

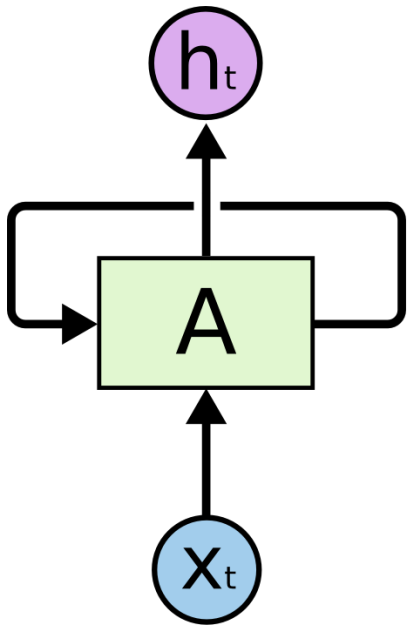
رسالة محمد

شبکه‌های عصبی بازگشتی

Recurrent Neural Networks

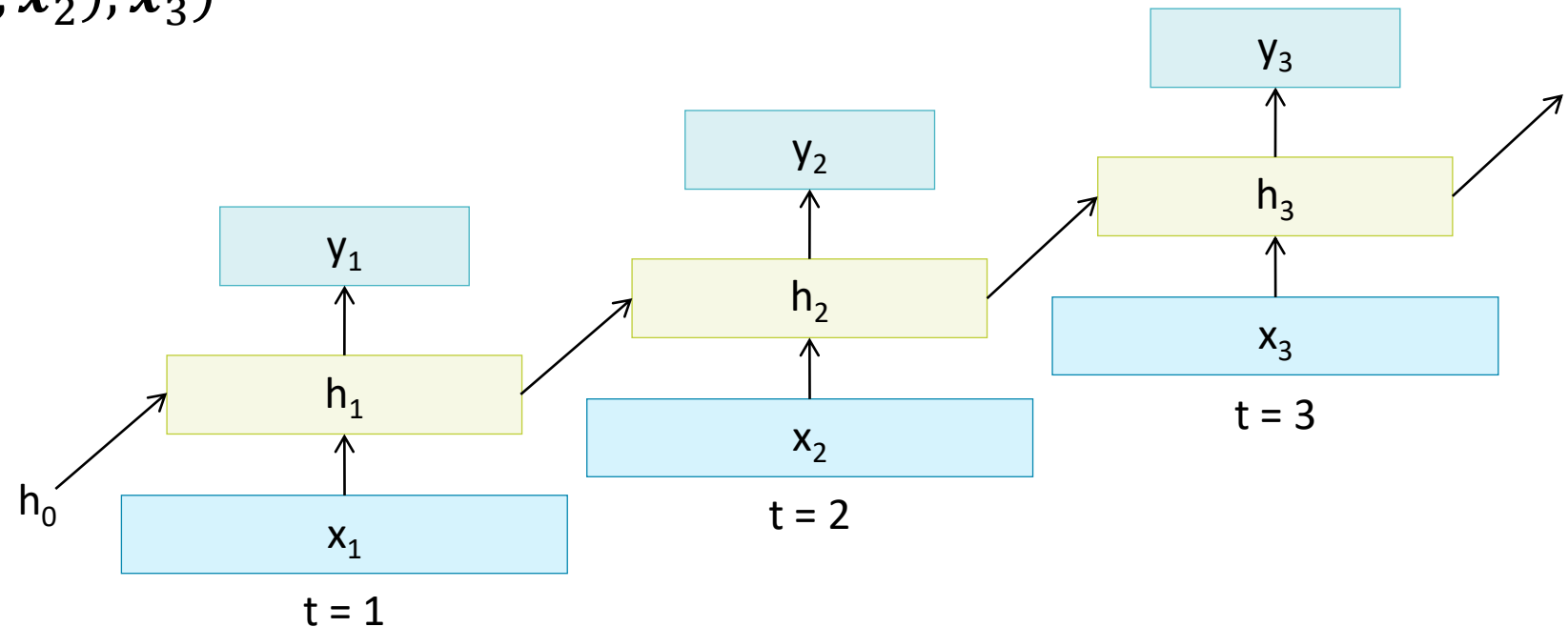
شبکه‌های عصبی بازگشتی

- شبکه‌های عصبی بازگشتی خروجی‌ها یا حالت‌های مخفی قبل را به عنوان ورودی دریافت می‌کنند
- ورودی ترکیبی شبکه در زمان t دارای اطلاعات زمانی از ورودی‌های گذشته ($T < t$) است
- RNNها برای تحلیل‌های زمانی بسیار مفید هستند زیرا مقادیر میانی (حالت) می‌توانند اطلاعات مربوط به ورودی‌های گذشته را ذخیره کنند



شبکه‌های عصبی بازگشتی

$$\begin{aligned} \mathbf{h}_3 &= f_W(\mathbf{h}_2, \mathbf{x}_3) \\ &= f_W(f_W(\mathbf{h}_1, \mathbf{x}_2), \mathbf{x}_3) \\ &= f_W(f_W(f_W(\mathbf{h}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3) \\ &= g^{(3)}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \end{aligned}$$



داده‌های متنی

- متن را می‌توان به عنوان دنباله‌ای از کاراکترها یا دنباله‌ای از کلمات در نظر گرفت
- یادگیری عمیق برای پردازش زبان طبیعی، شناسایی الگو است که بر روی کلمات، جملات و پاراگراف‌ها اعمال می‌شود، به همان روشی که یادگیری عمیق برای بینایی کامپیوتر، شناسایی الگوی اعمال شده روی پیکسل‌ها است
- از جمله کاربردهای آن دسته‌بندی اسناد، تحلیل احساسات نویسنده، بازشناسی نویسنده، ترجمه ماشینی، و پاسخ به پرسش (QA) است

پردازش متن

- مانند همه شبکه‌های عصبی دیگر، مدل‌های یادگیری عمیق متن خام را ورودی نمی‌گیرند: آنها فقط با تانسورهای عددی کار می‌کنند
- بردارسازی متن:
 - تقسیم متن به کلمه‌ها، و تبدیل هر کلمه به یک بردار
 - تقسیم متن به کاراکترها، و تبدیل هر کاراکتر به یک بردار
 - استخراج n-gram ها از کلمه‌ها یا کاراکترها، و تبدیل هر n-gram به یک بردار
 - n-gram ها گروه‌های دارای همپوشانی متشکل از چند کلمه یا کاراکتر متوالی هستند

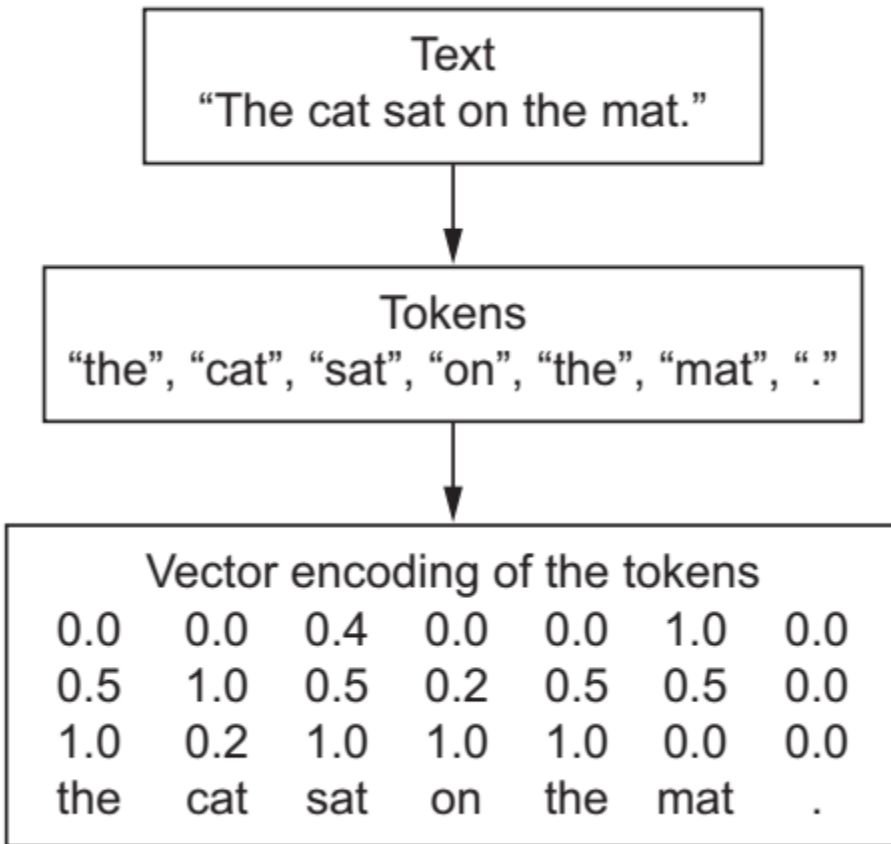
The cat sat on the mat

Bi-grams {"The cat", "cat sat", "sat on", "on the", "the mat"}

Tri-grams {"The cat sat", "cat sat on", "sat on the", "on the mat"}

Tokenization

- واحدهای مختلفی که می توان متن را به آنها تجزیه کرد (کلمات، کاراکترها یا n-gram Token نامیده می شوند و شکستن متن به آنها Tokenization نامیده می شود
- سپس برای هر Token باید یک بردار عددی در نظر گرفت



One-hot encoding

- ابتدایی ترین راه برای تبدیل توکن به بردار است
- یک عدد صحیح منحصر به فرد به هر کلمه اختصاص می یابد و سپس این عدد صحیح i به یک بردار باینری با اندازه N (اندازه واژگان) تبدیل می شود
- همه مقادیر این بردار صفر است به جز ورودی i ام که ۱ است
- این کار در سطح کاراکتر و n-gram هم قابل انجام است



One-hot encoding

Listing 6.3 Using Keras for word-level one-hot encoding

```
from keras.preprocessing.text import Tokenizer
```

```
samples = ['The cat sat on the mat.', 'The dog ate my homework.']
```

```
tokenizer = Tokenizer(num_words=1000)
```

```
tokenizer.fit_on_texts(samples)
```

```
sequences = tokenizer.texts_to_sequences(samples)
```

```
one_hot_results = tokenizer.texts_to_matrix(samples, mode='binary')
```

```
word_index = tokenizer.word_index
```

```
print('Found %s unique tokens.' % len(word_index))
```

Creates a tokenizer, configured to only take into account the 1,000 most common words

Turns strings into lists of integer indices

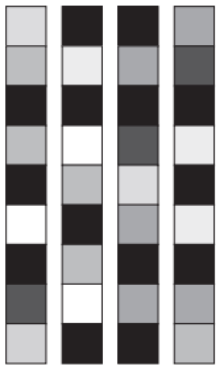
How you can recover the word index that was computed

Builds the word index

You could also directly get the one-hot binary representations. Vectorization modes other than one-hot encoding are supported by this tokenizer.

جانمایی کلمات (Word embedding)

- جانمایی کلمات اطلاعات بیشتر را در ابعاد بسیار کمتری قرار می‌دهد
- این بردارها را می‌توان با استفاده از حجم زیادی از متن پیش‌آموزش داد و در مجموعه داده‌های کوچک از آنها استفاده کرد



One-hot word vectors:

- Sparse
- High-dimensional
- Hardcoded

Word embeddings:

- Dense
- Lower-dimensional
- Learned from data

جانمایی کلمات (Word embedding)

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
↑ Gender	-1	1	-0.95	0.97	0.00	0.01
300 Royal	0.01	0.02	<u>0.93</u>	<u>0.95</u>	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
⋮ size cost alive verb	⋮	⋮				

e₅₃₉₁
e₉₈₅₃

Andrew Ng

جانمایی کلمات (Word embedding)

- دو روش برای دستیابی به جانمایی کلمات وجود دارد:

- آموزش همزمان با مسئله اصلی

- مقداردهی اولیه به صورت تصادفی

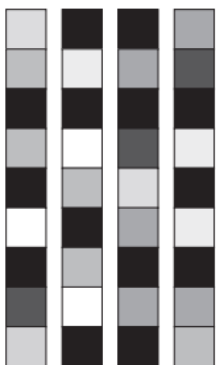
- بارگذاری مقادیری که از قبل بر اساس آموزش یک مسئله دیگر بدست آمده‌اند

- جانمایی کلمات پیش‌آمخته



One-hot word vectors:

- Sparse
- High-dimensional
- Hardcoded



Word embeddings:

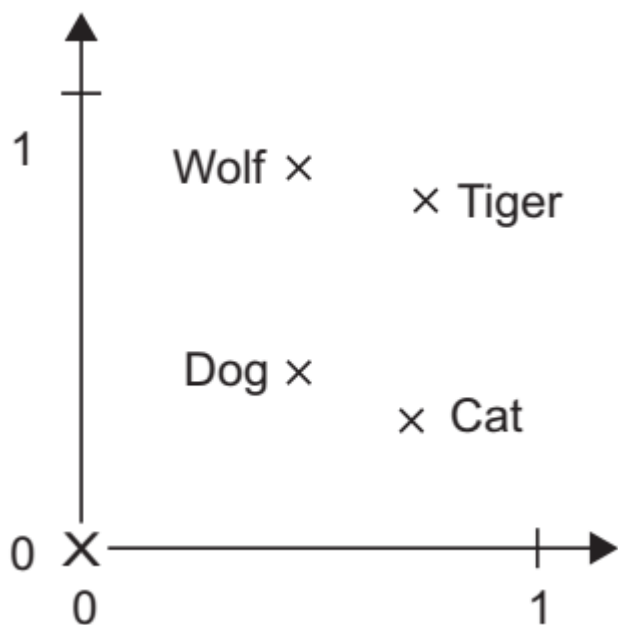
- Dense
- Lower-dimensional
- Learned from data

آموزش جانمایی کلمات

- اختصاص یک بردار تصادفی به هر کلمه
 - مشکل این رویکرد این است که این فضا ساختاری ندارد
 - به عنوان مثال، کلمات **accurate** و **exact** ممکن است با جانمایی‌های کاملاً متفاوتی همراه شوند، با این وجود که در اکثر جملات قابل تعویض هستند
- روابط هندسی بین بردارهای کلمات باید منعکس‌کننده روابط معنایی بین این کلمات باشد
 - جانمایی کلمات به معنای نگاشت زبان انسان به یک فضای هندسی است
 - در یک فضای جانمایی معقول، انتظار می‌رود کلمات مترادف دارای مقادیر مشابهی باشند

آموزش جانمایی کلمات

- انتظار می‌رود فاصله هندسی بین هر دو بردار کلمه با فاصله معنایی بین کلمات مرتبط باشد
- همچنین، انتظار می‌رود جهت‌های مختلف در فضای آموخته شده معنادار باشند



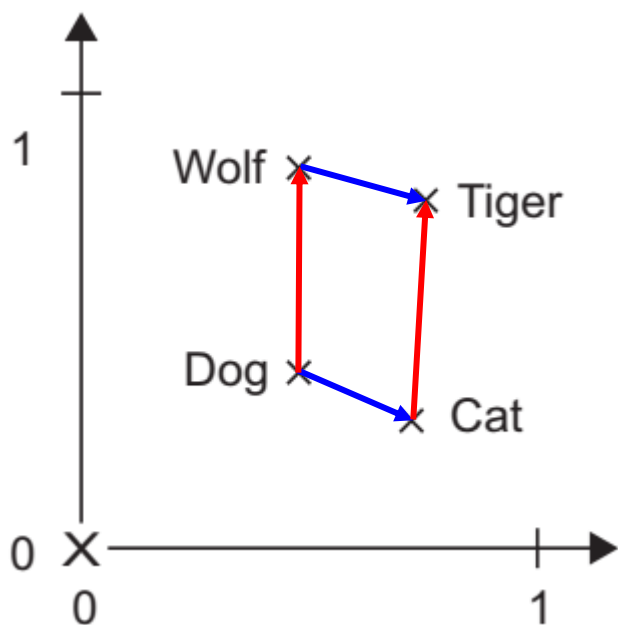
آموزش جانمایی کلمات

- انتظار می‌رود فاصله هندسی بین هر دو بردار کلمه با فاصله معنایی بین کلمات مرتبط باشد
- همچنین، انتظار می‌رود جهت‌های مختلف در فضای آموخته شده معنادار باشند
- در این مثال، جهت بردار از گربه به ببر و از سگ به گرگ مشابه است

- از حیوان خانگی به حیوان وحشی

- جهت بردار از سگ به گربه و از گرگ به ببر نیز مشابه است

- از خانواده سگ به خانواده گربه



آموزش جانمایی کلمات

- یک فضای مناسب تا حد زیادی به مسئله مورد نظر بستگی دارد
- فضای مناسب برای دسته‌بندی نقد فیلم ممکن است متفاوت از فضای مناسب برای دسته‌بندی اسناد حقوقی باشد، زیرا اهمیت برخی روابط معنایی متفاوت است
- آموزش یک فضای جانمایی جدید برای هر مسئله جدید منطقی است

لایه Embedding

- لغت‌نامه‌ای که اندیس صحیح مربوط به هر کلمه را به یک بردار معنایی نگاشت می‌کند
- این لایه، یک تانسور دوبعدی از اعداد صحیح را به عنوان ورودی می‌گیرد
- (samples, sequence_length)
- خروجی این لایه یک تانسور سه‌بعدی از اعداد اعشاری است
- (samples, sequence_length, embedding_dimensionality)

Listing 6.5 Instantiating an Embedding layer

```
from keras.layers import Embedding  
embedding_layer = Embedding(1000, 64)
```



The Embedding layer takes at least two arguments: the number of possible tokens (here, 1,000: 1 + maximum word index) and the dimensionality of the embeddings (here, 64).

لایه Embedding

- هنگامی که یک لایه Embedding را می‌سازیم، وزن‌های آن در ابتدا تصادفی است
- در طول آموزش، این بردارهای کلمات به تدریج از طریق پس‌انتشار تنظیم می‌شوند

Listing 6.5 Instantiating an Embedding layer

```
from keras.layers import Embedding  
embedding_layer = Embedding(1000, 64)
```



The Embedding layer takes at least two arguments: the number of possible tokens (here, 1,000: 1 + maximum word index) and the dimensionality of the embeddings (here, 64).

جانمایی کلمات پیش‌آموخته

- مشابه با مفهوم شبکه‌های کانولوشنی پیش‌آموخته
 - زمانیکه داده‌های کافی برای یادگیری ویژگی‌های قدرتمند نداریم، اما انتظار داریم ویژگی‌هایی که به آن نیاز داریم نسبتاً عمومی باشند
- بجای یادگیری جانمایی کلمات به طور مشترک با مسئله مورد نظر، می‌توان بردارهای جانمایی آموخته شده برای حل یک مسئله دیگر را بارگذاری کنیم
- معمولاً با استفاده از اطلاعات آماری وقوع کلمات محاسبه می‌شود
 - با یا بدون استفاده از شبکه‌های عصبی
- دو مورد از معروف‌ترین و موفق‌ترین آنها [Word2vec](#) و [GloVe](#) هستند