break, there was not a sound so I know the children all liked it. There parents also seemed engaged. The death and apparent death of characters brought about the appropriate gasps and comments. Hopefully people realize this movie was made for kids. As such, it was successful although I liked it too. Personally I liked the Scrat!!

sequences[5]

```
[178, 486, 29, 3, 46, 407, 487, 488, 489, 272,
160, 273, 40, 490, 40, 491, 178, 492, 272, 160,
493, 494, 193, 2, 3, 495, 496, 51, 488, 64,
385, 97, 497, 498, 8, 160, 273, 2, 499, 459,
178, 335, 29, 488, 293, 500, 272, 8, 196, 357,
501, 502, 29, 263, 110, 503, 263, 12, 504, 141,
391, 29, 505, 506, 110, 507, 508, 509, 510, 16,
3, 160, 511, 1, 199, 40, 377, 272, 160, 512,
489, 178, 500, 272, 513, 514, 178, 500, 29,
515]
```

Step 3: One-Hot Encoding

Result of encoding: $25K$ lists of integers; the i-th list has w_i elements.

texts[0]

For a movie that gets no respect there sure are a lot of memorable quotes listed for this gem. Imagine a movie where Joe Piscopo is actually funny! Maureen Stapleton is a scene stealer. The Moroni character is an absolute scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt.

sequences[0]

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 11, 12, 13,
14, 15, 1, 16, 17, 18, 2, 3, 19, 20, 21, 22,
23, 24, 25, 26, 22, 2, 27, 28, 29, 30, 31, 22,
32, 33, 34, 35, 1, 36, 29, 37, 38, 39, 40, 2,
41, 42]
```

$w_0 = 52$ tokens

texts[5]

I saw the movie with two grown children. Although it was not as clever as Shrek, I thought it was rather good. In a movie theatre surrounded by children who were on spring break, there was not a sound so I know the children all liked it. There parents also seemed engaged. The death and apparent death of characters brought about the appropriate gasps and comments. Hopefully people realize this movie was made for kids. As such, it was successful although I liked it too. Personally I liked the Scrat!!

sequences[5]

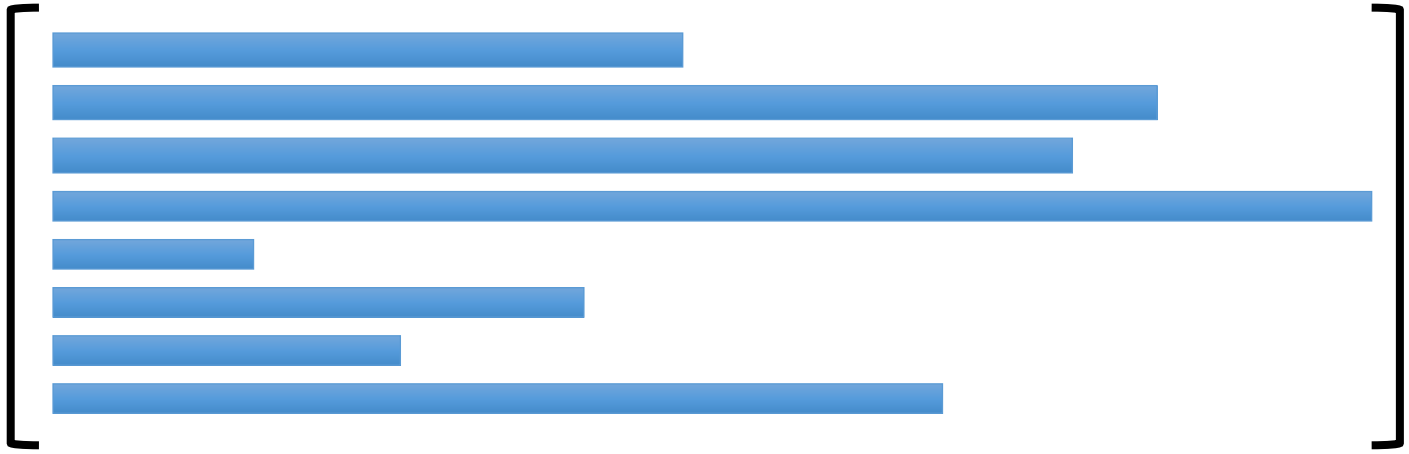
```
[178, 486, 29, 3, 46, 407, 487, 488, 489, 272,
160, 273, 40, 490, 40, 491, 178, 492, 272, 160,
493, 494, 193, 2, 3, 495, 496, 51, 488, 64,
385, 97, 497, 498, 8, 160, 273, 2, 499, 459,
178, 335, 29, 488, 293, 500, 272, 8, 196, 357,
501, 502, 29, 263, 110, 503, 263, 12, 504, 141,
391, 29, 505, 506, 110, 507, 508, 509, 510, 16,
3, 160, 511, 1, 199, 40, 377, 272, 160, 512,
489, 178, 500, 272, 513, 514, 178, 500, 29,
515]
```

$w_5 = 90$ tokens

Step 4: Align Sequences

Result of encoding: $25K$ lists of integers; the i-th list has w_i elements.

Problem: the training samples are not aligned (they have different lengths, w_i).



Step 4: Align Sequences

Result of encoding: $25K$ lists of integers; the i-th list has w_i elements.

Problem: the training samples are not aligned (they have different lengths, w_i).

Solution:

- Cut off the text to keep w words, e.g., $w = 7$.

"the fat cat sat still on the big red mat."

"sat still on the big red mat."

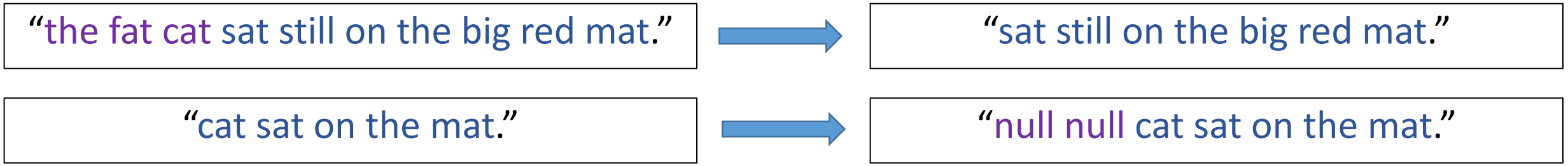
Step 4: Align Sequences

Result of encoding: $25K$ lists of integers; the i-th list has w_i elements.

Problem: the training samples are not aligned (they have different lengths, w_i).

Solution:

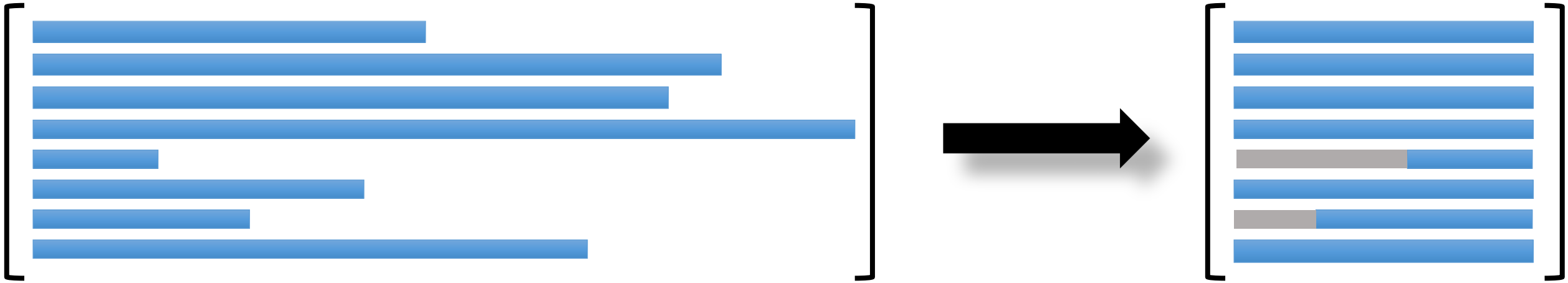
- Cut off the text to keep w words, e.g., $w = 7$.
- If the text is shorter than w, pad it with zeros.



Step 4: Align Sequences

Result of encoding: $25K$ lists of integers; the i-th list has w elements.

Aligned!



Text Processing in Keras

Steps 1, 2, & 3: Text to Sequence

```
from keras.preprocessing.text import Tokenizer

vocabulary = 10000
tokenizer = Tokenizer(num_words=vocabulary)
tokenizer.fit_on_texts(texts_train)

word_index = tokenizer.word_index
sequences_train = tokenizer.texts_to_sequences(texts_train)
```

Steps 1, 2, & 3: Text to Sequence

```
from keras.preprocessing.text import Tokenizer

vocabulary = 10000
tokenizer = Tokenizer(num_words=vocabulary)
tokenizer.fit_on_texts(texts_train)
```

1. Tokenize the movie reviews
2. Build a dictionary.

Steps 1, 2, & 3: Text to Sequence

```
from keras.preprocessing.text import Tokenizer

vocabulary = 10000
tokenizer = Tokenizer(num_words=vocabulary)
tokenizer.fit_on_texts(texts_train)

word_index = tokenizer.word_index
```

- word_index: dict[(string, int)]

Steps 1, 2, & 3: Text to Sequence

```
from keras.preprocessing.text import Tokenizer

vocabulary = 10000
tokenizer = Tokenizer(num_words=vocabulary)
tokenizer.fit_on_texts(texts_train)

word_index = tokenizer.word_index
sequences_train = tokenizer.texts_to_sequences(texts_train)
```

- texts_train: list[string]
- sequence_train: list[list[int]]

Steps 1, 2, & 3: Text to Sequence

```
from keras.preprocessing.text import Tokenizer

vocabulary = 10000
tokenizer = Tokenizer(num_words=vocabulary)
tokenizer.fit_on_texts(texts_train)

word_index = tokenizer.word_index
sequences_train = tokenizer.texts_to_sequences(texts_train)
```

```
print(sequences_train[0])
```

```
[15, 3, 17, 12, 211, 54, 1158, 47, 249, 23, 3, 173, 4, 903, 4381, 3559, 15, 11, 1525, 835, 3,
17, 118, 911, 6, 162, 160, 7262, 6, 3, 133, 1, 106, 6, 32, 1552, 2032, 103, 15, 1605, 1, 859
5, 1789, 14, 3, 565, 6259]
```

Steps 4: Align Sequences

```
from keras import preprocessing

word_num = 20
x_train = preprocessing.sequence.pad_sequences(sequences_train, maxlen=word_num)
```

```
x_train.shape
```

```
(25000, 20)
```

```
x_train[0]
```

```
array([7262,    6,    3,  133,    1,  106,    6,   32, 1552, 2032,  103,
         15, 1605,    1, 8595, 1789,   14,    3,  565, 6259], dtype=int32)
```

Texts to Sequences: Summary

`texts[0]:`

"For a movie that gets no respect there sure are a lot of memorable quotes listed for this gem. Imagine a movie where Joe Piscopo is actually funny! Maureen Stapleton is a scene stealer. The Moroni character is an absolute scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt."

Tokenization

`tokens[0]:`

```
['for', 'a', 'movie', 'that', 'gets', 'no', 'respect', 'there', 'sure', 'are', 'a',
'lot', 'of', 'memorable', 'quotes', 'listed', 'for', 'this', 'gem', 'imagine', 'a',
'movie', 'where', 'joe', 'piscopo', 'is', 'actually', 'funny', 'maureen',
'stapleton', 'is', 'a', 'scene', 'stealer', 'the', 'moroni', 'character', 'is',
'an', 'absolute', 'scream', 'watch', 'for', 'alan', 'the', 'skipper', 'hale', 'jr',
'as', 'a', 'police', 'sgt']
```

Texts to Sequences: Summary

`texts[0]:`

"For a movie that gets no respect there sure are a lot of memorable quotes listed for this gem. Imagine a movie where Joe Piscopo is actually funny! Maureen Stapleton is a scene stealer. The Moroni character is an absolute scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt."

Tokenization

`tokens[0]:`

```
['for', 'a', 'movie', 'that', 'gets', 'no', 'respect', 'there', 'sure', 'are', 'a',
'lot', 'of', 'memorable', 'quotes', 'listed', 'for', 'this', 'gem', 'imagine', 'a',
'movie', 'where', 'joe', 'piscopo', 'is', 'actually', 'funny', 'maureen',
'stapleton', 'is', 'a', 'scene', 'stealer', 'the', 'moroni', 'character', 'is',
'an', 'absolute', 'scream', 'watch', 'for', 'alan', 'the', 'skipper', 'hale', 'jr',
'as', 'a', 'police', 'sgt']
```

Encoding

`seqs[0]:`

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 11, 12, 13, 14, 15, 1, 16, 17, 18, 2, 3, 19, 20,
21, 22, 23, 24, 25, 26, 22, 2, 27, 28, 29, 30, 31, 22, 32, 33, 34, 35, 1, 36, 29,
37, 38, 39, 40, 2, 41, 42]
```

Texts to Sequences: Summary

`texts[0]:`

"For a movie that gets no respect there sure are a lot of memorable quotes listed for this gem. Imagine a movie where Joe Piscopo is actually funny! Maureen Stapleton is a scene stealer. The Moroni character is an absolute scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt."

Tokenization

`tokens[0]:`

```
['for', 'a', 'movie', 'that', 'gets', 'no', 'respect', 'there', 'sure', 'are', 'a',
'lot', 'of', 'memorable', 'quotes', 'listed', 'for', 'this', 'gem', 'imagine', 'a',
'movie', 'where', 'joe', 'piscopo', 'is', 'actually', 'funny', 'maureen',
'stapleton', 'is', 'a', 'scene', 'stealer', 'the', 'moroni', 'character', 'is',
'an', 'absolute', 'scream', 'watch', 'for', 'alan', 'the', 'skipper', 'hale', 'jr',
'as', 'a', 'police', 'sgt']
```

Encoding

`seqs[0]:`

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 11, 12, 13, 14, 15, 1, 16, 17, 18, 2, 3, 19, 20,
21, 22, 23, 24, 25, 26, 22, 2, 27, 28, 29, 30, 31, 22, 32, 33, 34, 35, 1, 36, 29,
37, 38, 39, 40, 2, 41, 42]
```

Alignment (keep the last $w = 20$ tokens)

Do the same to test data

- Do the same operations to test data.
  - Step 1: tokenization.
  - Step 3: encoding.
  - Step 4: alignment.

Do the same to test data

- Do the same operations to test data.
  - Step 1: tokenization.
  - Step 3: encoding.
  - Step 4: alignment.
- Use the dictionary built on the training data.
  - Don’t build a dictionary on the test data!
  - The dictionary for training and test must be the same!
- Otherwise, this may happen:
  - In training data, index 23 refers to “good”.
  - In test data, index 23 refers to “mediocre”.

Word Embedding: Word to Vector

How to map word to vector?

Word	Index	
“movie”	1	
“good”	2	
“fun”	3	
“boring”	4	
…	…	

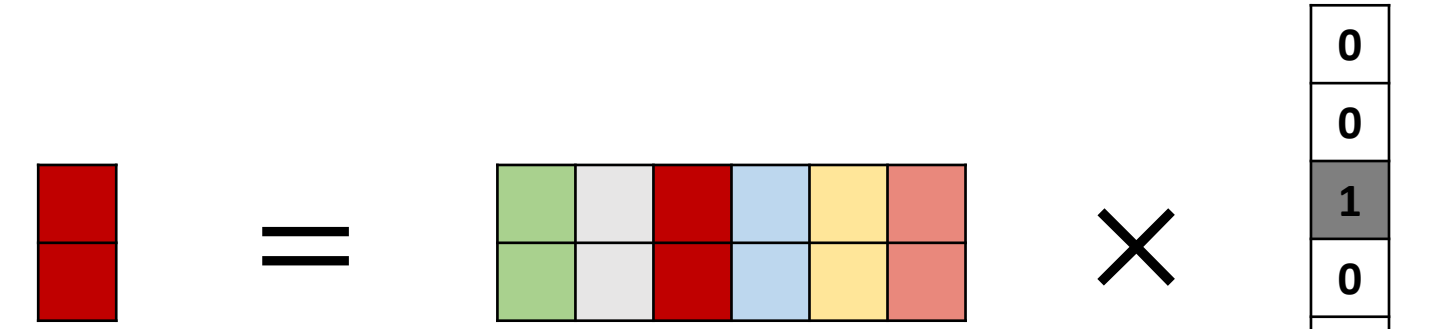
One-Hot Encoding

- First, represent words using one-hot vectors.
  - Suppose the dictionary contains v unique words (vocabulary$= v$).
  - Then the one-hot vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \cdots, \mathbf{e}_v$ are v-dimensional.

Word	Index	One-hot encoding
"movie"	1	$\mathbf{e}_1 = [1, 0, 0, 0, 0, \cdots, 0]$
"good"	2	$\mathbf{e}_2 = [0, 1, 0, 0, 0, \cdots, 0]$
"fun"	3	$\mathbf{e}_3 = [0, 0, 1, 0, 0, \cdots, 0]$
"boring"	4	$\mathbf{e}_4 = [0, 0, 0, 1, 0, \cdots, 0]$
…	…	…

Word Embedding

- Second, map the one-hot vectors to low-dimensional vectors by



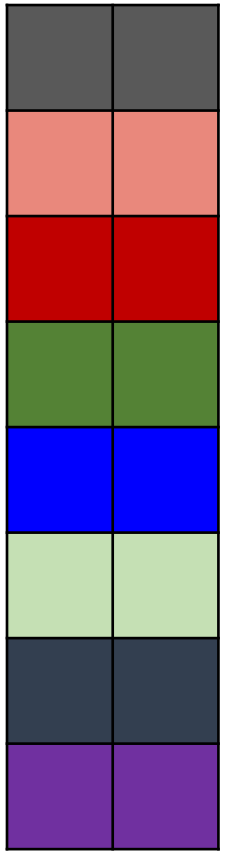
$\mathbf{x}_i$ $d \times 1$ $=$ $\mathbf{P}^T$ $d \times v$ $\times$ $\mathbf{e}_i$ $v \times 1$

- $\mathbf{P}$ is parameter matrix which can be learned from training data.
- $\mathbf{e}_i$ is the one-hot vector of the i-th word in dictionary.

How to interpret the parameter matrix?

Parameter matrix

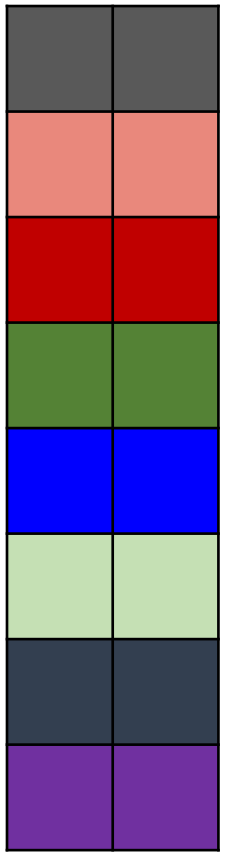
$$\mathbf{P} \in \mathbb{R}^{v \times d}$$



How to interpret the parameter matrix?

Parameter matrix

$$\mathbf{P} \in \mathbb{R}^{v \times d}$$



1: “movie”

2: “good”

3: “fun”

4: “boring”

5: “poor”

6: “mediocre”

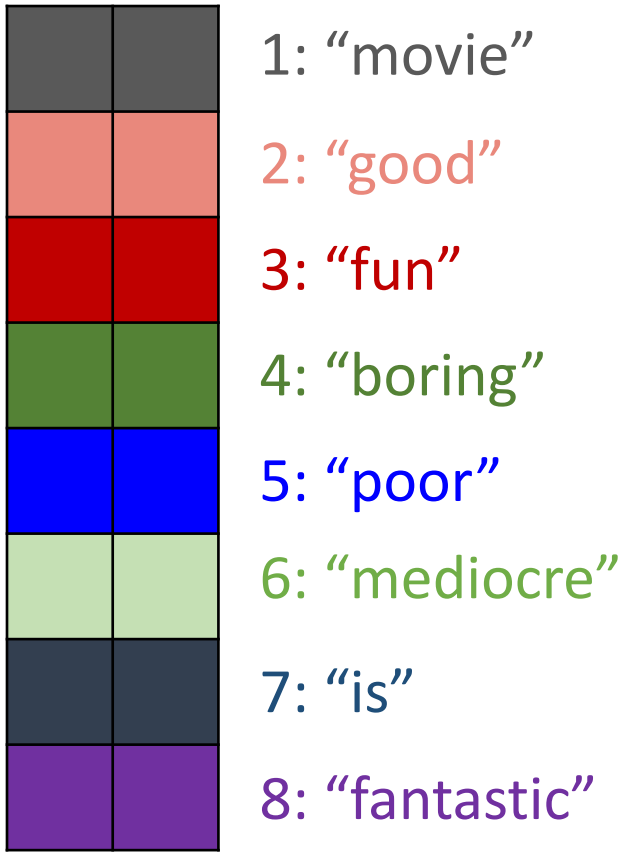
7: “is”

8: “fantastic”

How to interpret the parameter matrix?

Parameter matrix

$$\mathbf{P} \in \mathbb{R}^{v \times d}$$



1: “movie”

2: “good”

3: “fun”

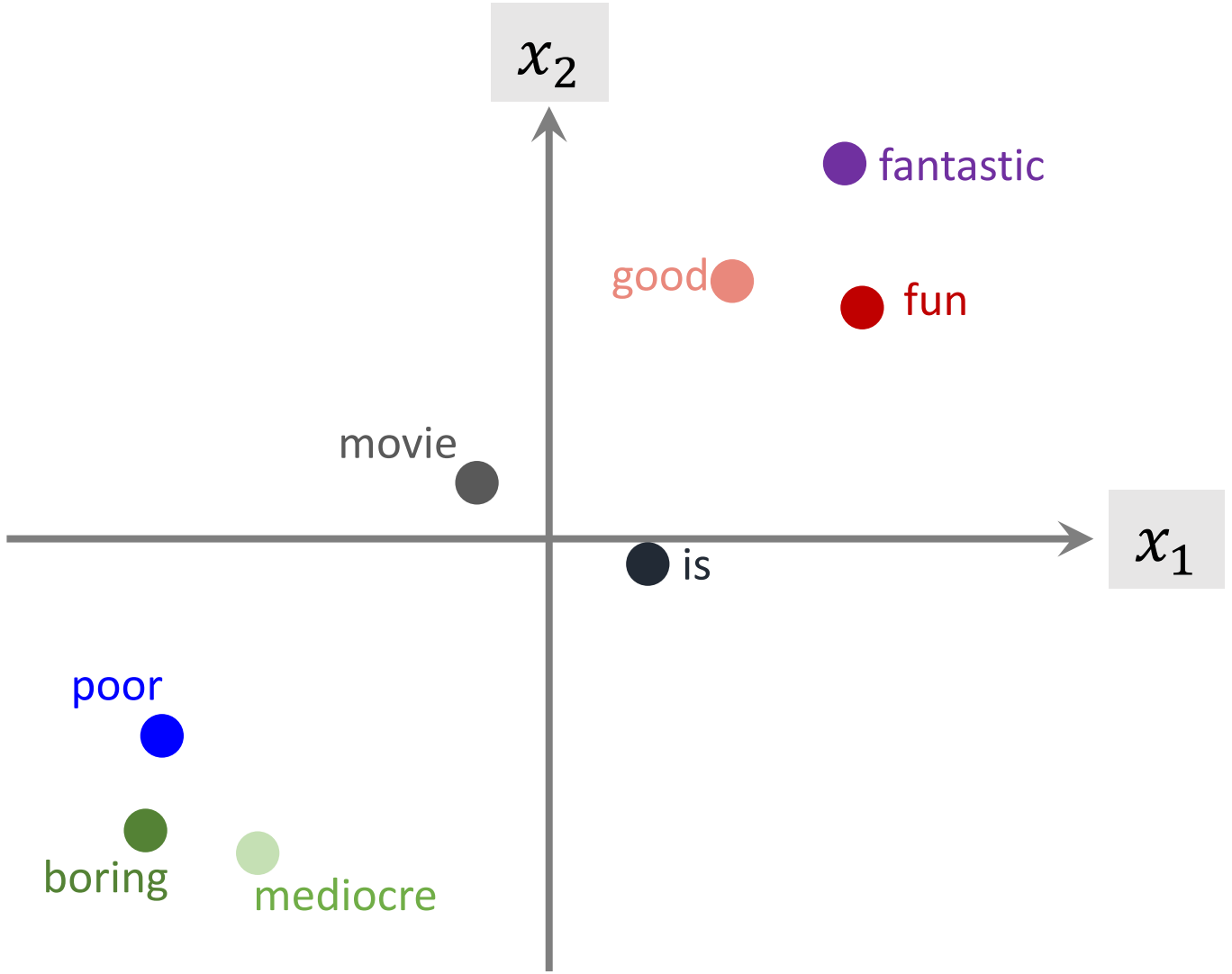
4: “boring”

5: “poor”

6: “mediocre”

7: “is”

8: “fantastic”



```
from keras.models import Sequential
from keras.layers import Flatten, Dense, Embedding

embedding_dim = 8

model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
```

$v = 10K$ $d = 8$ $= 20$

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 20, 8)             80000
```

word_num

embedding_dim

= vocabulary ×embedding_dim

Logistic Regression for Binary Classification

```python
from keras.models import Sequential
from keras.layers import Flatten, Dense, Embedding

embedding_dim = 8

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 20, 8)             80000
_________________________________________________________________
flatten_1 (Flatten)          (None, 160)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 161
=================================================================
Total params: 80,161
Trainable params: 80,161
Non-trainable params: 0
```

```
from keras import optimizers

epochs = 50

model.compile(optimizer=optimizers.RMSprop(lr=0.0001),
              loss='binary_crossentropy', metrics=['acc'])
```

```
from keras import optimizers

epochs = 50

model.compile(optimizer=optimizers.RMSprop(lr=0.0001),
              loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train, epochs=epochs,
                    batch_size=32, validation_data=(x_valid, y_valid))
```

- The training set is randomly split to a training set and a validation set.
- 80% for training and 20% for validation.

- x_train: $20{,}000 \times 20$ matrix
- X_valid: $5{,}000 \times 20$ matrix

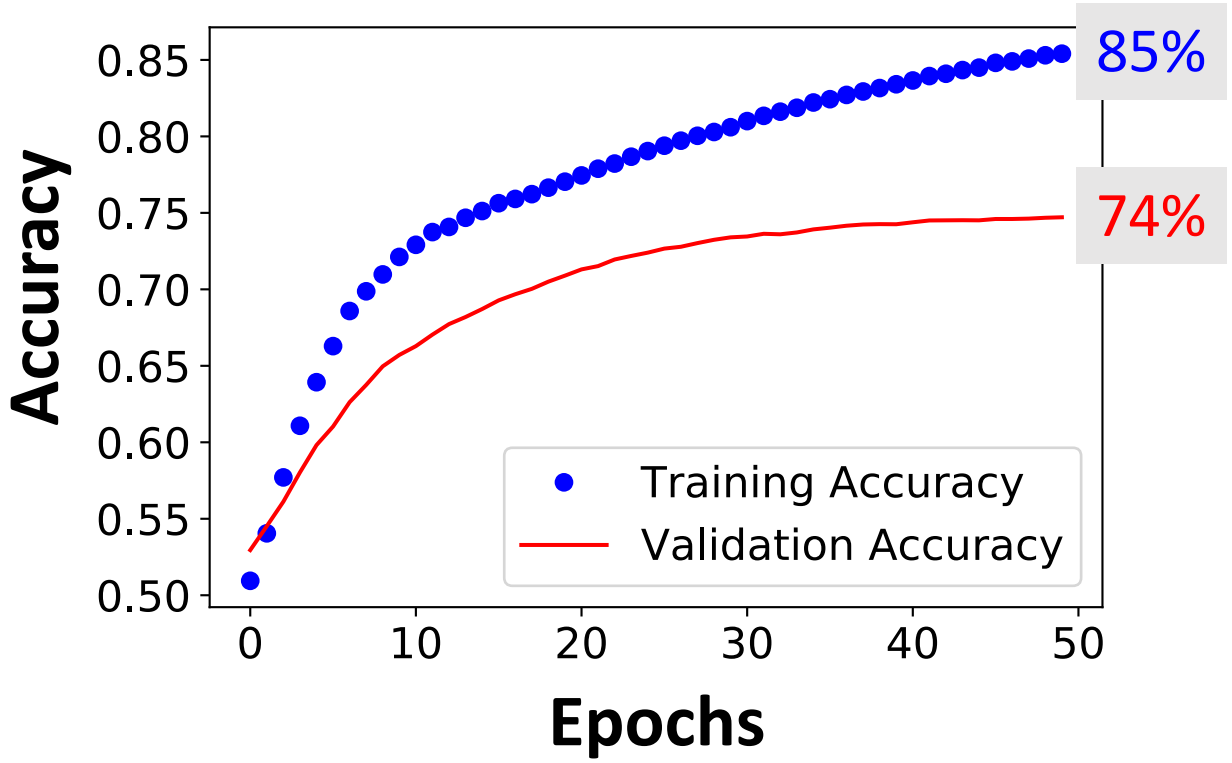
```python
from keras import optimizers

epochs = 50

model.compile(optimizer=optimizers.RMSprop(lr=0.0001),
              loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train, epochs=epochs,
                    batch_size=32, validation_data=(x_valid, y_valid))
```

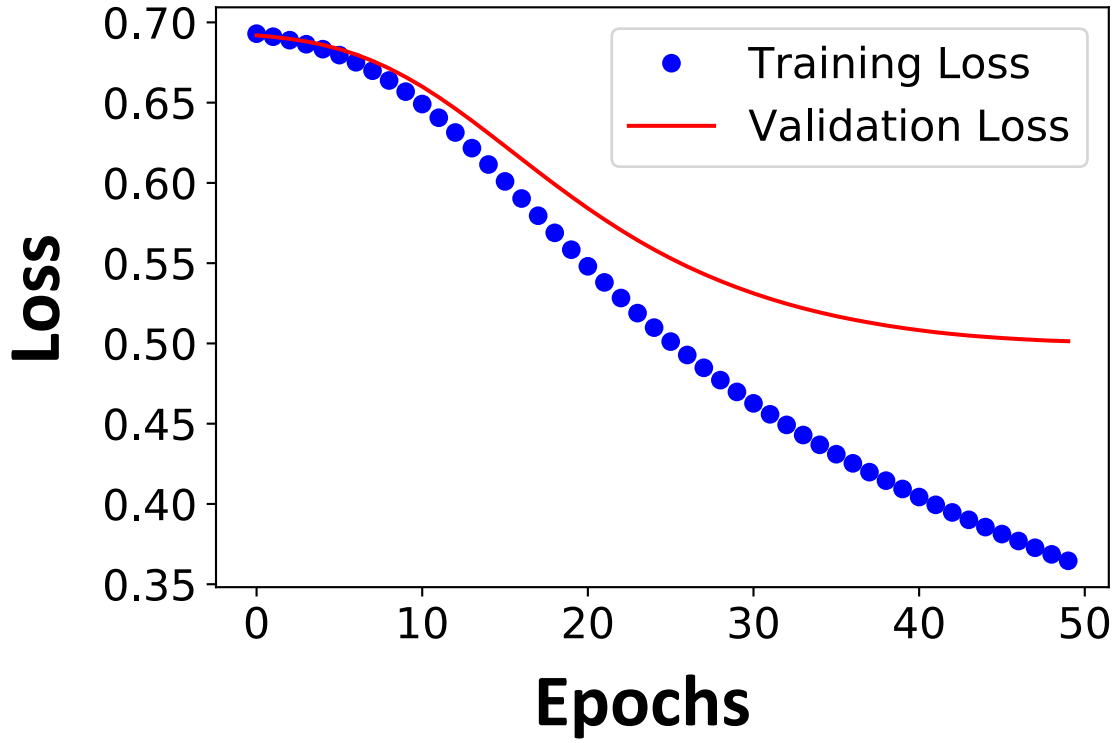
```
Epoch 1/50
12500/12500 [==============================] - 1s 74us/step - loss: 0.6930 - acc: 0.5094 - val_loss: 0.6919 - val_acc: 0.5295
Epoch 2/50
12500/12500 [==============================] - 1s 60us/step - loss: 0.6911 - acc: 0.5405 - val_loss: 0.6910 - val_acc: 0.5452
Epoch 3/50
12500/12500 [==============================] - 1s 57us/step - loss: 0.6889 - acc: 0.5770 - val_loss: 0.6898 - val_acc: 0.5612
```

-
-
-

```
Epoch 49/50
12500/12500 [==============================] - 1s 56us/step - loss: 0.3686 - acc: 0.8530 - val_loss: 0.5017 - val_acc: 0.7468
Epoch 50/50
12500/12500 [==============================] - 1s 56us/step - loss: 0.3646 - acc: 0.8541 - val_loss: 0.5014 - val_acc: 0.7471
```

Performance on training and validation sets





Performance on test set

```
loss_and_acc = model.evaluate(x_test, labels_test)
print('loss = ' + str(loss_and_acc[0]))
print('acc = ' + str(loss_and_acc[1]))
```

```
25000/25000 [==============================] - 0s 18us/step
loss = 0.5025235502243042
acc = 0.74928
```

- About 75% accuracy on the test set.
- Not bad, because we use only the last 20 words in each movie review. (`word_num=20`)

Summary

Texts to Sequences

`texts[i]:`

"For a movie that gets no respect there sure are a lot of memorable quotes listed for this gem. Imagine a movie where Joe Piscopo is actually funny! Maureen Stapleton is a scene stealer. The Moroni character is an absolute scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt."

Tokenization

`tokens[i]:`

```
['for', 'a', 'movie', 'that', 'gets', 'no', 'respect', 'there', 'sure', 'are', 'a',
'lot', 'of', 'memorable', 'quotes', 'listed', 'for', 'this', 'gem', 'imagine', 'a',
'movie', 'where', 'joe', 'piscopo', 'is', 'actually', 'funny', 'maureen',
'stapleton', 'is', 'a', 'scene', 'stealer', 'the', 'moroni', 'character', 'is',
'an', 'absolute', 'scream', 'watch', 'for', 'alan', 'the', 'skipper', 'hale', 'jr',
'as', 'a', 'police', 'sgt']
```

Encoding

`seqs[i]:`

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 11, 12, 13, 14, 15, 1, 16, 17, 18, 2, 3, 19, 20,
21, 22, 23, 24, 25, 26, 22, 2, 27, 28, 29, 30, 31, 22, 32, 33, 34, 35, 1, 36, 29,
37, 38, 39, 40, 2, 41, 42]
```

Texts to Sequences

`texts[i]:`

"For a movie that gets no respect there sure are a lot of memorable quotes listed for this gem. Imagine a movie where Joe Piscopo is actually funny! Maureen Stapleton is a scene stealer. The Moroni character is an absolute scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt."

Tokenization

`tokens[i]:`

```
['for', 'a', 'movie', 'that', 'gets', 'no', 'respect', 'there', 'sure', 'are', 'a',
'lot', 'of', 'memorable', 'quotes', 'listed', 'for', 'this', 'gem', 'imagine', 'a',
'movie', 'where', 'joe', 'piscopo', 'is', 'actually', 'funny', 'maureen',
'stapleton', 'is', 'a', 'scene', 'stealer', 'the', 'moroni', 'character', 'is',
'an', 'absolute', 'scream', 'watch', 'for', 'alan', 'the', 'skipper', 'hale', 'jr',
'as', 'a', 'police', 'sgt']
```

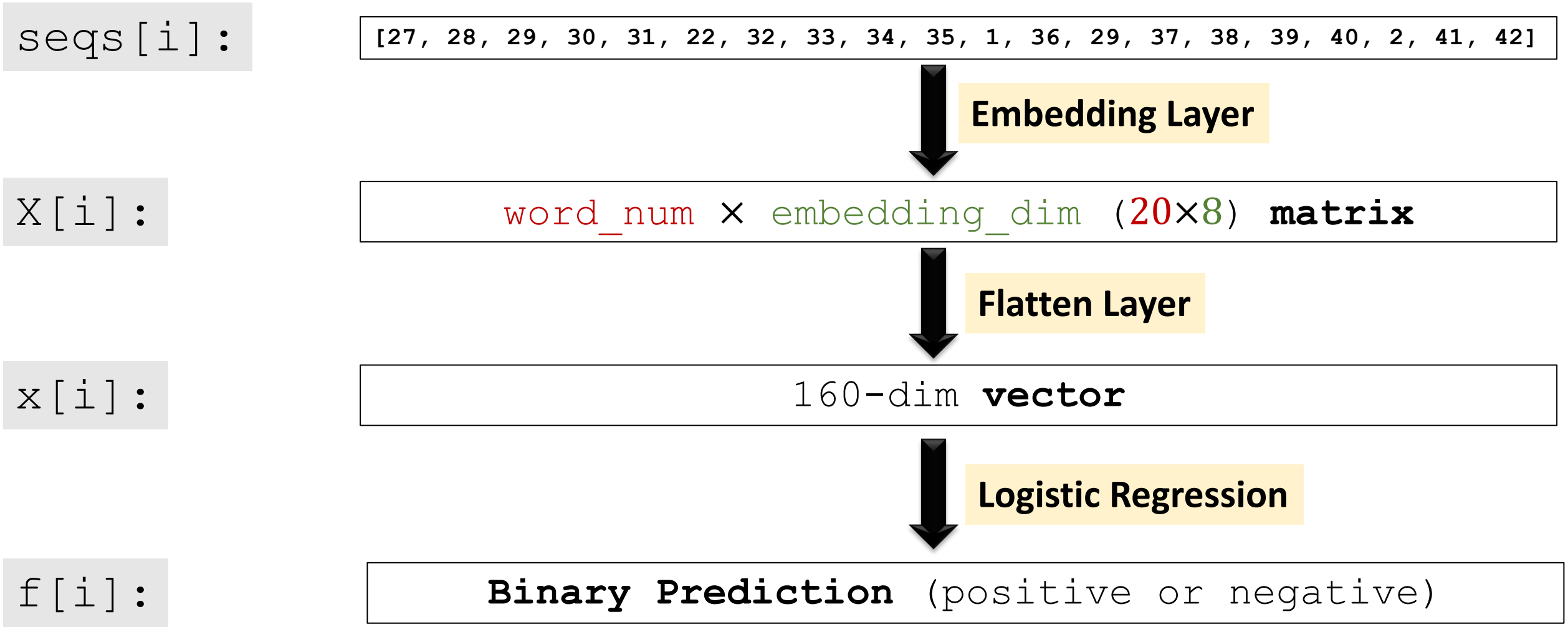
Encoding

`seqs[i]:`

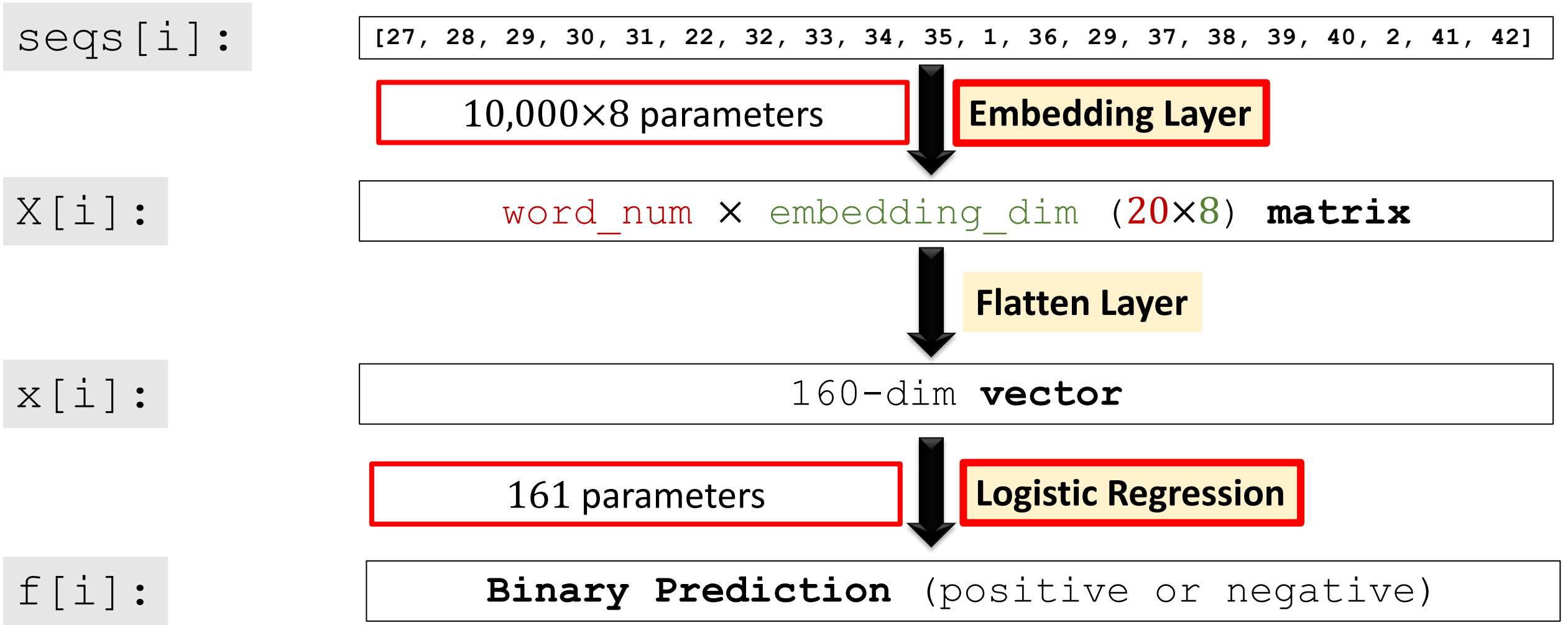
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 2, 11, 12, 13, 14, 15, 1, 16, 17, 18, 2, 3, 19, 20,
21, 22, 23, 24, 25, 26, 22, 2, 27, 28, 29, 30, 31, 22, 32, 33, 34, 35, 1, 36, 29,
37, 38, 39, 40, 2, 41, 42]
```

Alignment (keep the last $w = 20$ tokens)

Logistic Regression for Sentiment Analysis

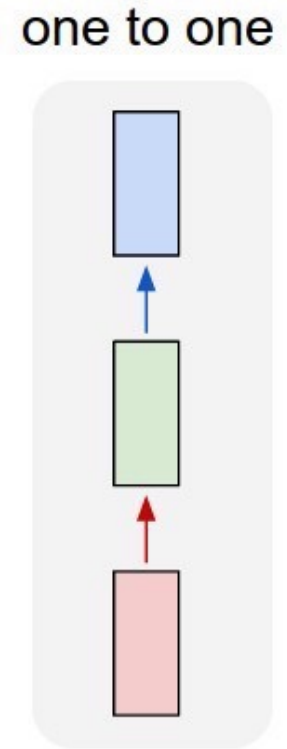


Logistic Regression for Sentiment Analysis



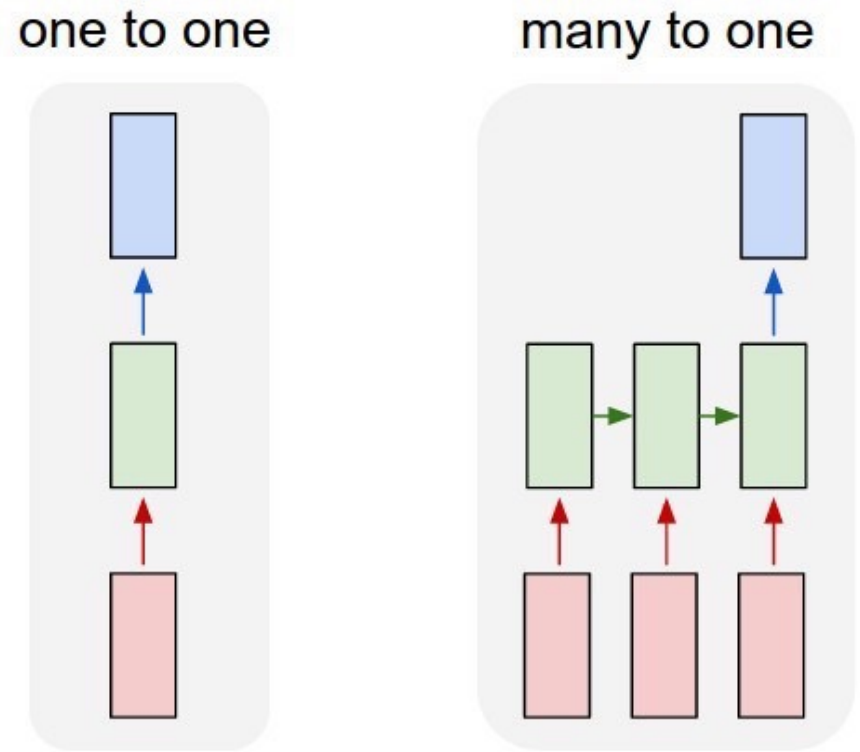
Recurrent Neural Networks (RNNs)

How to model sequential data?



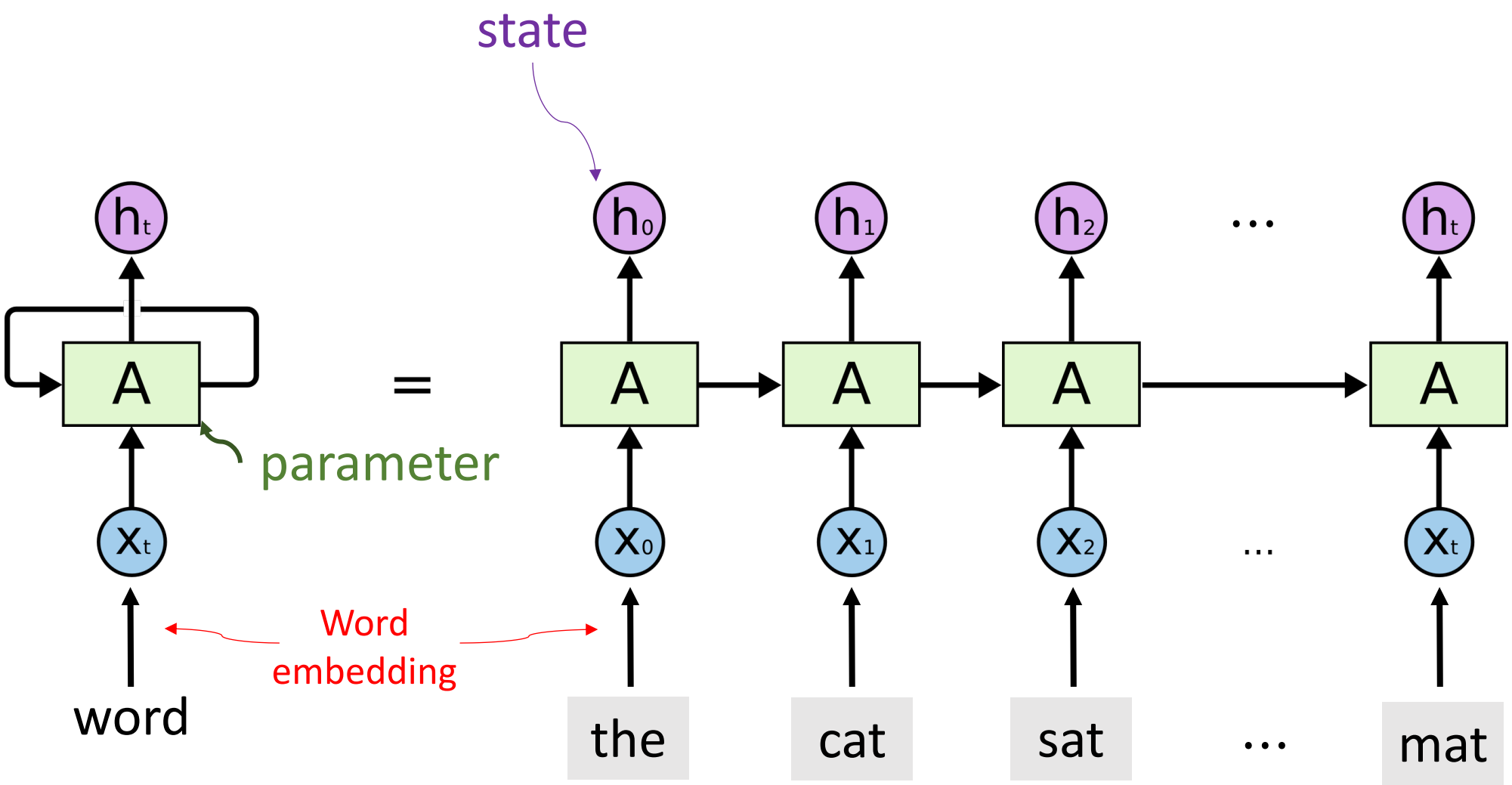
- Limitations of FC Nets and ConvNets:
  - Process a paragraph as a whole.
  - Fixed-size input (e.g., image).
  - Fixed-size output (e.g., predicted probabilities).

How to model sequential data?



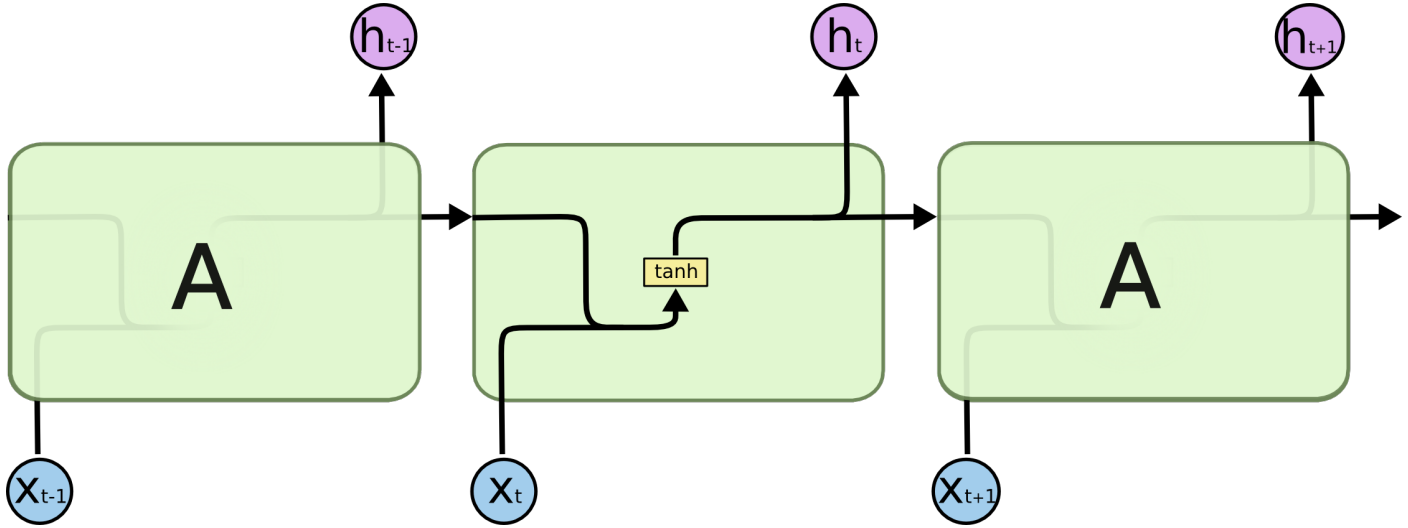
- Limitations of FC Nets and ConvNets:
  - Process a paragraph as a whole.
  - Fixed-size input (e.g., image).
  - Fixed-size output (e.g., predicted probabilities).
- RNNs are better ways to model the sequential data (e.g., text, speech, and time series).

Recurrent Neural Networks (RNNs)

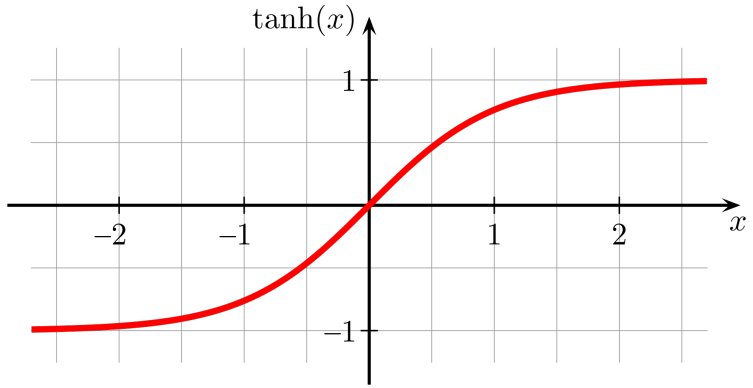


Simple RNN Model

Simple RNN



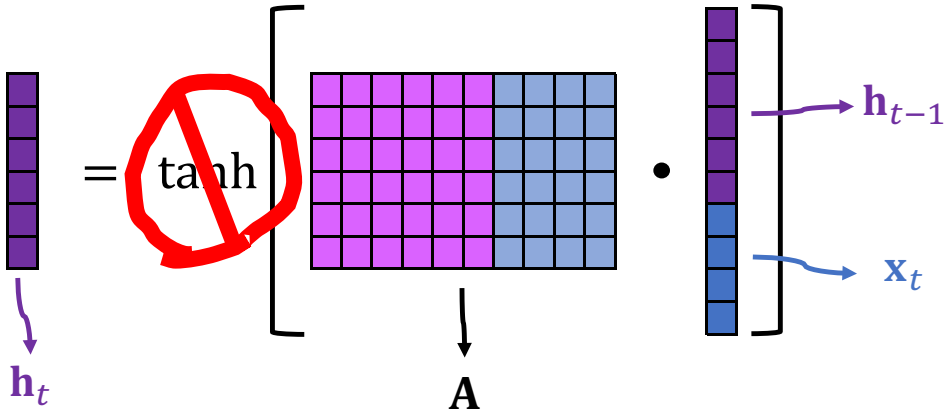
$$\mathbf{h}_t = \tanh\left[\mathbf{A} \cdot \begin{bmatrix}\mathbf{h}_{t-1} \\ \mathbf{x}_t\end{bmatrix}\right]$$

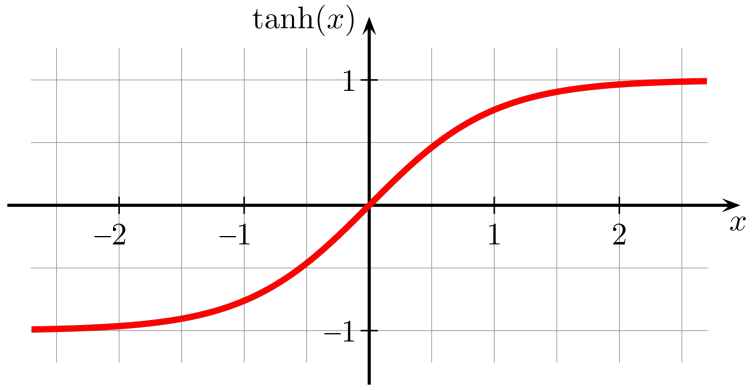


hyperbolic tangent function

Simple RNN

Question: Why do we need the tanh function?



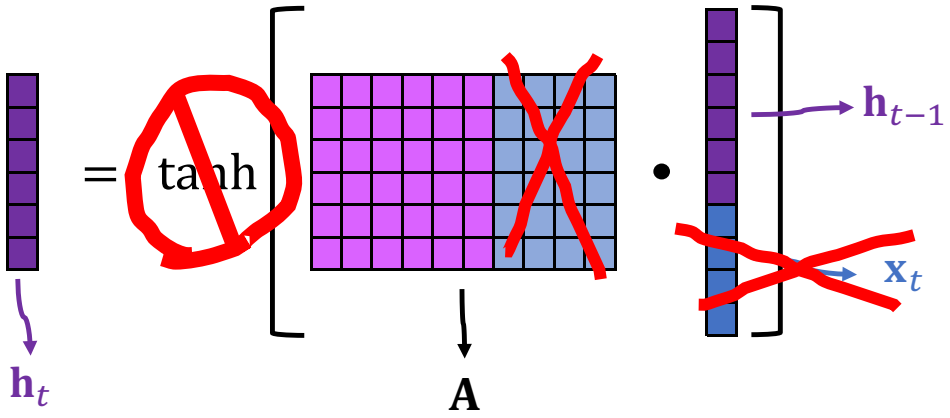


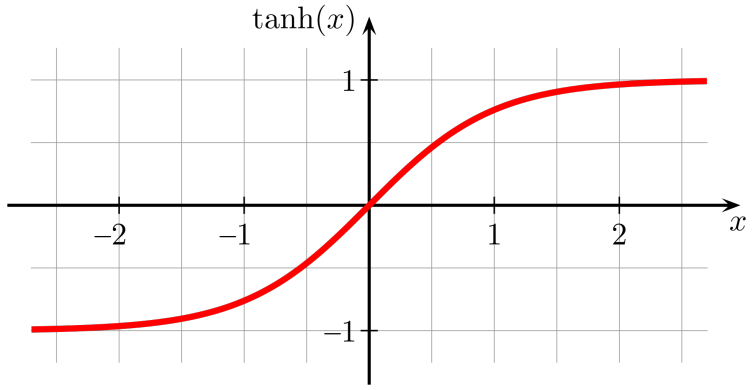
hyperbolic tangent function

Simple RNN

Question: Why do we need the tanh function?

- Suppose $\mathbf{x}_0 = \cdots = \mathbf{x}_{100} = 0$.



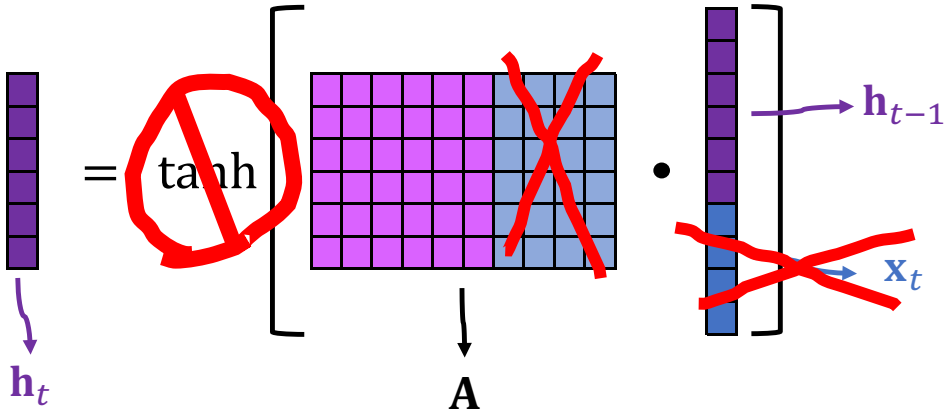


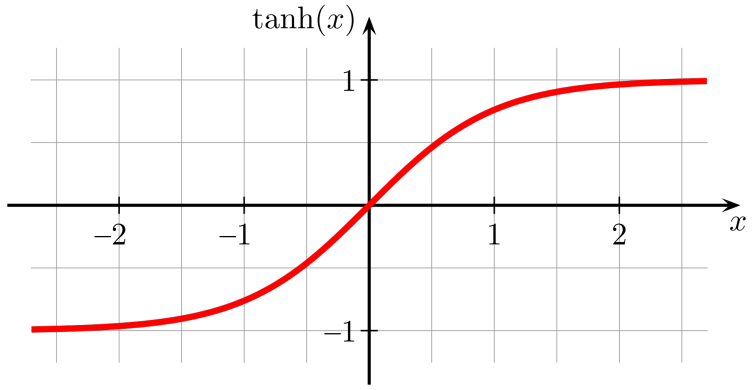
hyperbolic tangent function

Simple RNN

Question: Why do we need the tanh function?

- Suppose $\mathbf{x}_0 = \cdots = \mathbf{x}_{100} = 0$.
- $\mathbf{h}_{100} = \mathbf{A}\mathbf{h}_{99} = \mathbf{A}^2\mathbf{h}_{98} = \cdots = \mathbf{A}^{100}\mathbf{h}_0$.
- What will happen if $\lambda_{\max}(\mathbf{A}) = 0.9$?
- What will happen if $\lambda_{\max}(\mathbf{A}) = 1.2$?





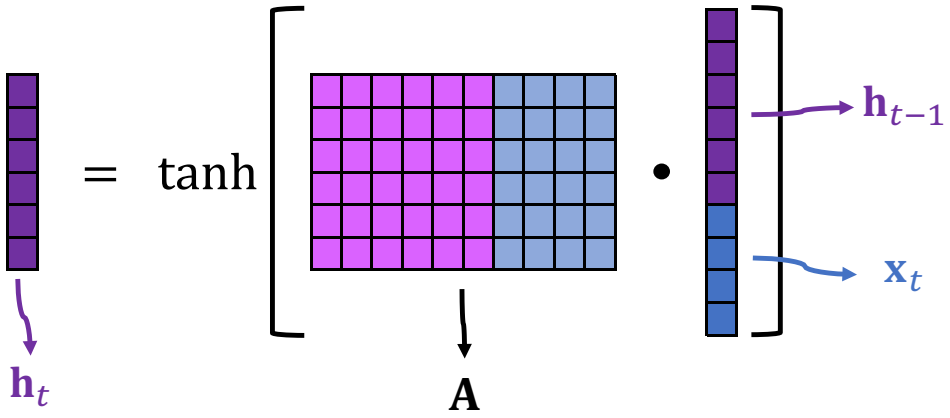
hyperbolic tangent function

Simple RNN

Trainable parameters: matrix **A**

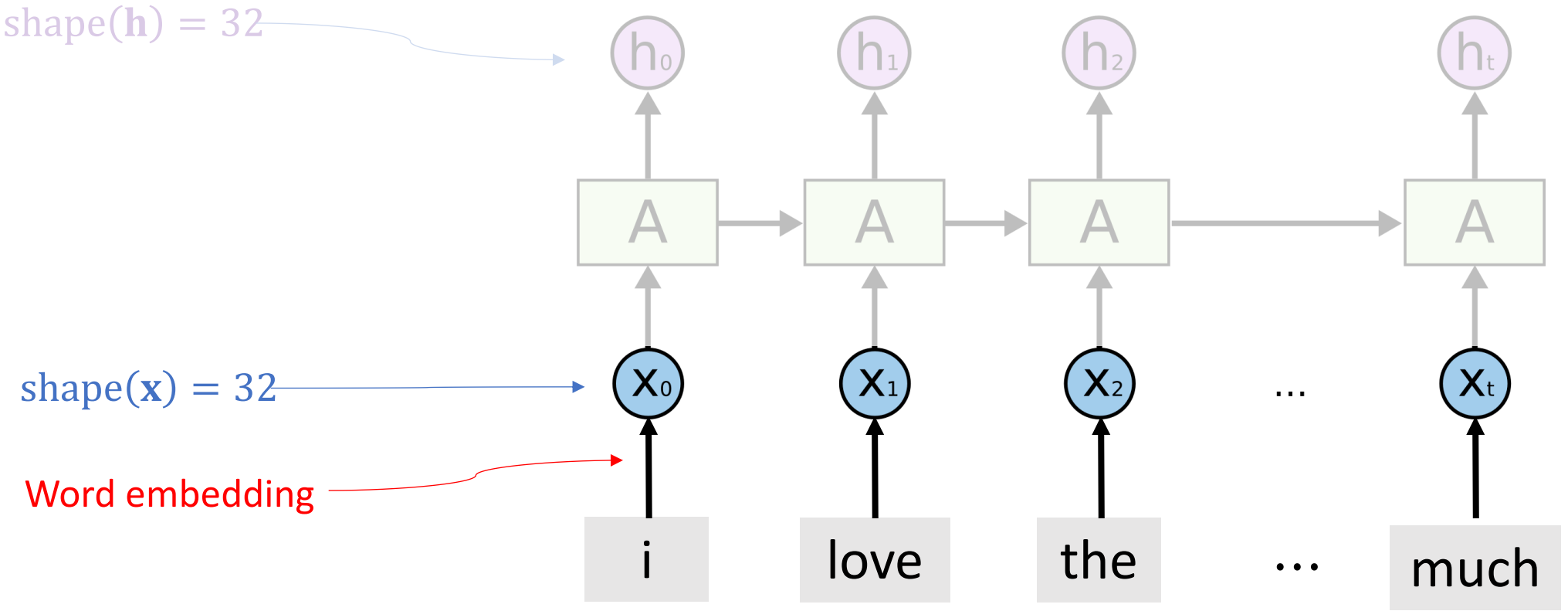
- #rows of **A**: shape(**h**)
- #cols of **A**: shape(**h**)+shape(**x**)

- Total #parameter: shape(**h**)× [shape(**h**)+shape(**x**)]

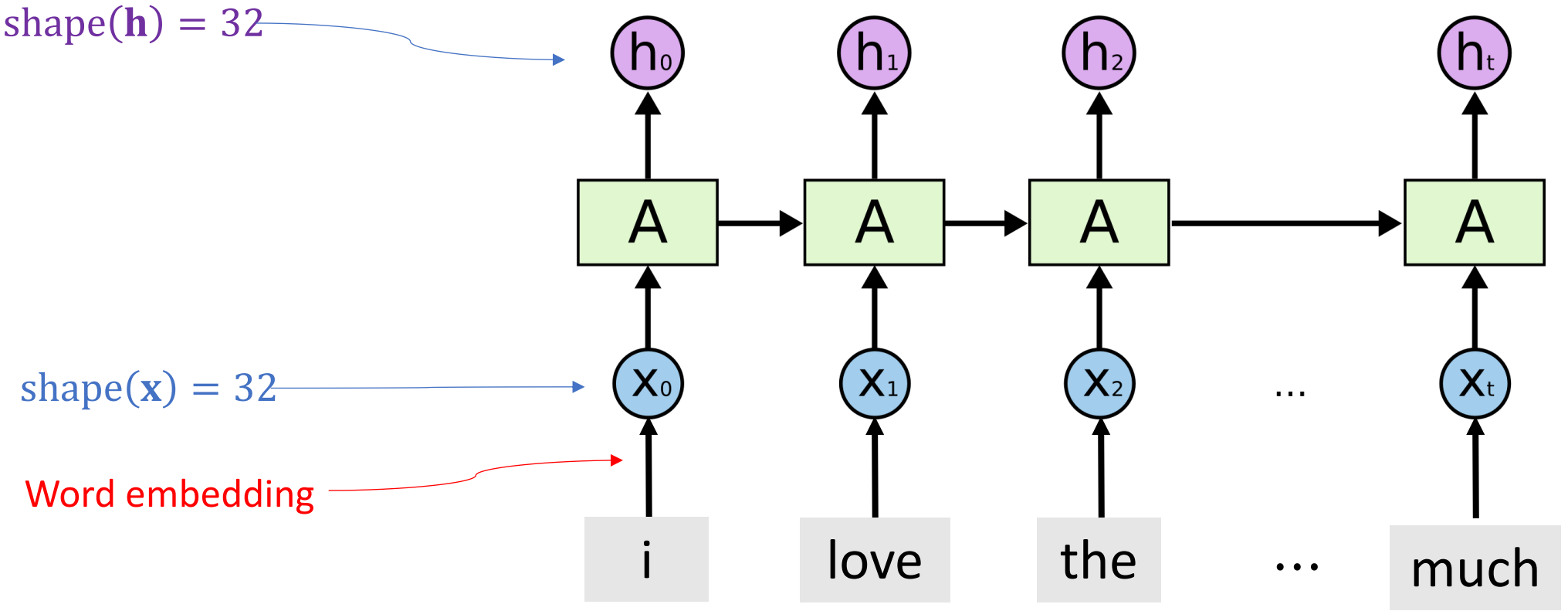


Simple RNN for Movie Review Analysis

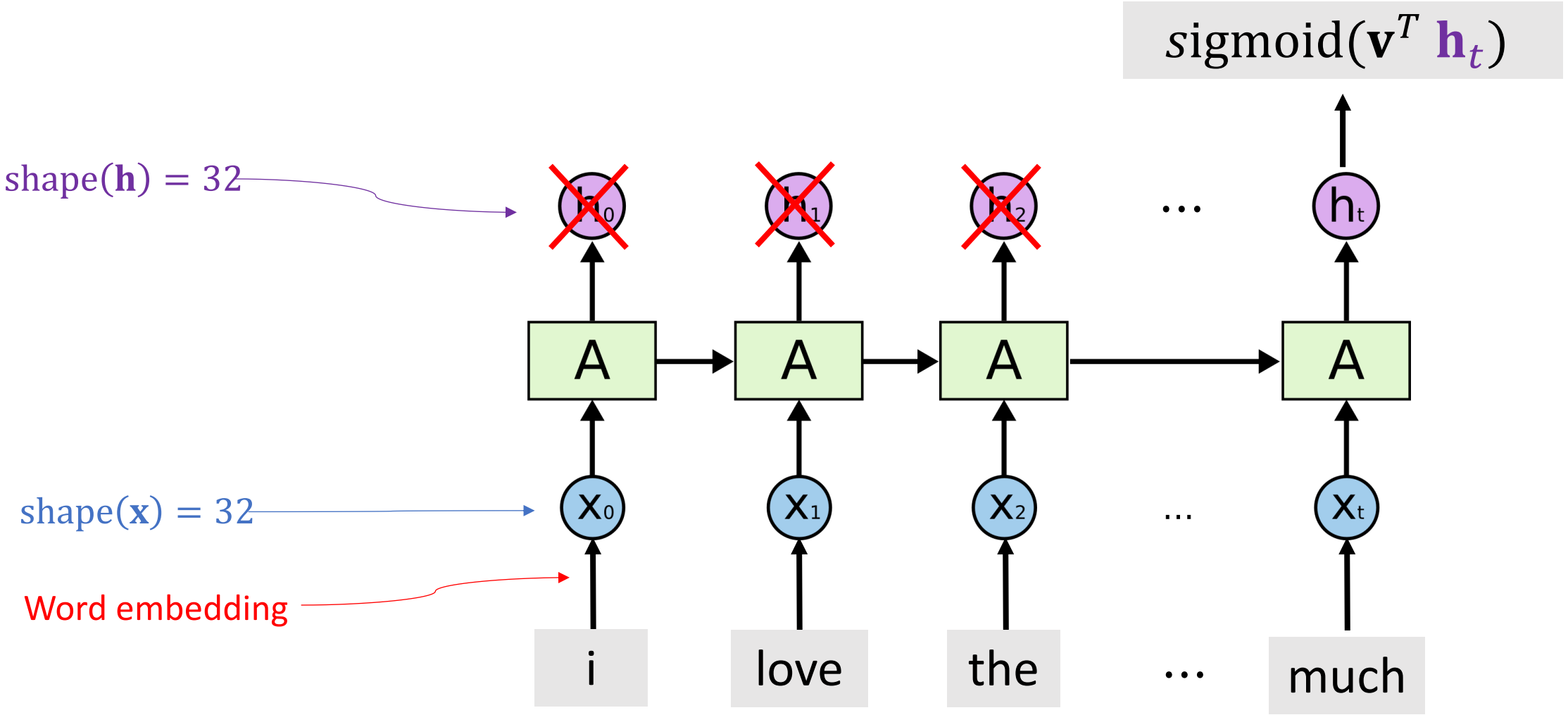
Simple RNN for IMDB Review



Simple RNN for IMDB Review



Simple RNN for IMDB Review



Simple RNN for IMDB Review

```
from keras.models import Sequential
from keras.layers import SimpleRNN, Embedding, Dense

vocabulary = 10000    #unique words in the dictionary
embedding_dim = 32    shape(x) = 32
word_num = 500        sequence length
state_dim = 32        shape(h) = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(SimpleRNN(state_dim, return_sequences=False))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Only return the last state h_t.

Simple RNN for IMDB Review

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 32)	320000
simple_rnn_1 (SimpleRNN)	(None, 32)	2080
dense_1 (Dense)	(None, 1)	33

Total params: 322,113
Trainable params: 322,113
Non-trainable params: 0

#parameters in RNN:

$$2080 = 32 \times (32 + 32) + 32$$

$\text{shape}(\mathbf{h}) = 32$

$\text{shape}(\mathbf{x})$

Simple RNN for IMDB Review

```
from keras import optimizers

epochs = 3

model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss='binary_crossentropy', metrics=['acc'])
history = model.fit(x_train, y_train, epochs=epochs,
                    batch_size=32, validation_data=(x_valid, y_valid))
```

Early stopping alleviates overfitting

```
Train on 20000 samples, validate on 5000 samples
Epoch 1/3
20000/20000 [==============================] - 65s 3ms/step - loss: 0.5514 - acc: 0.6959 - val_loss: 0.4095 - val_acc: 0.8176
Epoch 2/3
20000/20000 [==============================] - 66s 3ms/step - loss: 0.3336 - acc: 0.8620 - val_loss: 0.3296 - val_acc: 0.8658
Epoch 3/3
20000/20000 [==============================] - 65s 3ms/step - loss: 0.2774 - acc: 0.8918 - val_loss: 0.3569 - val_acc: 0.8428
```

Simple RNN for IMDB Review

```
loss_and_acc = model.evaluate(x_test, labels_test)
print('loss = ' + str(loss_and_acc[0]))
print('acc = ' + str(loss_and_acc[1]))
```

```
25000/25000 [==============================] - 21s 829us/step
loss = 0.36507524153709414
acc = 0.84364
```

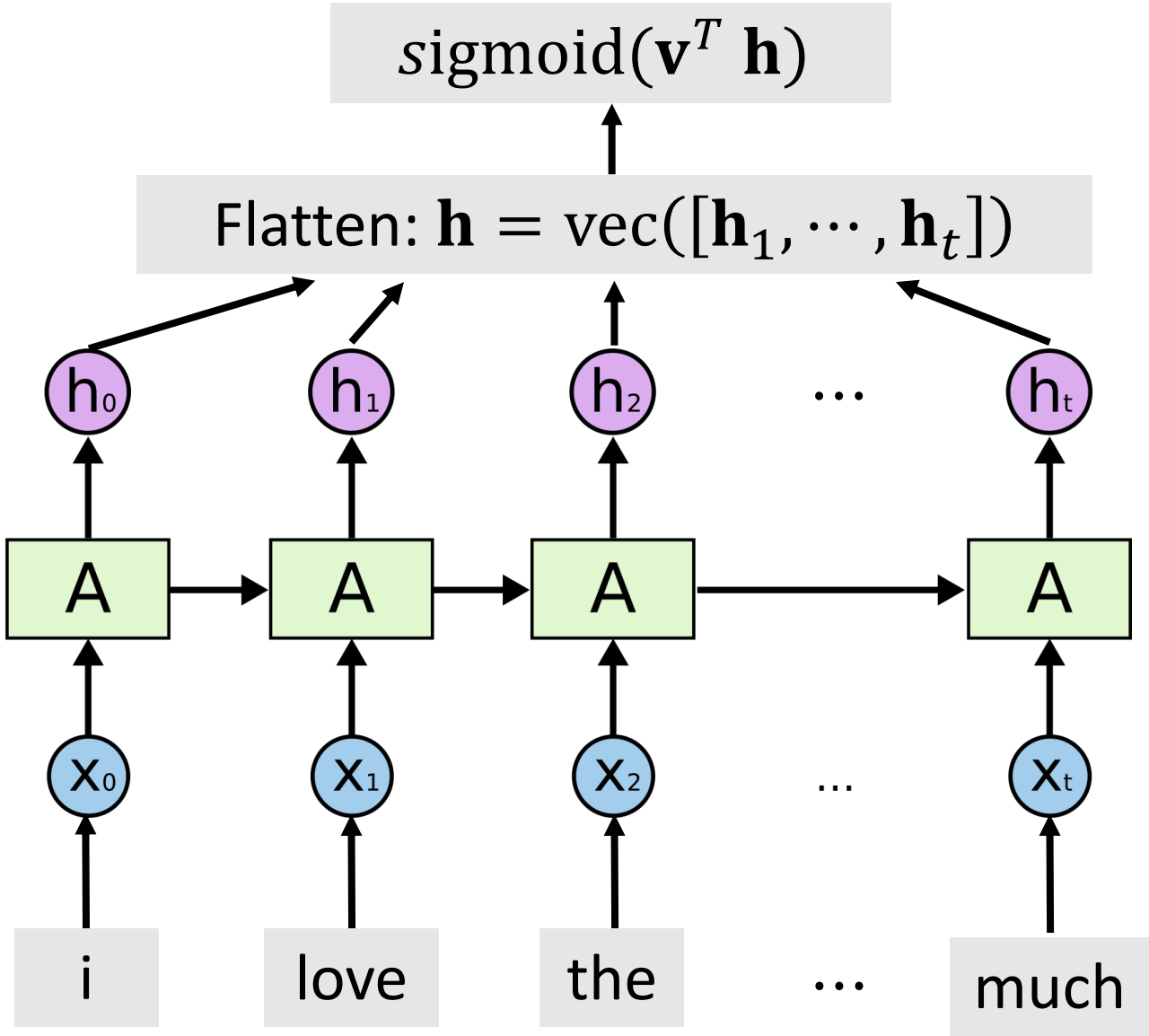
Higher than a naïve shallow model (whose test accuracy is about 75%).

Simple RNN for IMDB Review

- Training Accuracy: 89.2%
- Validation Accuracy: 84.3%
- Test Accuracy: 84.4%

Higher than a naïve shallow model (whose test accuracy is about 75%).

Simple RNN for IMDB Review



Simple RNN for IMDB Review

```
from keras.models import Sequential
from keras.layers import SimpleRNN, Embedding, Dense

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(SimpleRNN(state_dim, return_sequences=True))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Return all the states $h_1, \cdots, h_t$.

Simple RNN for IMDB Review

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 500, 32)           320000
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, 500, 32)           2080
_________________________________________________________________
flatten_2 (Flatten)          (None, 16000)             0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 16001
=================================================================
Total params: 338,081
Trainable params: 338,081
Non-trainable params: 0
```

Simple RNN for IMDB Review

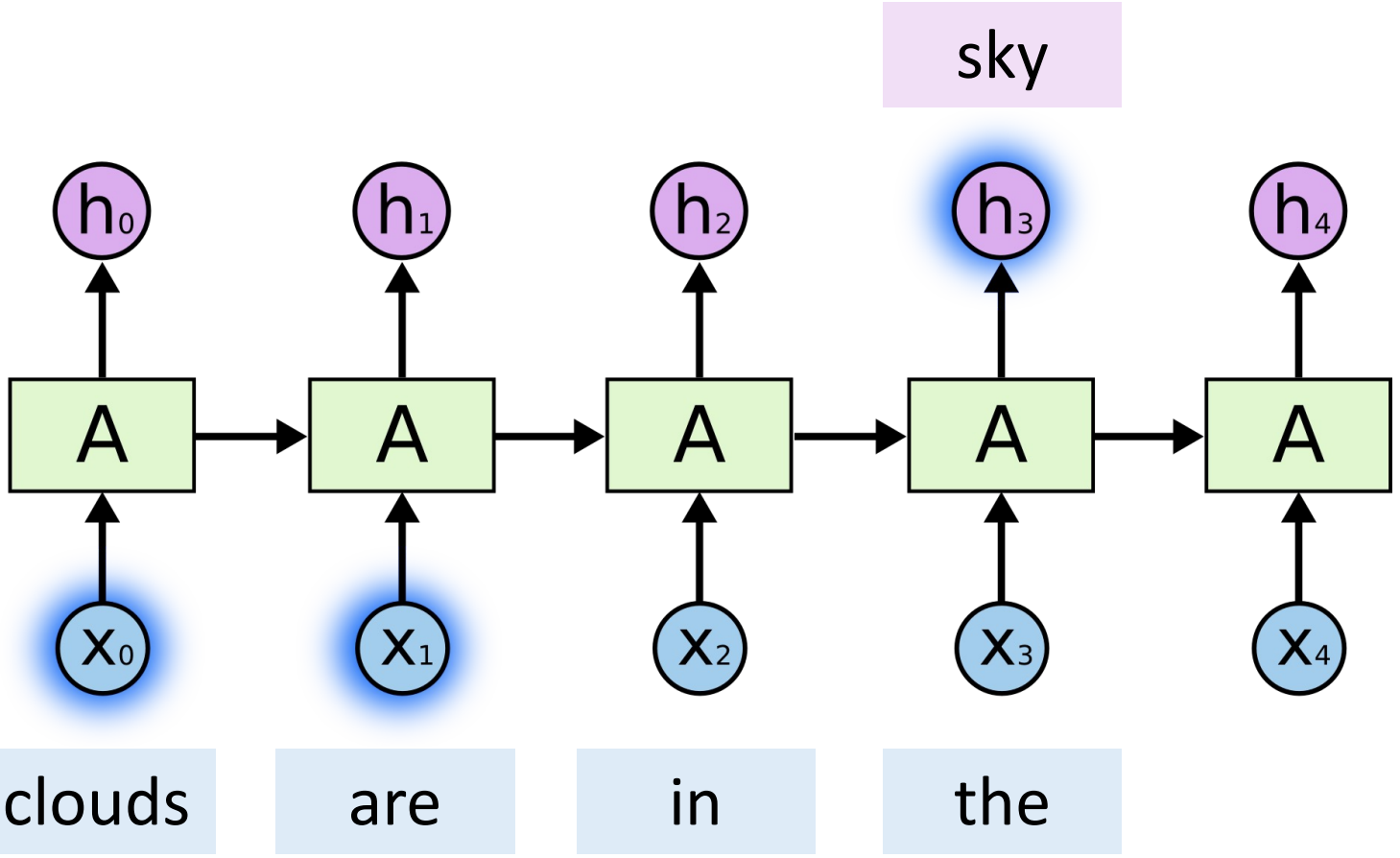
- Training Accuracy: 96.3%
- Validation Accuracy: 85.4%
- Test Accuracy: 84.7%

Not really better than using only the final state (whose accuracy is 84.4%).

Shortcomings of SimpleRNN

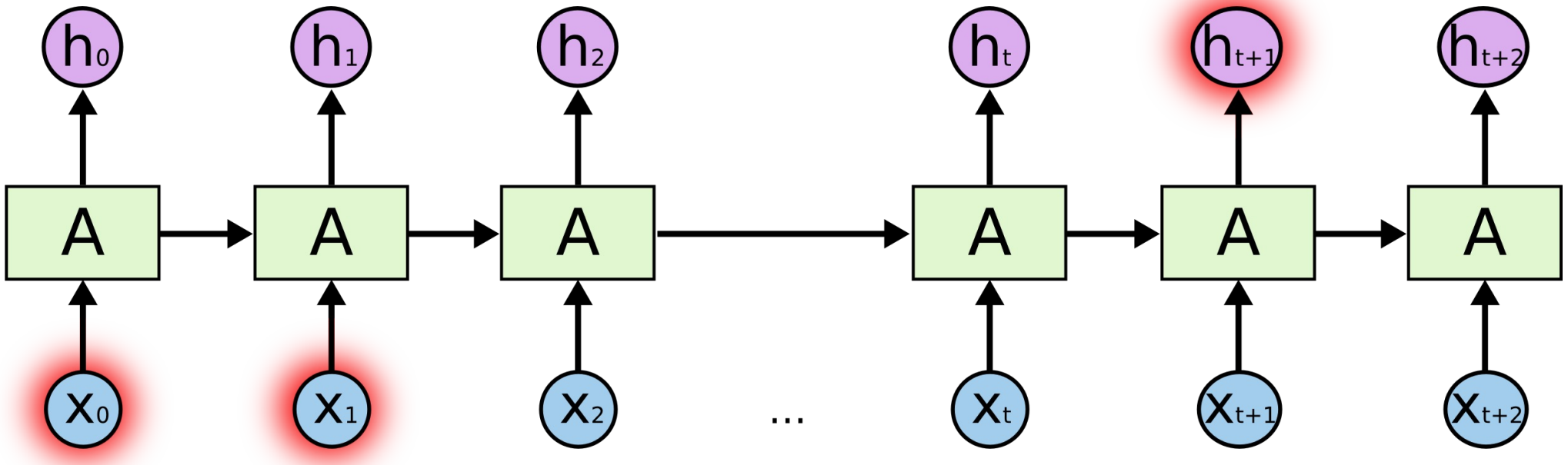
SimpleRNN is good at short-term dependence.

Predicted next words:



Figures are from Christopher Olah's blog: Understanding LSTM Networks.

SimpleRNN is bad at long-term dependence.



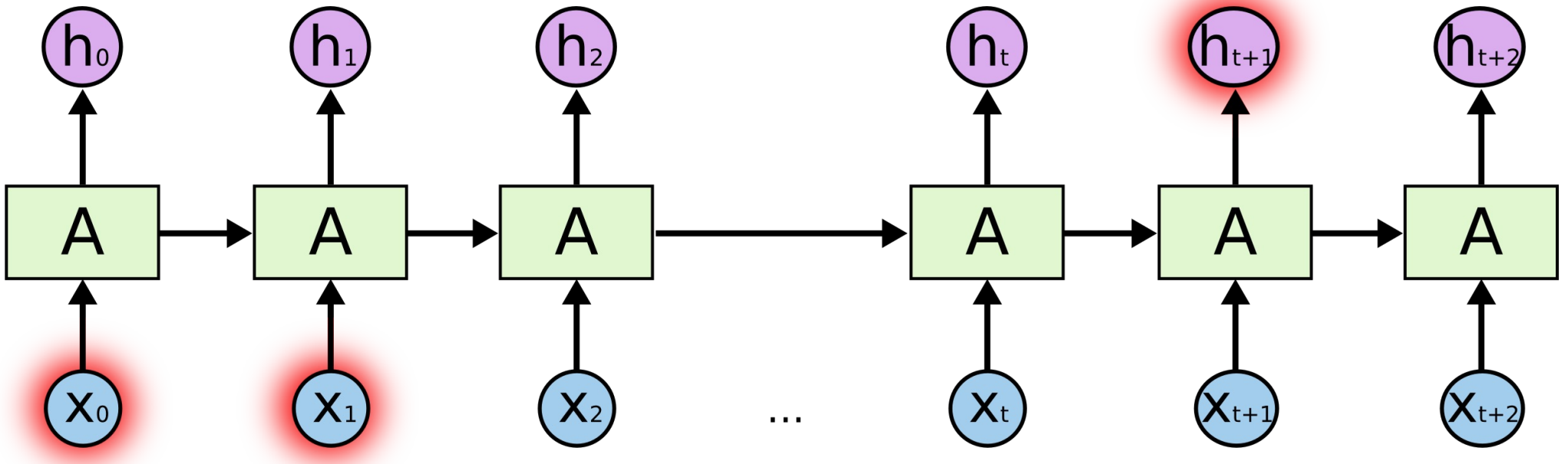
$\mathbf{h}_{100}$ is almost irrelevant to $\mathbf{x}_1$: $\frac{\partial\, \mathbf{h}_{100}}{\partial\, \mathbf{x}_1}$ is near zero.

Figures are from Christopher Olah's blog: Understanding LSTM Networks.

SimpleRNN is bad at long-term dependence.

Predicted next words:

Chinese



Input text:

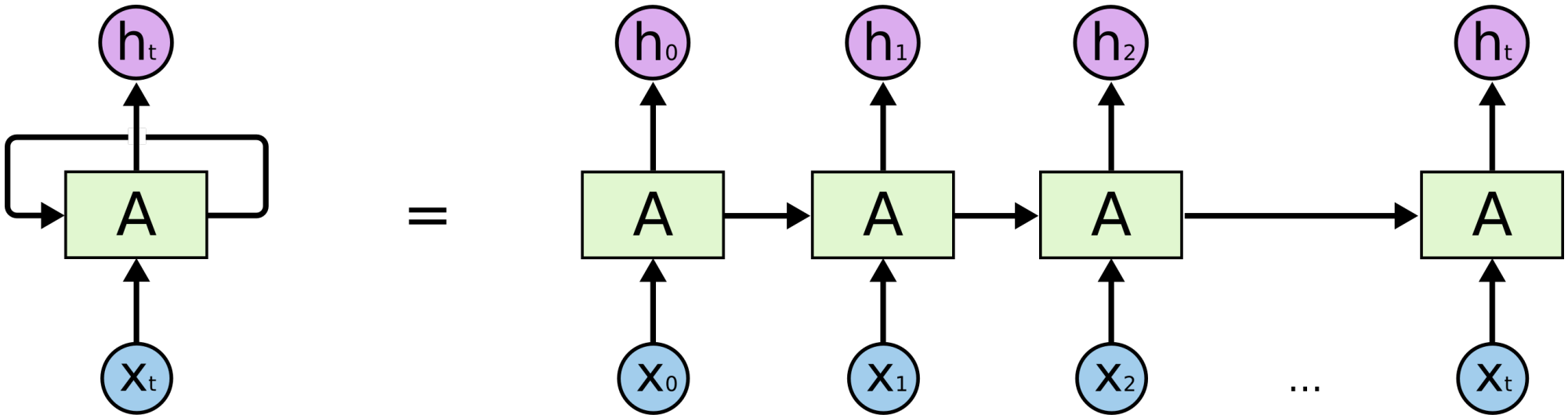
in | China | ... | speak | fluent

Figures are from Christopher Olah's blog: Understanding LSTM Networks.

Summary

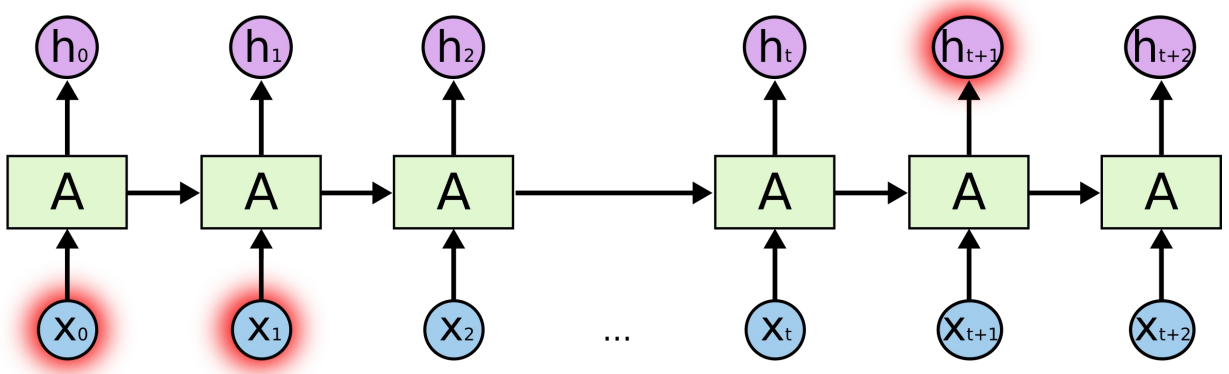
Summary

- RNN for text, speech, and time series data.
- Hidden state $\mathbf{h}_t$ aggregates information in the inputs $\mathbf{x}_0, \cdots, \mathbf{x}_t$.



Summary

- RNN for text, speech, and time series data.
- Hidden state $\mathbf{h}_t$ aggregates information in the inputs $\mathbf{x}_0, \cdots, \mathbf{x}_t$.
- RNNs can forget early inputs.
  - It forgets what it has seen early on.
  - If t is large, $\mathbf{h}_t$ is almost irrelevant to $\mathbf{x}_0$.



Number of Parameters

- SimpleRNN has a parameter matrix (and perhaps an intercept vector).
- Shape of the parameter matrix is

$$\texttt{shape(h)} \times \texttt{[shape(h)+shape(x)]}.$$

Number of Parameters

- SimpleRNN has a parameter matrix (and perhaps an intercept vector).
- Shape of the parameter matrix is

$$\texttt{shape}(\mathbf{h}) \times [\texttt{shape}(\mathbf{h}) + \texttt{shape}(\mathbf{x})].$$

- Only one such parameter matrix, no matter how long the sequence is.

Thank You!