From Shallow to Deep Language Representations

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

KDD'19 Anchorage

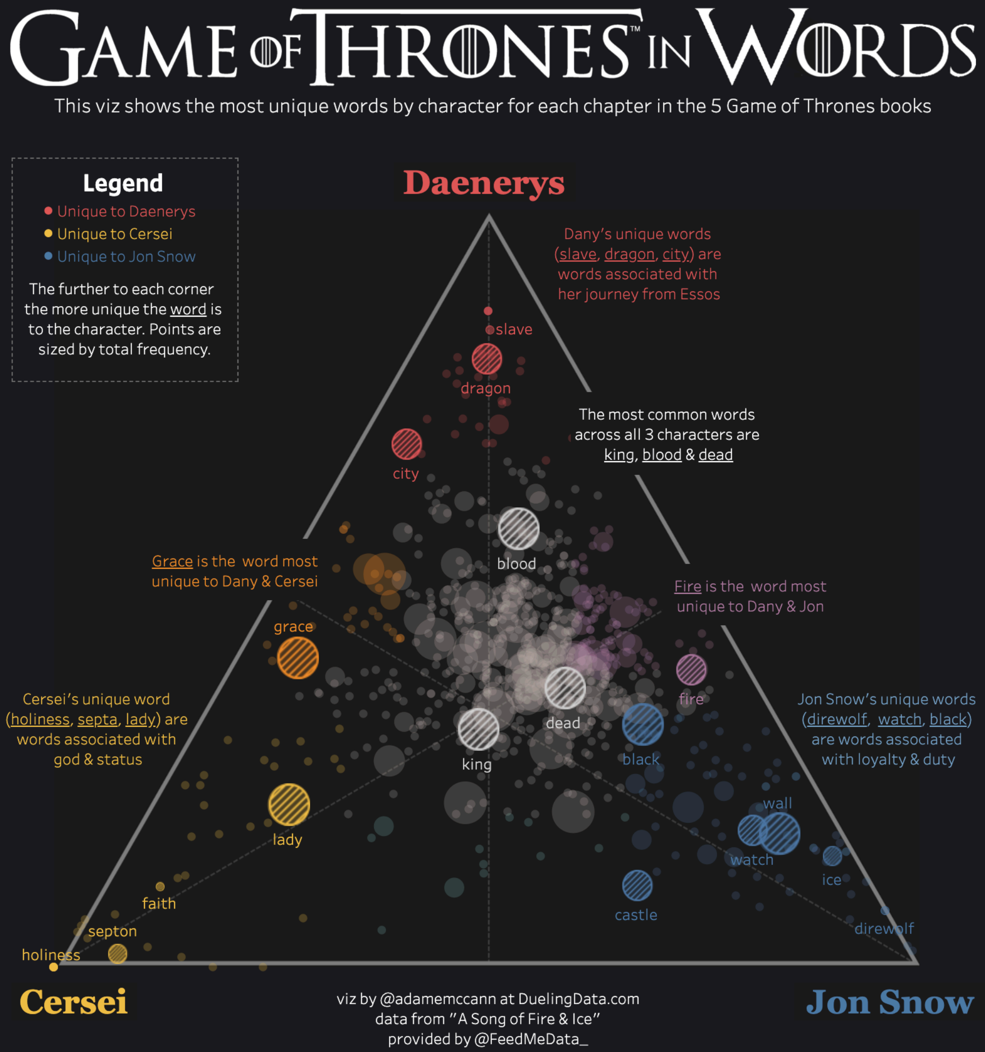
Aston Zhang, Haibin Lin, Leonard Lausen, Sheng Zha, Alex Smola

d2l.ai 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**

Word2Vec

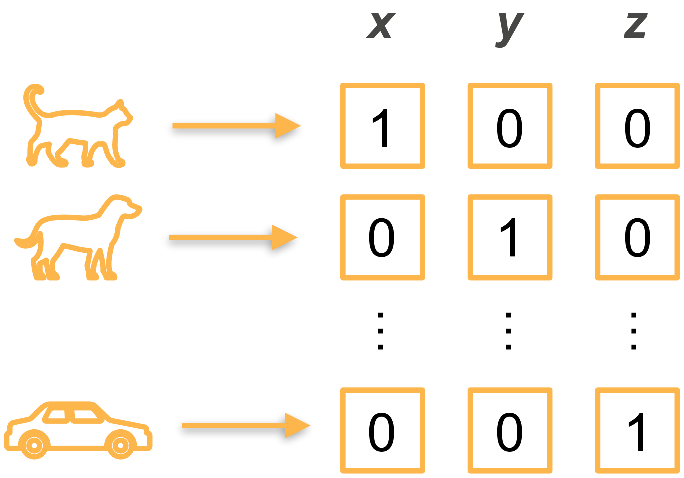


GAME OF THRONES IN WORDS
This viz shows the most unique words by character for each chapter in the 5 Game of Thrones books
Legend
Unique to Daenerys
Unique to Cersei
Unique to Jon Snow
The further to each corner the more unique the word is to the character. Points are sized by total frequency.
Daenerys
Dany's unique words (slave, dragon, city) are words associated with her journey from Essos
slave
dragon
city
The most common words across all 3 characters are king, blood & dead
blood
Grace is the word most unique to Dany & Cersei
Fire is the word most unique to Dany & Jon
grace
fire
dead
black
king
Cersei's unique word (holiness, septa, lady) are words associated with god & status
Jon Snow's unique words (direwolf, watch, black) are words associated with loyalty & duty
lady
wall
watch
ice
castle
faith
septon
holiness
direwolf
Cersei
Jon Snow
viz by @adamemccann at DuelingData.com
data from "A Song of Fire & Ice"
provided by @FeedMeData_

One-Hot Encoding

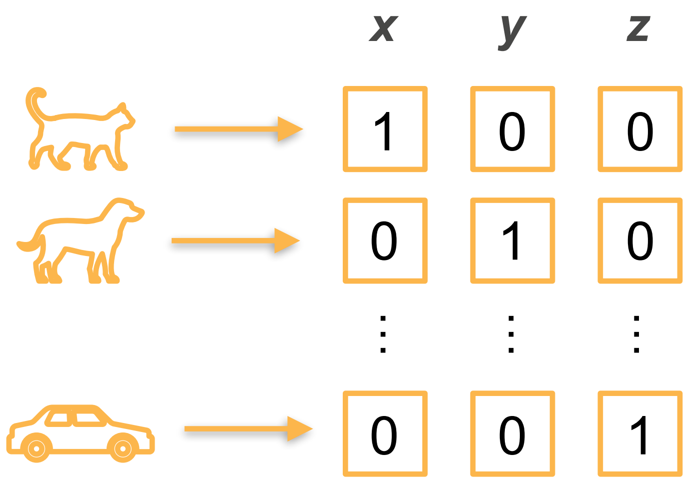
- One-hot vectors map objects (words) to fixed-length vectors
- Vectors contain only ID, **no semantic meaning**

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



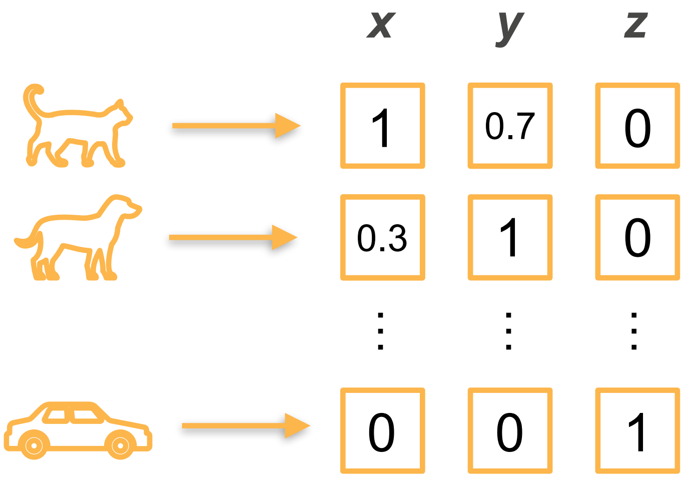
Word2vec

- Embedding vector for semantic information
- Use inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ to 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



Word2vec

- Embedding vector for semantic information
- Use inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ to 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



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.
```

Word Context

- Context of a word helps define sense

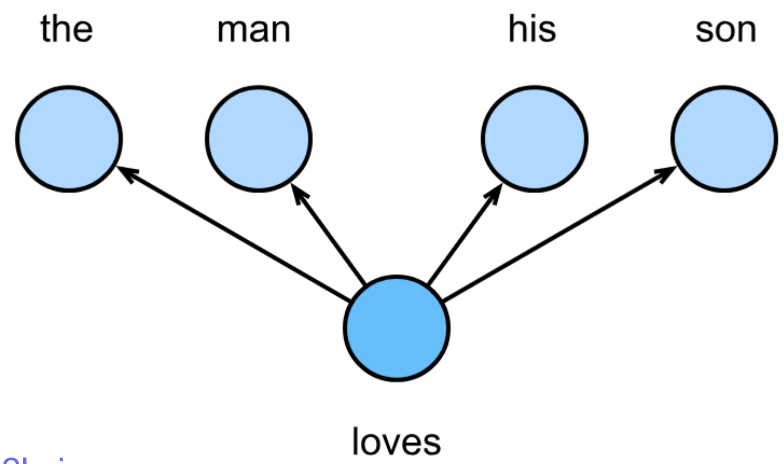
```
Malört tastes bitter.
I get drunk from Malört.
Gin and Malört go well together.

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**



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

Likelihood Function

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

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

V : all context words

- Likelihood for sequence

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

Likelihood Function

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

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

- 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

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

- Change likelihood function to

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

One more hack … Model Cooccurrence

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

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

- Change likelihood function to

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

Naive solution: infinity

Negative Sampling

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

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

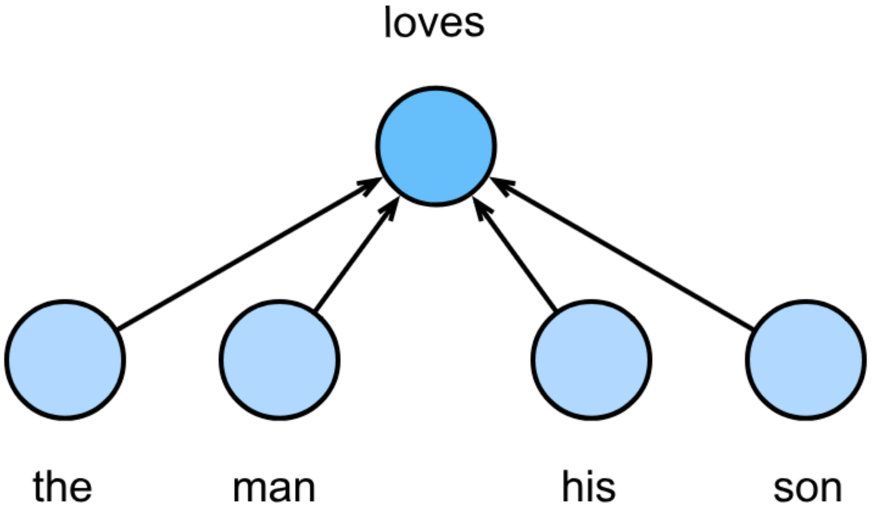
- Add into the likelihood function as well

$$\prod_{t=1}^{T} \prod_{-m \le j \le m,\ j \ne 0} \mathbb{P}(D=1 \mid w^{(t)}, w^{(t+j)}) \boxed{\prod_{k=1,\ w_k \sim \mathbb{P}(w)}^{K} P(D=0 \mid w^{(t)}, w_k)}$$

Continuous Bag Of Words (CBOW)

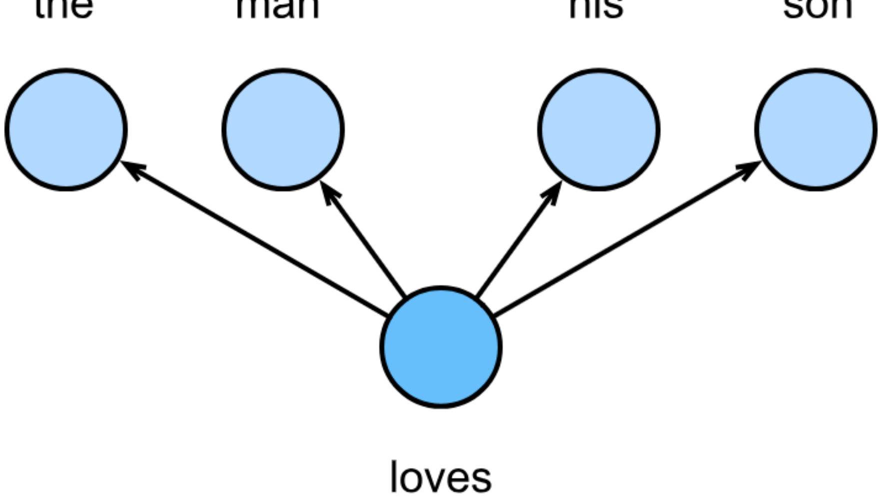
- **CBOW**
  Center word is based on the context

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



- **Skip-gram**
  Context is based on Center word

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



Likelihood Function

- Ignore order, just average

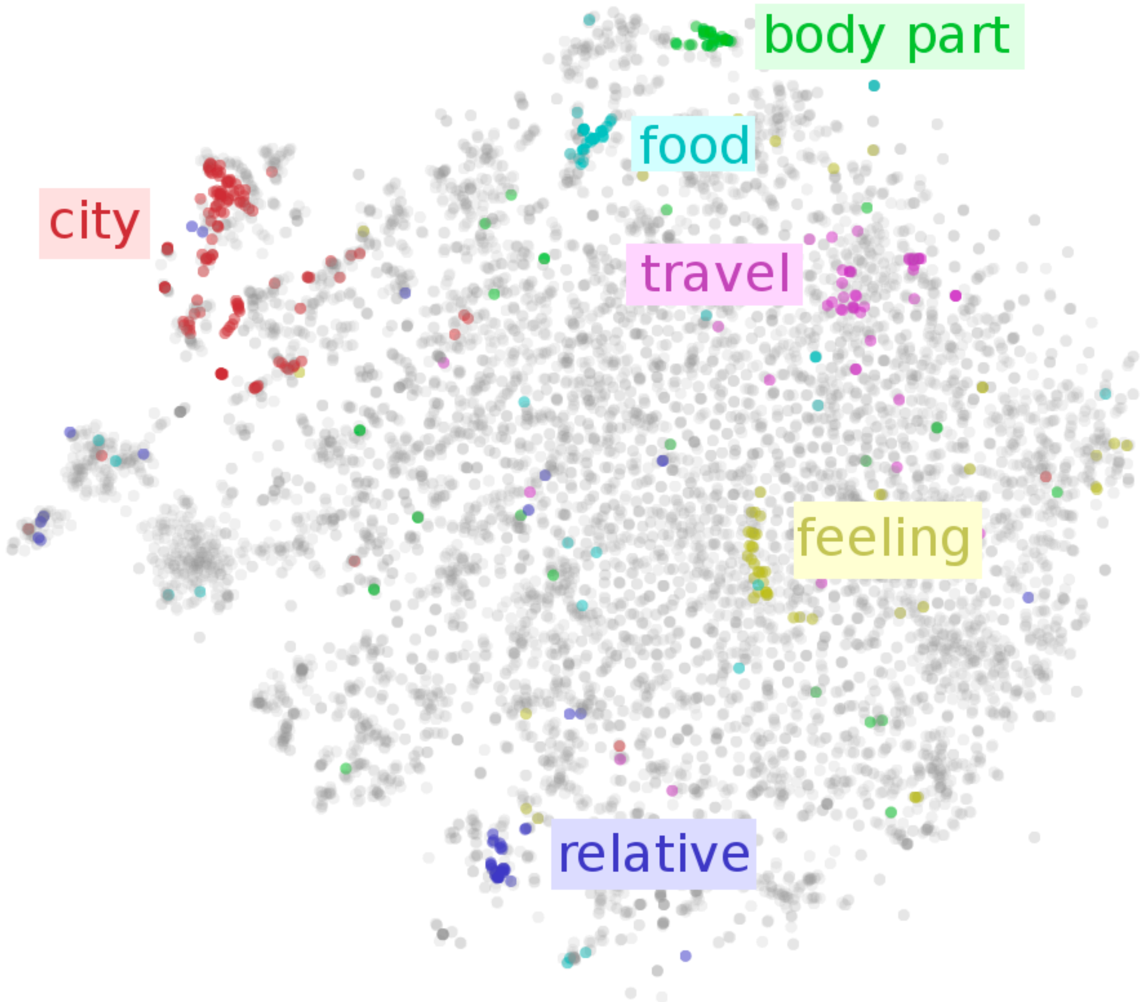
$$\phi(\text{context}) = \frac{1}{2m} \sum_{i \in [-m,m] \setminus \{0\}} \mathbf{v}_i$$

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

- 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**

$$\sum_{t=1}^{T} \sum_{j \in [-m,m] \setminus \{0\}} -\log q(w_{t+j} \mid w_t) = \sum_{w,w' \in V} -n(w', w) \log q(w' \mid w)$$

$$= \sum_{w \in V} n(w) \sum_{w' \in V} -\underbrace{\frac{n(w', w)}{n(w)}}_{=: p(w' \mid w)} \log q(w' \mid w)$$

Word Embedding with Global Vectors (GloVe)

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

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

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

$$q(w'|w) \longrightarrow \exp \mathbf{u}_{w'}^\top \mathbf{v}_w \quad p(w'|w) \longrightarrow n(w', w)$$

$$b_w + c_{w'}$$

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

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

Code…