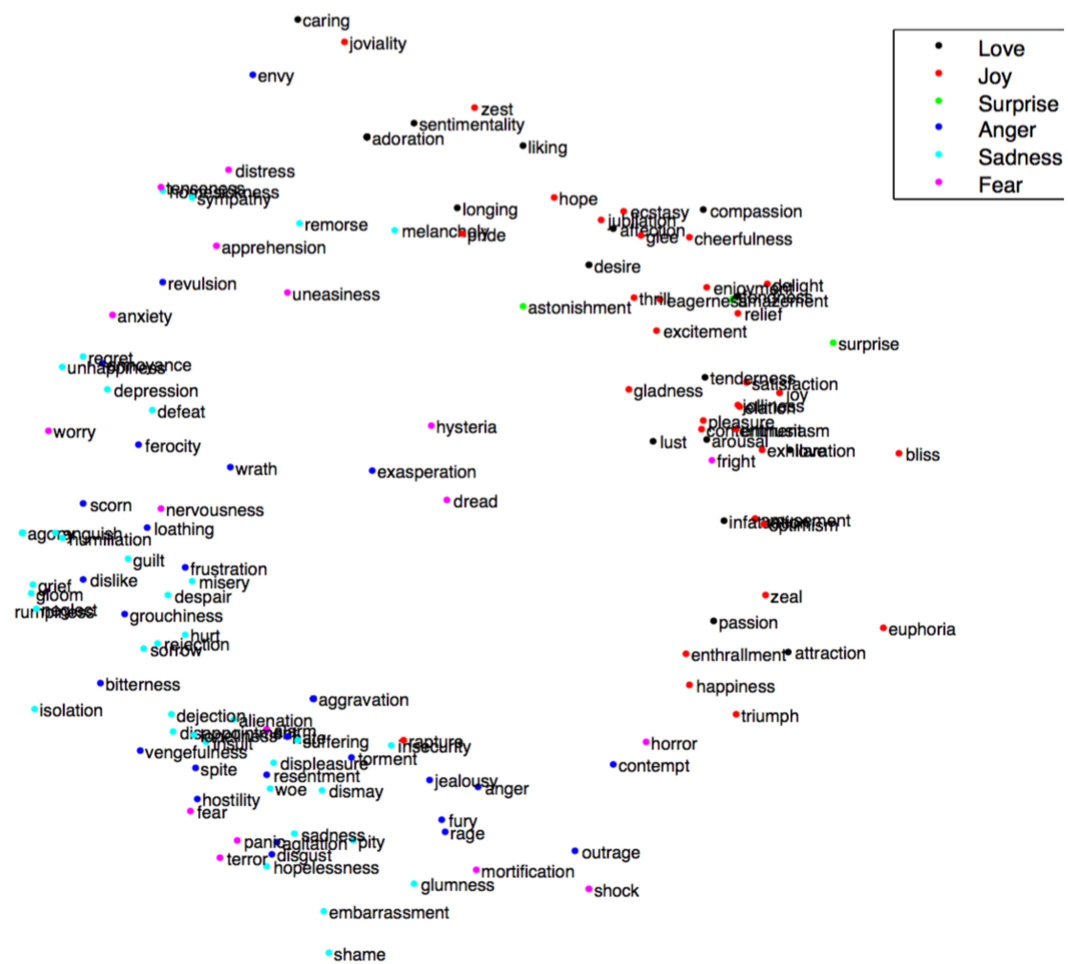
Feature extraction given Text data

Word embeddings, word2vec

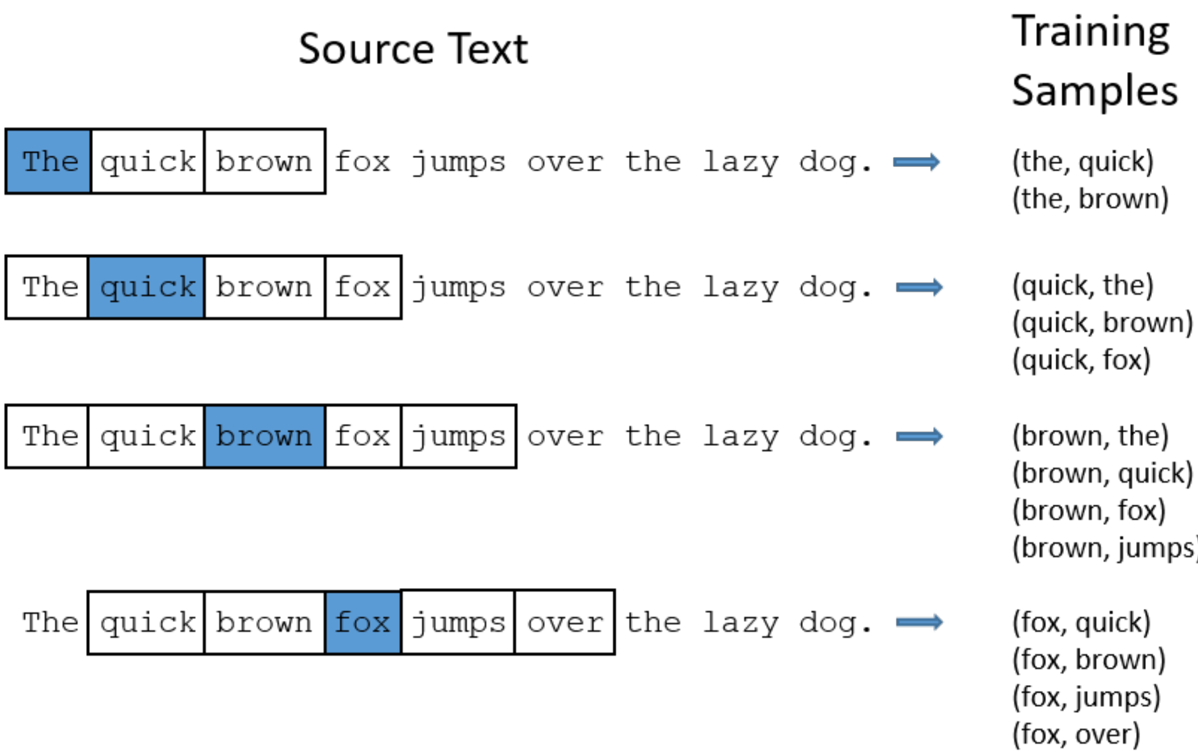
Can we **embed words** into a latent space?

This embedding came from directly querying for relationships.

word2vec is a popular unsupervised learning approach that just uses a text corpus (e.g. nytimes.com)

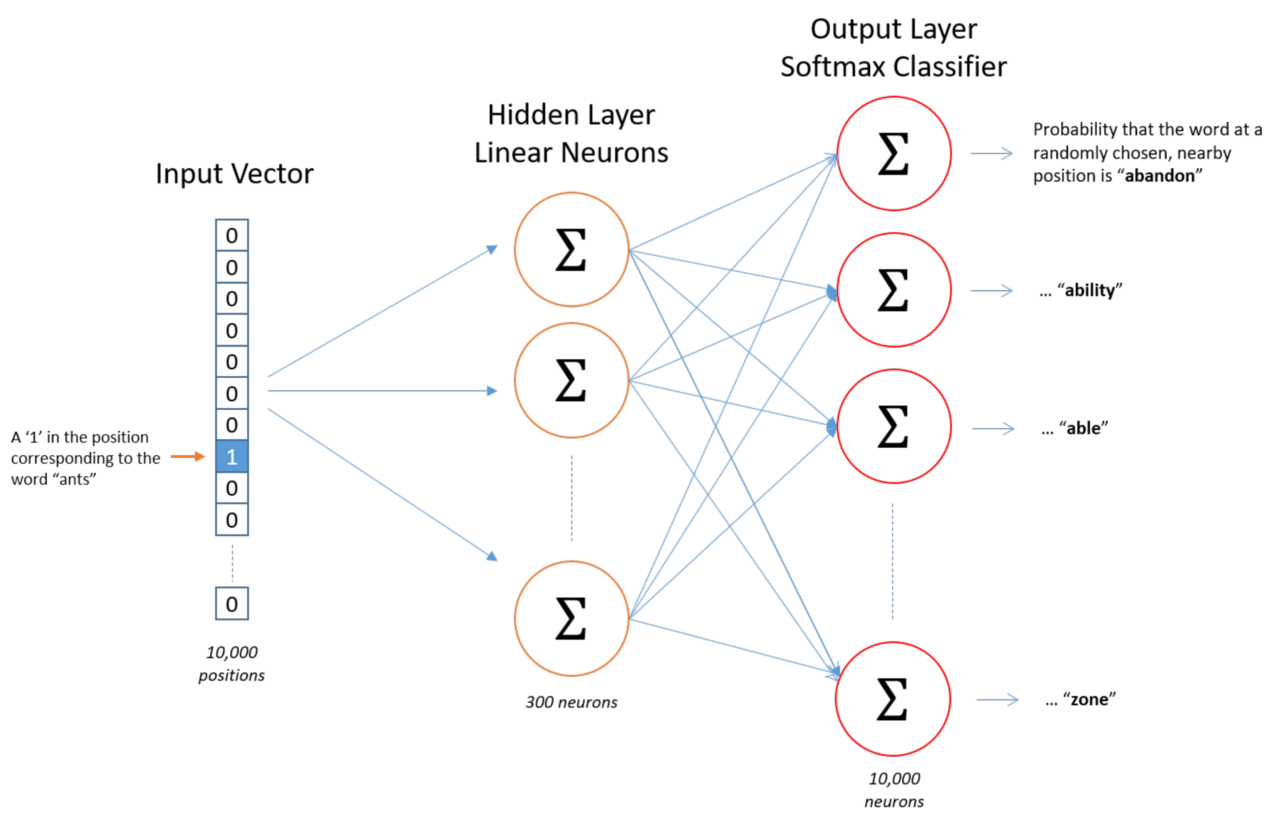


Word embeddings, word2vec



slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

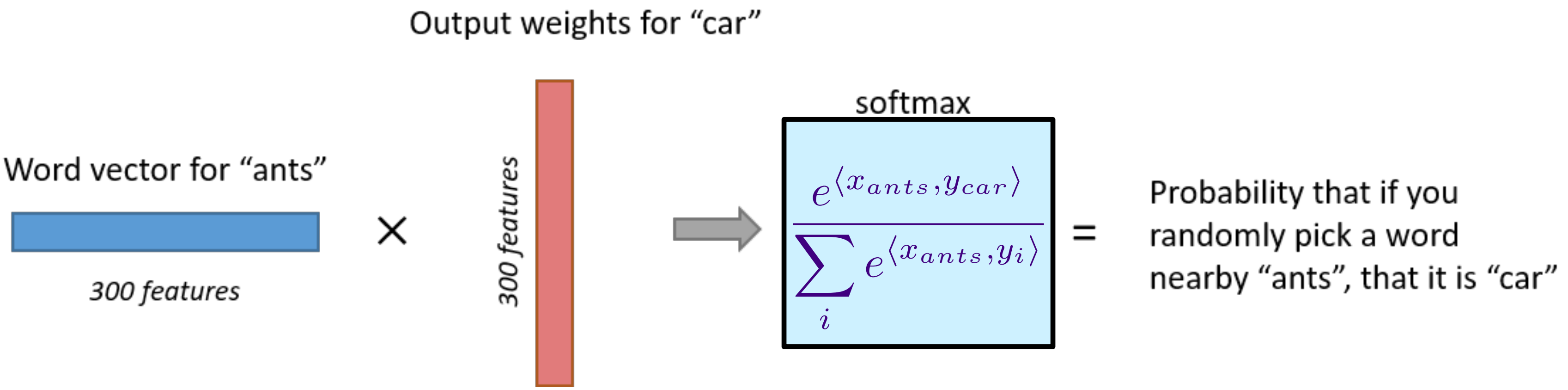
Word embeddings, word2vec



Training neural network to predict co-occuring words. Use first layer weights as embedding, throw out output layer

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

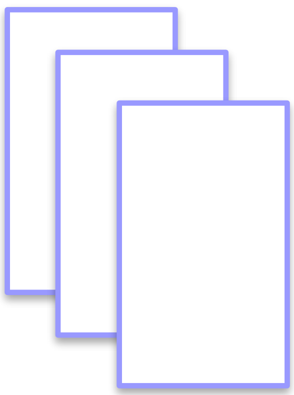
Word embeddings, word2vec



Training neural network to predict co-occuring words. Use first layer weights as embedding, throw out output layer

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Bag of Words



n documents/articles with lots of text

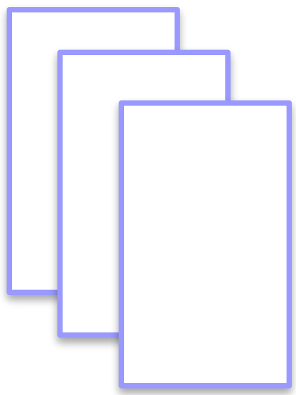
Questions:
- How to get a feature representation of each article?
- How to cluster documents into topics?

Bag of words model:

i*th* document: $x_i \in \mathbb{R}^D$

$$x_{i,j} = \text{ proportion of times } j\text{th word occurred in } i\text{th document}$$

Bag of Words



n documents/articles with lots of text

- **Can we embed each document into a feature space?**

Bag of words model:

i*th* document: $x_i \in \mathbb{R}^D$

$$x_{i,j} = \text{ proportion of times } j\text{th word occurred in } i\text{th document}$$

Given vectors, run k-means or Gaussian mixture model to find k clusters/topics

Nonnegative matrix factorization (NMF)

$A \in \mathbb{R}^{m \times n}$ $\quad A_{i,j} = \text{frequency of } j\text{th word in document } i$

Nonnegative Matrix factorization:

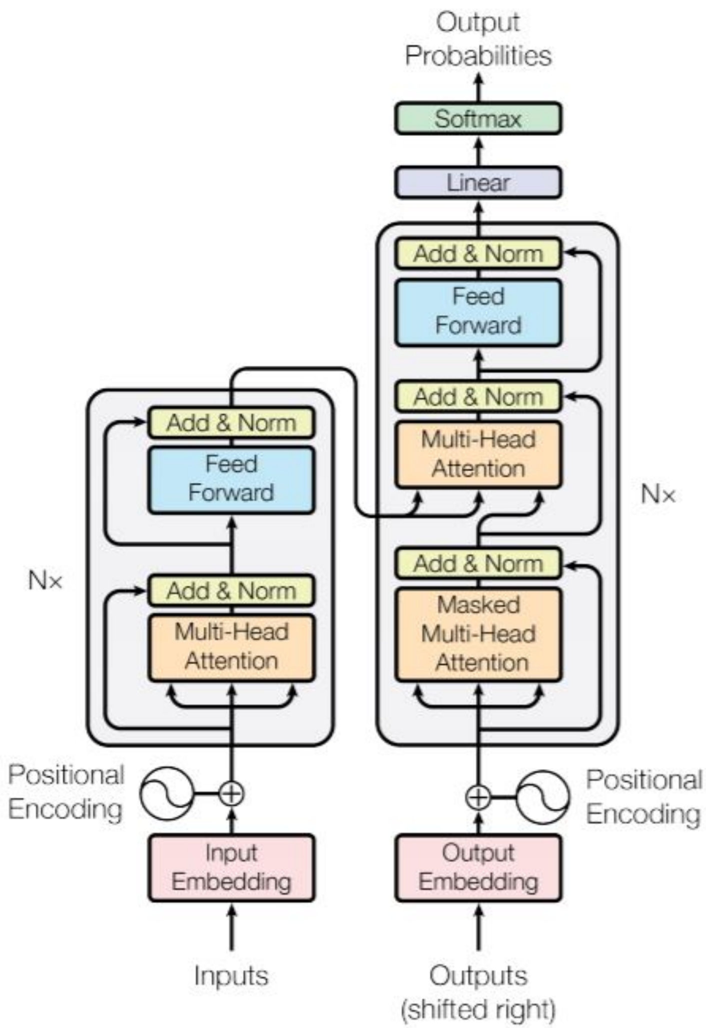
$$\min_{W \in \mathbb{R}_+^{m \times d},\, H \in \mathbb{R}_+^{n \times d}} \|A - WH^T\|_F^2$$

d is number of topics

Each column of H represents a cluster of a topic,
Each row W is some weights a combination of topics

Also see latent Dirichlet factorization (LDA)

BERT


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

Feature extraction given sequential data

Time-dependent data

$x_t \in \mathbb{R}$: AAPL stock price at time t

Prediction model: $p(x_{t+1} | x_t, x_{t-1}, x_{t-2}, \dots)$

Time-dependent data

$x_t \in \mathbb{R}$: AAPL stock price at time t

$h_t \in \mathbb{R}^d$: hidden latent state of AAPL

$$\text{Prediction model: } p(x_{t+1} | x_t, x_{t-1}, x_{t-2}, \dots)$$
$$\approx p(x_{t+1} | x_t, h_{t+1})$$

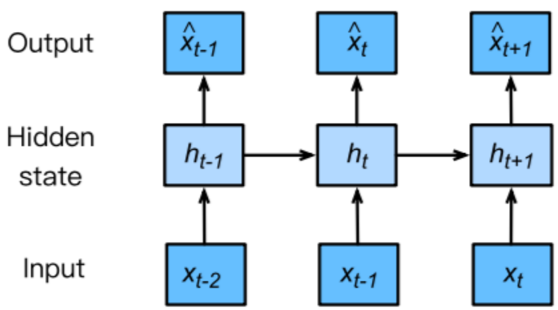
Time-dependent data

$x_t \in \mathbb{R}$: AAPL stock price at time t

$h_t \in \mathbb{R}^d$: hidden latent state of AAPL

$$\text{Prediction model: } p(x_{t+1} | x_t, x_{t-1}, x_{t-2}, \dots)$$

$$\approx p(x_{t+1} | x_t, h_{t+1})$$



$$h_{t+1} = g(h_t, x_t)$$

Hidden state and g never observed, but learned!

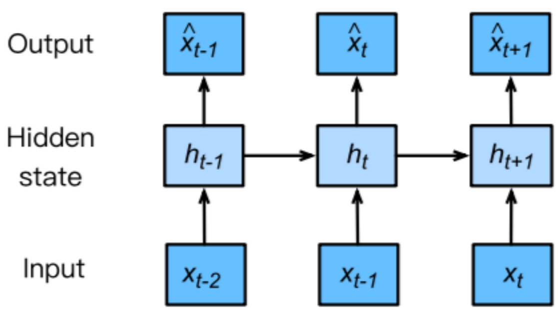
Zhang et al. "Dive into Deep Learning"

Time-dependent data

$x_t \in \mathbb{R}$: AAPL stock price at time t

$h_t \in \mathbb{R}^d$: hidden latent state of AAPL

$$\text{Prediction model: } p(x_{t+1} | x_t, x_{t-1}, x_{t-2}, \dots) \approx p(x_{t+1} | x_t, h_{t+1})$$



$$h_{t+1} = g(h_t, x_t)$$

Hidden state and g never observed, but learned!

Explicit:

$$h_{t+1} = \sigma(Ah_t + Bx_t)$$
$$\widehat{x}_{t+1} = Ch_{t+1} + Dx_t$$

$$\sum_t (x_t - \widehat{x}_t)^2$$

Zhang et al. “Dive into Deep Learning”

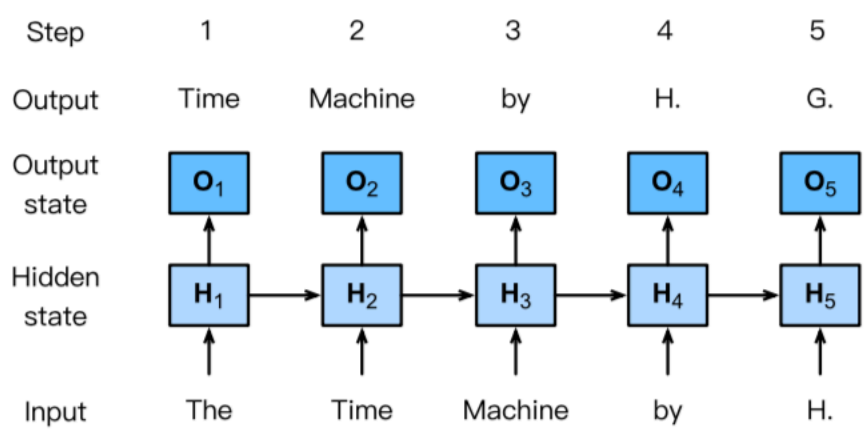
Time-dependent data

$$\text{Prediction model: } p(x_{t+1}|x_t, x_{t-1}, x_{t-2}, \dots) \approx p(x_{t+1}|x_t, h_{t+1})$$

$$h_{t+1} = g(h_t, x_t)$$

Hidden state and g never observed, but learned!

Model also works with text!



Zhang et al. "Dive into Deep Learning"

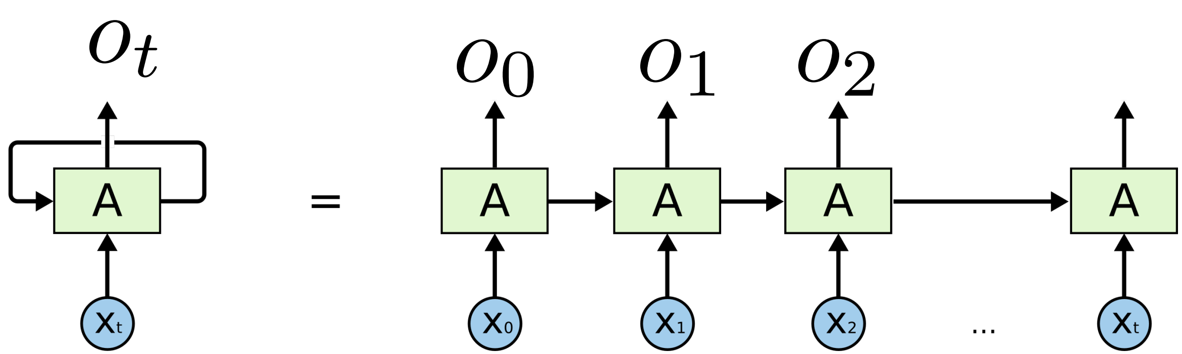
Time-dependent data

$$\text{Prediction model: } p(x_{t+1}|x_t, x_{t-1}, x_{t-2}, \dots)$$
$$\approx p(x_{t+1}|x_t, h_{t+1})$$

$$h_{t+1} = g(h_t, x_t)$$

Hidden state and g never observed, but learned!

Recurrent Neural Network



Slide: http://colah.github.io/posts/2015-08-Understanding-LSTMs/