

Lecture 8/13 (LAST CLASS, no class FRI 8/15)

- word embedding
- GloVe + HW6 (optional) demo

Logistics: HWS done (closed)

[HW6] due FRI incl demo
PROJ: submitted, look for discussion?

Introduction to Deep Learning

22. Embeddings, Word2vec, fastText, GloVe

STAT 157, Spring 2019, UC Berkeley

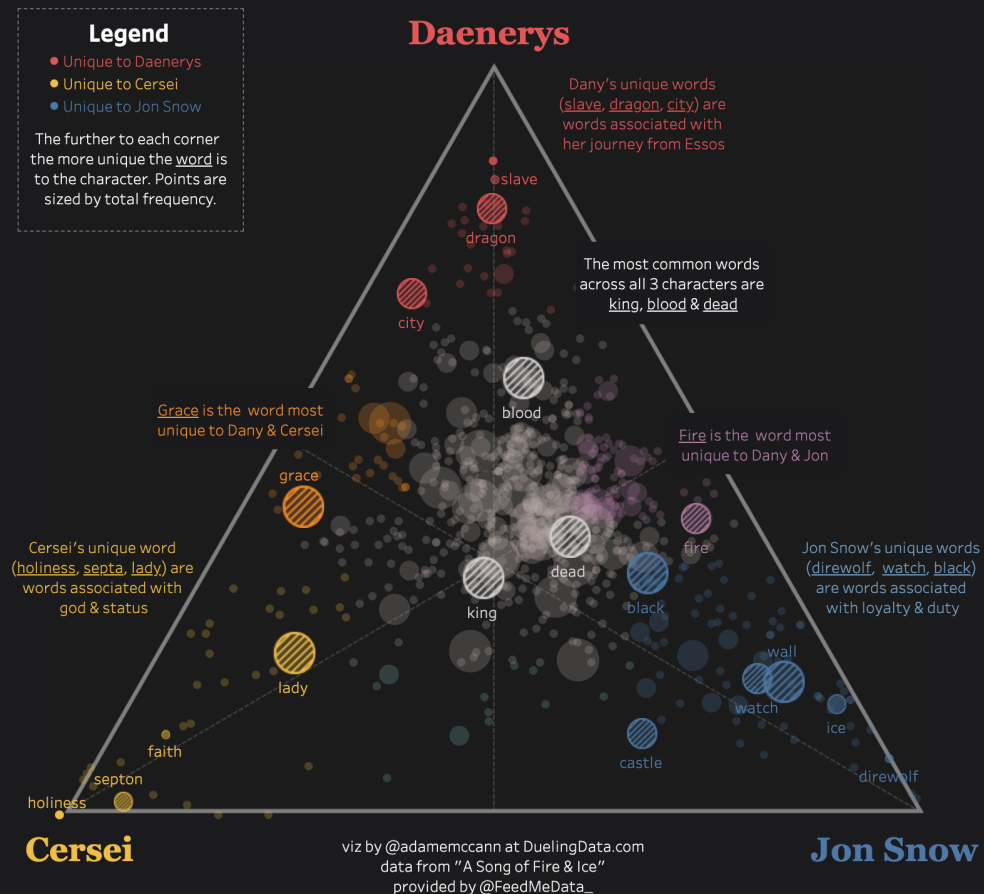
Alex Smola and Mu Li

courses.d2l.ai/berkeley-stat-157

Word2Vec

GAME OF THRONES™ IN WORDS

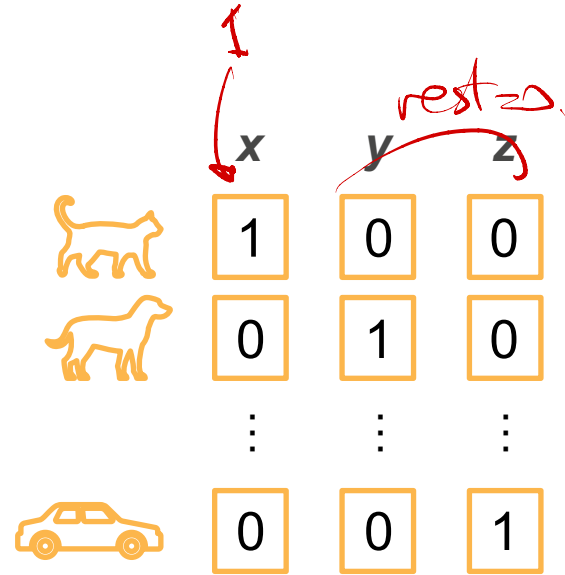
This viz shows the most unique words by character for each chapter in the 5 Game of Thrones books



Motivation

- Enc $\Rightarrow$ word ID in 300-dim vector $\equiv$ 300 words $\Rightarrow$ #dim = #vocab.*
- One-hot vectors map objects/words into fixed-length vectors
 - These vectors only contain the identity information, not semantic meaning, e.g. *no easy sim()*

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



Word2vec

- Learn an embedding vector for each word
- Use $\langle \mathbf{x}, \mathbf{y} \rangle$ to measure the similarity

dot prod for sim

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle > \langle \mathbf{z}, \mathbf{y} \rangle$$

- Build a probability model
- Maximize the likelihood function to learn the model

multiple ways

loss min $\Rightarrow$ meaningful word vectors

- *Tricks for fast computation*

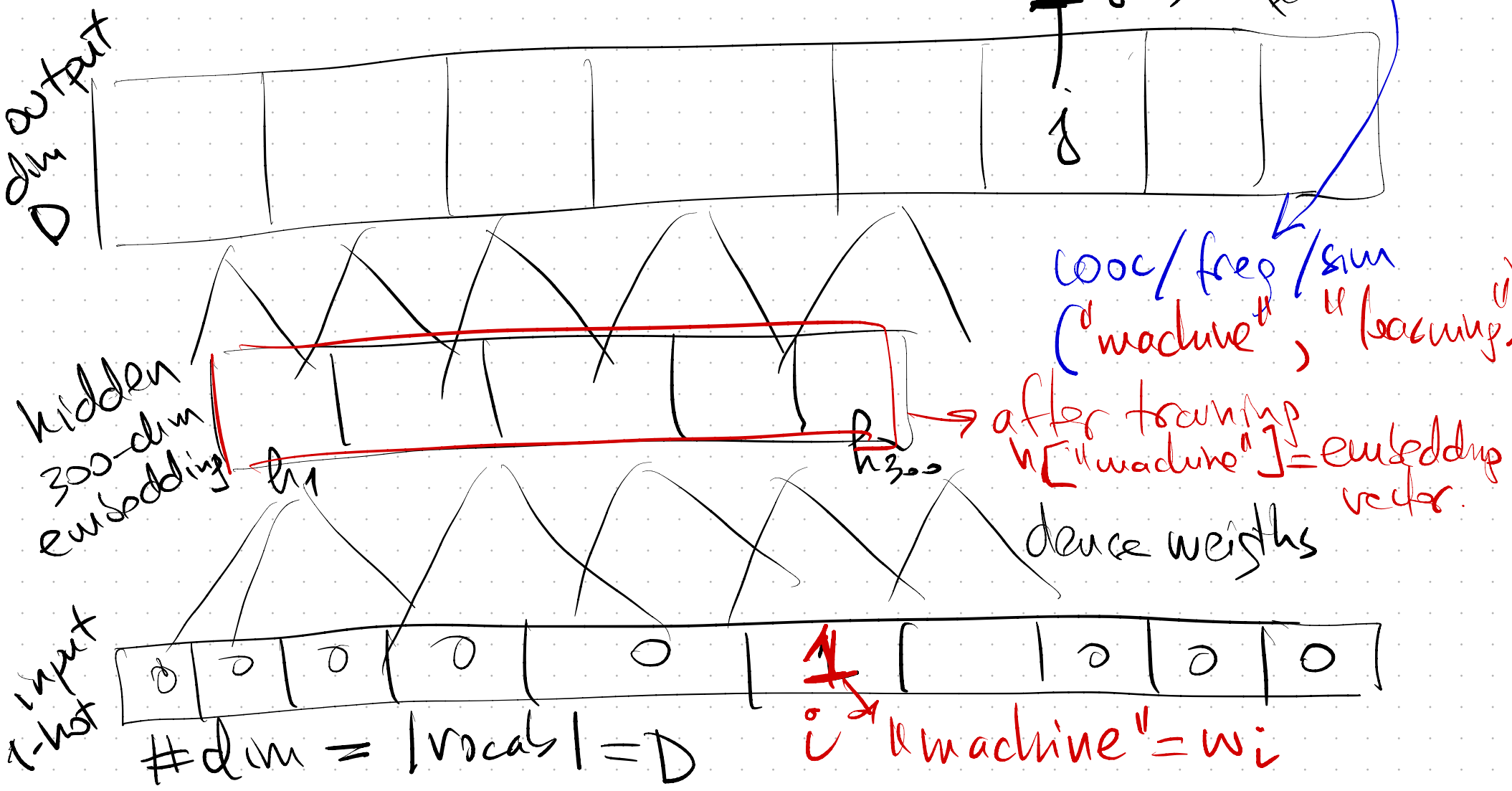
	x	y	z
	1	0	0
	0	1	0
	$\vdots$	$\vdots$	$\vdots$
	0	0	1

Simple/basic idea

- one-hot representation

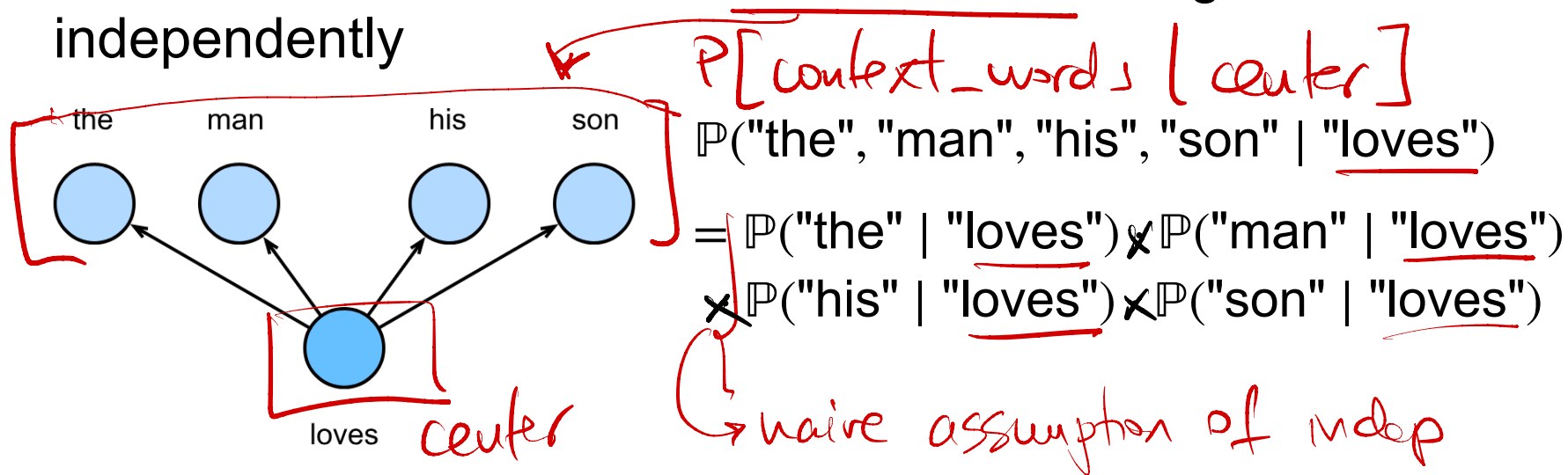
• NN-encoder/similarity

$w_j^x = \text{"learning"}$



The Skip-Gram Model

- A word can be used to generate the words surround it
- Given the center word, the context words are generated independently



Likelihood Function

Summing over all words is too expensive

softmax (dot prod) score

$P(\text{context } w | \text{center})$

$$\mathbb{P}(w_o | w_c) = \frac{\exp(\mathbf{u}_o^T \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^T \mathbf{v}_c)}$$

$\mathcal{V}$: all context words

normalization over all words.

Word Embedding

Center w_c $\mathbf{v}_c \in \mathbb{R}^d$

Context w_o $\mathbf{u}_o \in \mathbb{R}^d$

emb'd as context

- Given length T sequence, context window m , the likelihood function:

each word

naïve assumption

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

TRICKS for computation

- Simplify/approx SOFTMAX; do not compute normaliz denominator $\boxed{\text{sigmoid}(u_c^T \cdot v_o)}$ instead
— much easier to compute. $\text{prob}(\text{in window})$
- $P(D=1 | w^t, w^{t+j}) = \text{prob that } w^t, w^{t+j} \text{ are in same context window}$
- Negative sampling: include words w_n NOT in context window
— select some (not all)
— use probab $\boxed{P(\quad) = 1 - \text{sigmoid}(u_c^T \cdot v_n)}$
 $\text{prob}(\text{not in window})$
- Subsampling frequent words (pos & neg) $P(w_i) = \max\left(0, 1 - \sqrt{\frac{f(w_i)}{f_{\max}}}\right)$
not include all counts.

Negative Sampling

• Simplify/approx GPTMAX

- Treat a center word and a context word appear in the same context window as an event

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

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

$$\mathbb{P}(D = 0 | w_c, w_n) = 1 - \sigma(\mathbf{u}_n^T \mathbf{v}_c)$$

- Add into the likelihood function as well
- Maximizing the likelihood equals to solve a binary classification problem with a binary logistic regression loss

The skip-gram model parameters are the center word vector and context word vector for each word in the vocabulary. In training, we learn the model parameters by maximizing the likelihood function (i.e., maximum likelihood estimation). This is equivalent to minimizing the following loss function:

$$-\sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log P(w^{(t+j)} | w^{(t)}). \quad (15.1.6)$$

When using stochastic gradient descent to minimize the loss, in each iteration we can randomly sample a shorter subsequence to calculate the (stochastic) gradient for this subsequence to update the model parameters. To calculate this (stochastic) gradient, we need to obtain the gradients of the log conditional probability with respect to the center word vector and the context word vector. In general, according to (15.1.4) the log conditional probability involving any pair of the center word w_c and the context word w_o is

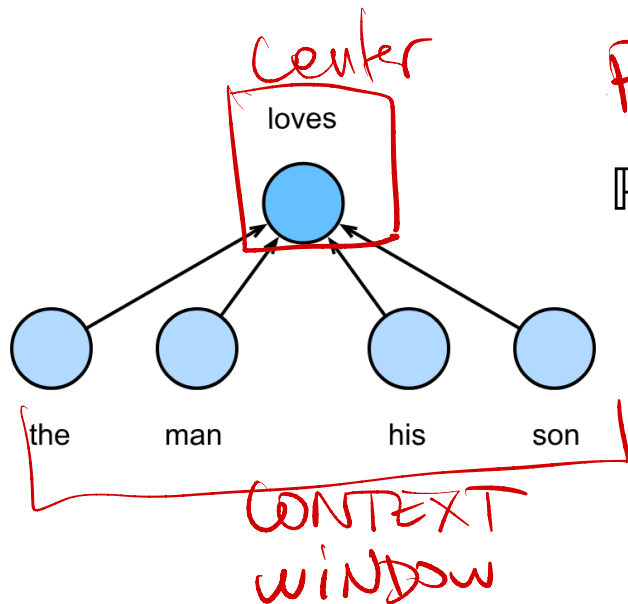
$$\log P(w_o | w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log \left(\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c) \right). \quad (15.1.7)$$

Through differentiation, we can obtain its gradient with respect to the center word vector $\mathbf{v}_c$ as

$$\begin{aligned} \frac{\partial \log P(w_o | w_c)}{\partial \mathbf{v}_c} &= \mathbf{u}_o - \frac{\sum_{j \in \mathcal{V}} \exp(\mathbf{u}_j^\top \mathbf{v}_c) \mathbf{u}_j}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \\ &= \mathbf{u}_o - \sum_{j \in \mathcal{V}} \left(\frac{\exp(\mathbf{u}_j^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \right) \mathbf{u}_j \\ &= \mathbf{u}_o - \sum_{j \in \mathcal{V}} P(w_j | w_c) \mathbf{u}_j. \end{aligned} \quad (15.1.8)$$

Continuous Bag Of Words (CBOW) *REVERSE COND PROB*

- The center word is generated based on the context words



$$P(\text{center} \mid \text{context_words})$$

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

*no naive product-prob/indep
instead evaluate score
based on all context*

Likelihood Function

- Compute the probability

$$\mathbb{P}(w_c \mid w_{o_1}, \dots, w_{o_{2m}}) = \frac{\exp\left(\frac{1}{2m} \mathbf{u}_c^\top (\mathbf{v}_{o_1} + \dots + \mathbf{v}_{o_{2m}})\right)}{\sum_{i \in \mathcal{V}} \exp\left(\frac{1}{2m} \mathbf{u}_i^\top (\mathbf{v}_{o_1} + \dots + \mathbf{v}_{o_{2m}})\right)}$$

dot product on whole window

normal

SOFTMAX

- Likelihood

each word

$$\prod_{t=1}^T \mathbb{P}(w^{(t)} \mid w^{(t-m)}, \dots, w^{(t-1)}, w^{(t+1)}, \dots, w^{(t+m)})$$

min

Training continuous bag of words models is almost the same as training skip-gram models. The maximum likelihood estimation of the continuous bag of words model is equivalent to minimizing the following loss function:

$$-\sum_{t=1}^T \log P(w^{(t)} \mid w^{(t-m)}, \dots, w^{(t-1)}, w^{(t+1)}, \dots, w^{(t+m)}). \quad (15.1.13)$$

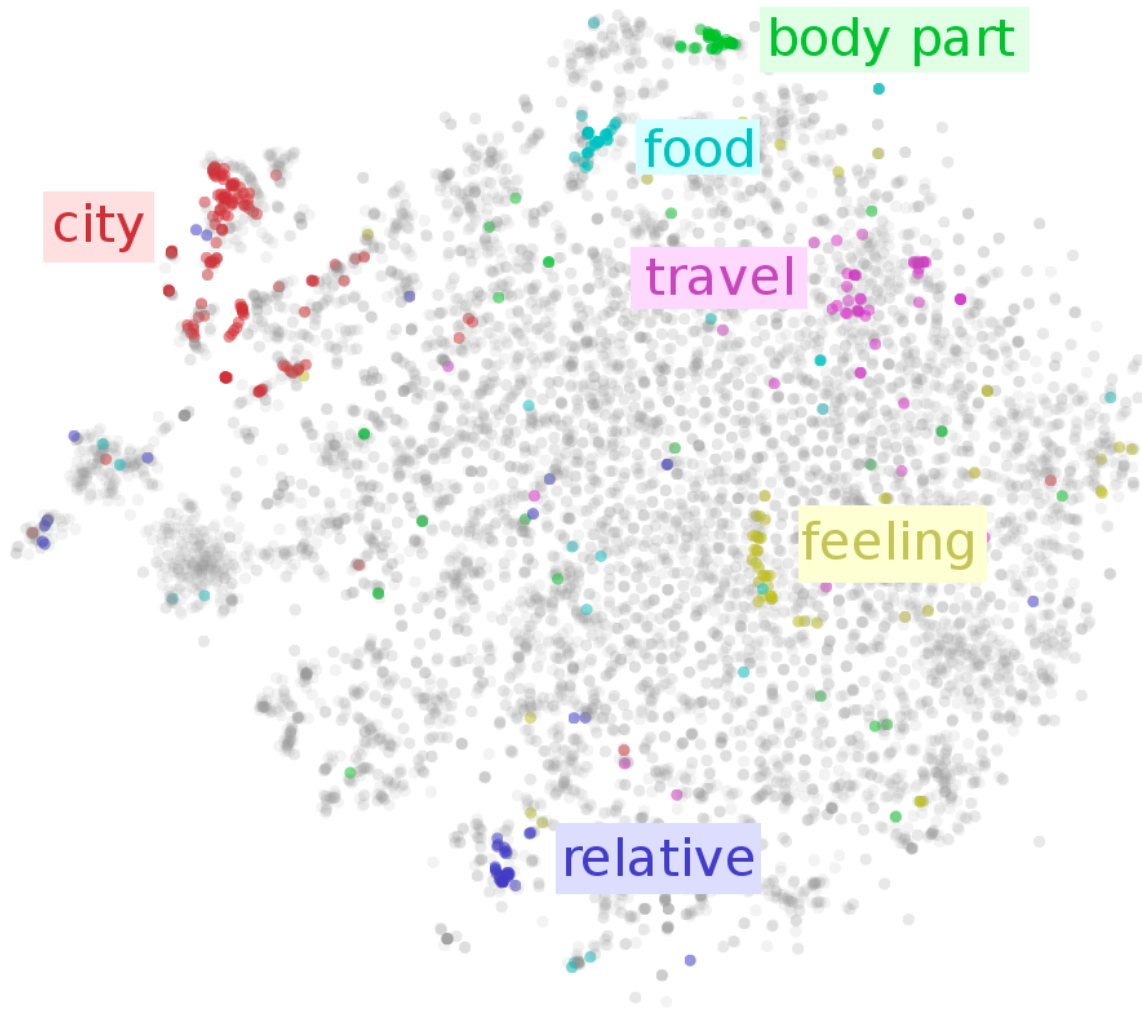
Notice that

$$\log P(w_c \mid \mathcal{W}_o) = \mathbf{u}_c^\top \bar{\mathbf{v}}_o - \log \left(\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o) \right). \quad (15.1.14)$$

Through differentiation, we can obtain its gradient with respect to any context word vector $\mathbf{v}_{o_i} (i = 1, \dots, 2m)$ as

$$\frac{\partial \log P(w_c \mid \mathcal{W}_o)}{\partial \mathbf{v}_{o_i}} = \frac{1}{2m} \left(\mathbf{u}_c - \sum_{j \in \mathcal{V}} \frac{\exp(\mathbf{u}_j^\top \bar{\mathbf{v}}_o) \mathbf{u}_j}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)} \right) = \frac{1}{2m} \left(\mathbf{u}_c - \sum_{j \in \mathcal{V}} P(w_j \mid \mathcal{W}_o) \mathbf{u}_j \right)$$

More Embedding Models



FastText

stemming / lexicographics

"friend" = "friends" = "friendship" = "friendness" . . .

- English words usually have internal structures and formation methods
 - dog, dogs, dogcatcher
- Each center word is represented as a set of subwords
 - "where" -> "<where>" -> n -gram
 - $n=3$: "<wh", "whe", "her", "ere", "re>"
- Useful for long but infrequent words
 - e.g. pneumonoultramicroscopicsilicovolcanoconiosis



embedding = $\sum \text{embed}(\text{grams})$

FastText

- For word w , $\mathcal{G}_w$ is the union of subwords with length from 3 to 6
- The center vector is then

$$\mathbf{u}_w = \sum_{g \in \mathcal{G}_w} \mathbf{u}_g$$

Handwritten red note: "sum (grams)" with an arrow pointing to the summation symbol.

- The rest model is same as skip-gram

Word Embedding with Global Vectors (GloVe)

general
wordvec
pretrained

SOFTMAX

- Denote by

$$q_{ij} = \frac{\exp(\mathbf{u}_j^\top \mathbf{v}_i)}{\sum_{k \in \mathcal{V}} \exp(\mathbf{u}_k^\top \mathbf{v}_i)}$$

(large corpus)

- Rewrite the negative log-likelihood function of skip-gram

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

tune for your
specific corpus.

- as $-\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} x_{ij} \log q_{ij}$ with proper counts x_{ij}

MULTICOUNT
Count(i, j) in context window



Glove

download
various sizes

not approx, same
as original.

- Further rewrite

$$-\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} x_{ij} \log q_{ij} = -\sum_{i \in \mathcal{V}} x_i \sum_{j \in \mathcal{V}} p_{ij} \log q_{ij}$$

with $x_i = \sum_j x_{ij}$,

$$p_{ij} = x_i / x_{ij}$$

x_i / ratio - normalized

Cross entropy

Glove

- Replace the cross entropy with a log square loss

$$\sum_{j \in \mathcal{V}} p_{ij} \log q_{ij} \rightarrow \sum_{j \in \mathcal{V}} (\log p_{ij} - \log q'_{ij})^2$$

approx
easier to
compute

with an easy to compute $q'_{ij} = \exp(\mathbf{u}_j^\top \mathbf{v}_i)$

- Add bias term for center and context words
- Replace the weights x_i with a monotone increasing function in $[0,1]$

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} h(x_{ij}) \left(\mathbf{u}_j^\top \mathbf{v}_i + b_i + c_j - \log x_{ij} \right)^2$$

monotonic func(x) control scale and sensitivity

bias precomp