

# From Shallow to Deep Language Representations

1 Basics · 2 Shallow Models · 3 Transformers · 4 BERT

**KDD'19 Anchorage**

**Aston Zhang, Haibin Lin, Leonard Lausen, Sheng Zha, Alex Smola**

[www.d2l.ai](http://www.d2l.ai) [gluon-nlp.mxnet.io](http://gluon-nlp.mxnet.io)

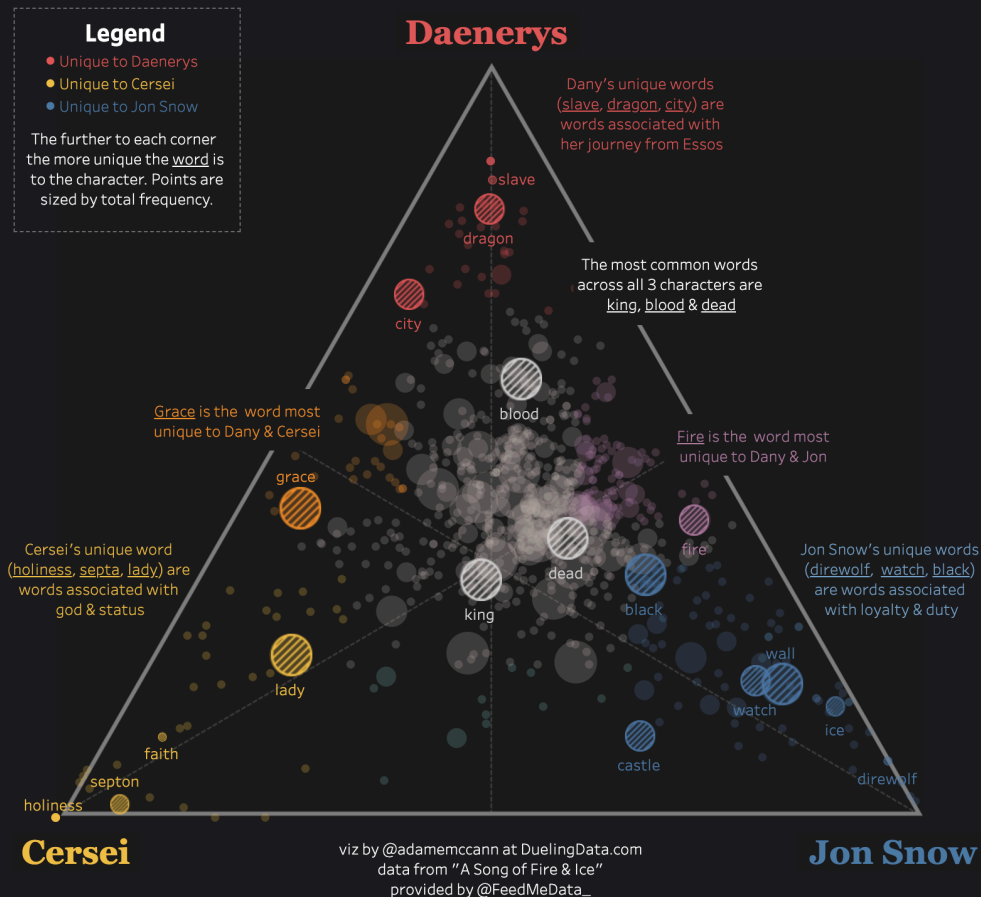
# Outline (Shallow Models)

- **Word Embedding**
  - word2vec
  - fastText
  - GloVe
- **Applications (hands-on)**
  - **Similarity and Analogy**
  - **Sentiment Analysis with RNN**
  - **Sentiment Analysis with CNN**

# GAME OF THRONES™ IN WORDS

This viz shows the most unique words by character for each chapter in the 5 Game of Thrones books

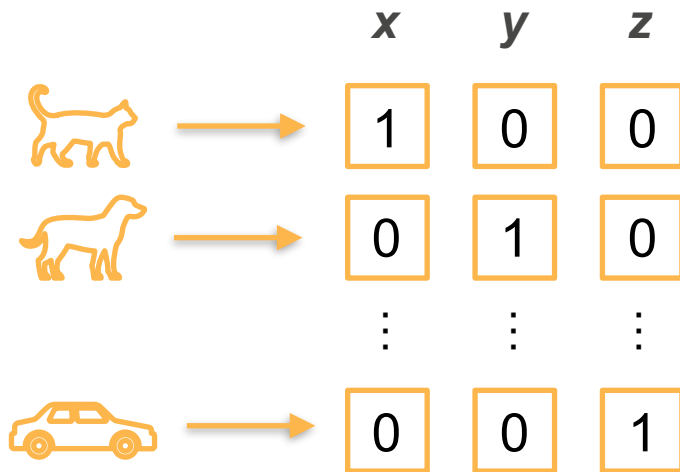
## Word2Vec



# One-Hot Encoding




- One-hot vectors map objects (words) to fixed-length vectors
- Vectors contain only ID, **no semantic meaning**

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{z}, \mathbf{y} \rangle = 0$$



# Word2vec

- Embedding vector for semantic information
- Use inner product  $\langle \mathbf{x}, \mathbf{y} \rangle$  to measure similarity
- $\langle \mathbf{x}, \mathbf{y} \rangle > \langle \mathbf{x}, \mathbf{z} \rangle$   
implies that  $\mathbf{x}$  is more similar to  $\mathbf{y}$
- Build auxiliary probabilistic model
- Maximize the likelihood function to learn embedding

	$\mathbf{x}$	$\mathbf{y}$	$\mathbf{z}$
	1	0.7	0
	0.3	1	0
	$\vdots$	$\vdots$	$\vdots$
	0	0	1

# Word Context

- Context of a word helps define sense

Malört tastes bitter.

I get drunk from Malört.

Gin and Malört go well together.

I installed the Malört yesterday.

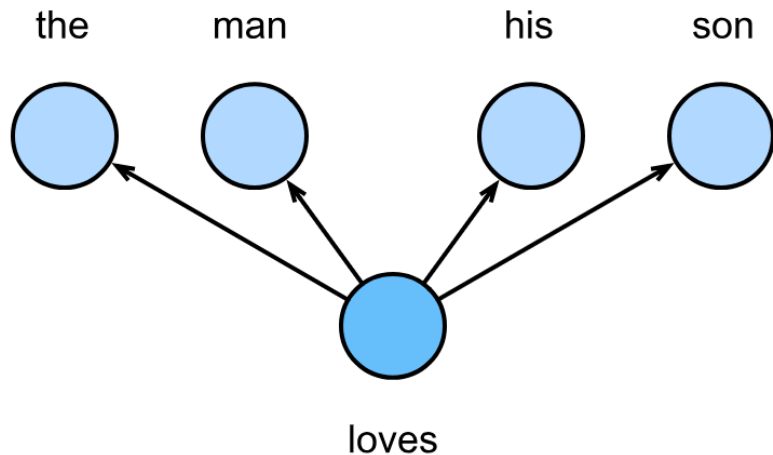
A broken Malört caused the car crash.

Always buy a new Malört.

Fun fact - Malört is one of the 10 worst drinks in the world.

# Skip-Gram Model

- Heuristic
  - Model context words given central word
  - Model each context word **independently**



$$P(\text{the, man, his, son} \mid \text{loves}) =$$

$$\begin{aligned} &P(\text{the} \mid \text{loves}) \\ &\cdot P(\text{man} \mid \text{loves}) \\ &\cdot P(\text{his} \mid \text{loves}) \\ &\cdot P(\text{son} \mid \text{loves}) \end{aligned}$$

# Likelihood Function

Summing over all words  
is too expensive

$$P(w_o \mid w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in V} \exp(\mathbf{u}_i^\top \mathbf{v}_c)}$$

$V$  : all context words

	Word	Embedding
Center	$w_c$	$\mathbf{v}_c \in \mathbb{R}^d$
Context	$w_o$	$\mathbf{u}_o \in \mathbb{R}^d$

- Likelihood for sequence

$$\prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} P(w^{(t+j)} \mid w^{(t)})$$



# One more hack ... Model Cooccurrence

- Treat cooccurrence of center word and context word in the same window as an event

$$P(D = 1 | w_c, w_o) = \sigma(\mathbf{u}_c^T \mathbf{v}_o) \quad \sigma(x) = \frac{1}{1 + \exp(-x)}$$

- Change likelihood function to

$$\prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} \mathbb{P}(D = 1 | w^{(t)}, w^{(t+j)})$$

Naive solution: infinity

# Negative Sampling

- Sample noise word  $w_n$  that doesn't appear in the window

$$P(D = 0 | w_c, w_n) = 1 - \sigma(\mathbf{u}_n^\top \mathbf{v}_c) = \frac{1}{1 + \exp(\mathbf{u}_n^\top \mathbf{v}_c)}$$

- Add into the likelihood function as well

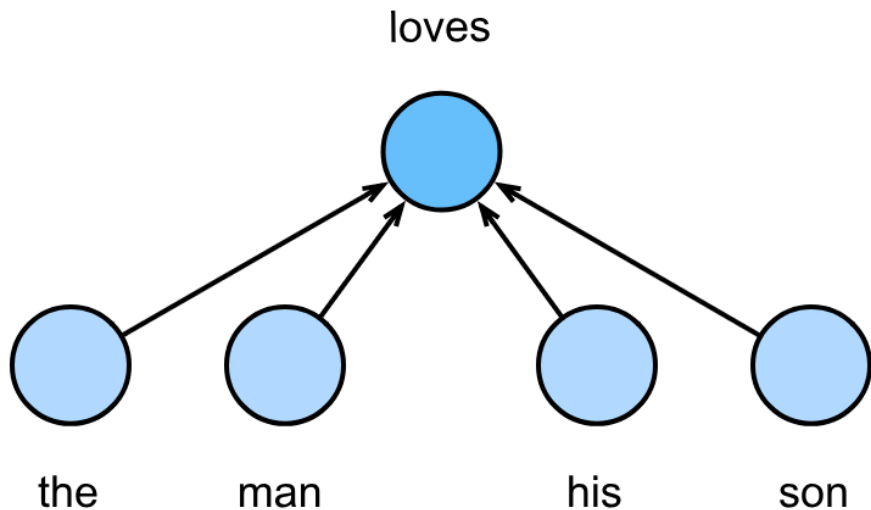
$$\prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} \mathbb{P}(D = 1 | w^{(t)}, w^{(t+j)}) \prod_{k=1, w_k \sim \mathbb{P}(w)}^K P(D = 0 | w^{(t)}, w_k)$$

# Continuous Bag Of Words (CBOW)

- **CBOW**

Center word is based on the context

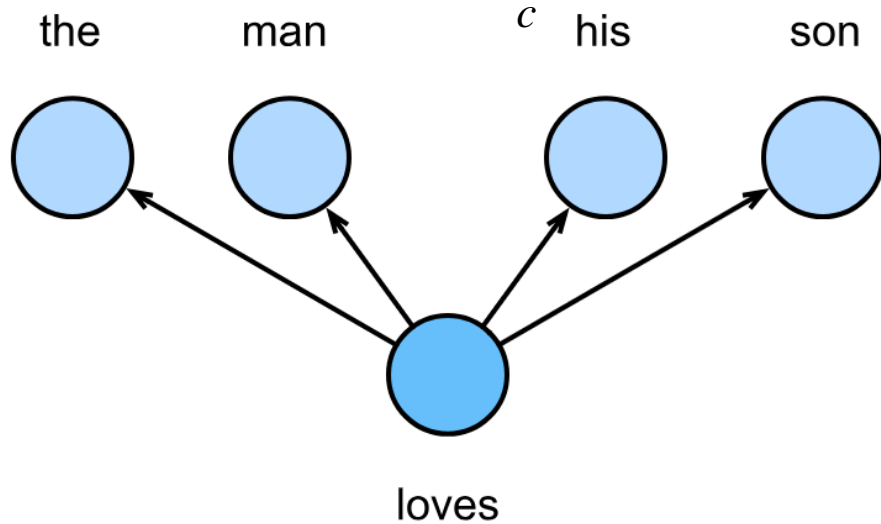
$$p(w | \text{context}) = p(w | \{w_c\})$$



- **Skip-gram**

Context is based on Center word

$$p(\text{context} | w) = \prod p(w_c | w)$$



# Likelihood Function

- Ignore order, just average

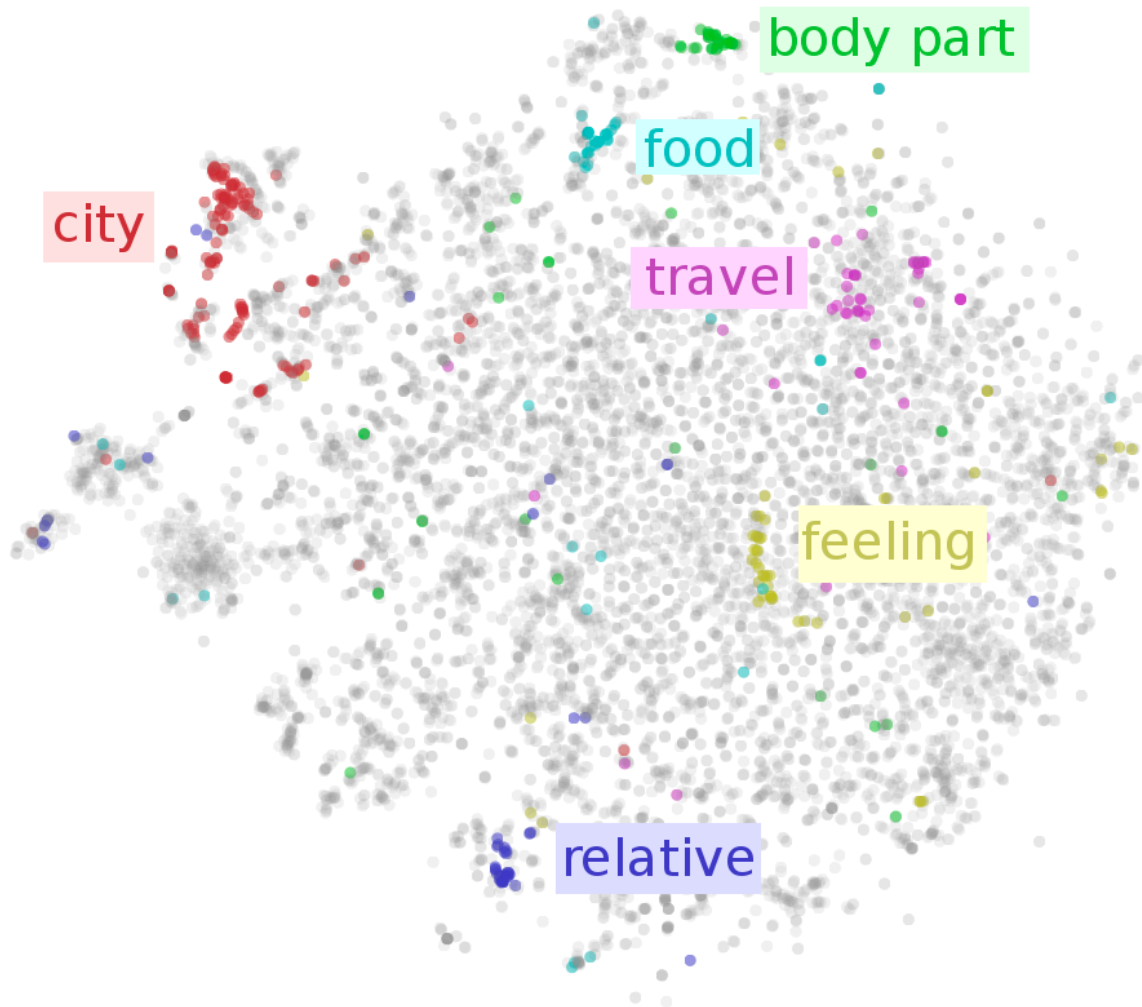
$$\phi(\text{context}) = \frac{1}{2m} \sum_{i \in [-m, m] \setminus \{0\}} \mathbf{v}_i$$

$$P(w \mid \text{context}) = \frac{\exp \mathbf{u}_w^\top \phi(\text{context})}{\sum_{w' \in V} \exp \mathbf{u}_{w'}^\top \phi(\text{context})}$$

- Likelihood

$$\prod_{t=1}^T P(w_t \mid \text{context}_t)$$

# More Embedding Models



# FastText

- Word2Vec learns each embedding independently. Big problem for **rare** words.
- Words have lots of structure. Use it!  
**dog, catch** → **dogcatcher**  
pneumonoultramicroscopicsilicovolcanoconiosis
- Decompose into n-grams  
<where> → 5-grams <wher where here>  
4-grams <whe wher here ere>  
3-grams <wh whe her ere re>  
2-grams <w wh he er re e>



# FastText

- Use subwords as features (FastText uses lengths 3-6)
- $G_w$  is union of subwords. This yields

$$\mathbf{u}_w = \sum_{g \in G_w} \mathbf{u}_g + \bar{\mathbf{u}}_w$$

Sometimes use dedicated embedding  $\bar{\mathbf{u}}_w$  for word, too.

- Rest of model is the same as skip-gram

# Word Embedding with Global Vectors (GloVe)

- Goal - get rid of negative sampling
- **Step 1 - Cross Entropy reformulation**

$$\begin{aligned}\sum_{t=1}^T \sum_{j \in [-m, m] \setminus \{0\}} -\log q(w_{t+j} | w_t) &= \sum_{w, w' \in V} -n(w', w) \log q(w' | w) \\ &= \sum_{w \in V} n(w) \sum_{w' \in V} \underbrace{\frac{n(w', w)}{n(w)}}_{=: p(w' | w)} \log q(w' | w)\end{aligned}$$



# Word Embedding with Global Vectors (GloVe)

- **Step 2 - Least mean squares approximation**

$$\sum_{w' \in V} -p(w' | w) \log q(w' | w) \longrightarrow \sum_{w' \in V} (\log p(w' | w) - \log q(w' | w))^2$$

- **Step 3 - Remove log-partition function (and add bias)**

$$q(w' | w) \longrightarrow \exp \mathbf{u}_{w'}^\top \mathbf{v}_w \quad p(w' | w) \longrightarrow n(w', w) \quad \begin{matrix} \downarrow \\ b_w + c_{w'} \end{matrix}$$

- **Step 4 - Weighting for  $n(w', w)$  (downweigh large terms)**

$$\sum_{w, w' \in W} h(n(w', w)) (\mathbf{u}_{w'}^\top \mathbf{v}_w + b_w + c_{w'} - \log n(w', w))^2$$

# Code...