Natural Language Processing
CSE 325/425

Sihong Xie

<u>Lecture 4:</u>
- Word vectors
- Word2Vec, Glove

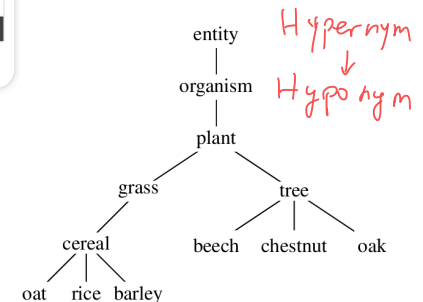# How to represent the word meanings

Meaning <=> Semantics

Use a dictionary
- an entry is a lemma;
- can have multiple senses;
- each sense can have examples of using the lemma.

Use WordNet
- Basic unit: word sense.
- Synsets: synset("good-1") = {"excellent", "great", …}
- Hypernyms: broader meanings
- Hyponyms: more specific meanings.

Advantages:

- represent human knowledge.

- sort of comprehensive.

- encode some relations.

Disadvantages:

- meanings become outdated easily.

- not so comprehensive.

- symbolic.

Handwritten annotations:

Merriam-Webster
el·o·quent
\ˈe-lə-kwənt\
adjective
1 : marked by forceful and fluent expression <an eloquent preacher>
2 : vividly or movingly expressive or revealing <an eloquent monument>
— el·o·quent·ly   adverb

Examples
· His success serves as an eloquent reminder of the value of hard

entity — *Hypernym*
organism — ↓ *Hyponym*
plant
grass   tree
cereal   beech chestnut oak
oat rice barley

ID / String → Concept

→ AI Symbolic
Statistical ML

(Not probabilistic distribution)

Distributional representation

- Non-symbolic.

- Can be learned from the latest text data.

One-hot vectors

The corpus = {
  D1=[<s> I am Sam </s>]
  D2=[<s> Sam I am </s>]
  D3=[<s> I do not like green eggs and ham </s>]
}

Index of "sam"
↓ in the V

Sam = [0 0 0 0 0 0 0 1 0 0]

I= [1 0 0 0 0 0 0 0 0 0]

ham = [0 0 0 0 0 0 0 0 0 1]

Vector dimension = number of words in vocabulary (e.g., 500,000)

V = [<s> , I , am, Sam , do , not , -- </s>]
     ↑   ↑   ↑   - - -
     0   1   2

In ML, Stat, AI,
cosine similarity is used

to measure similarity of two vectors

$$\cos \alpha = \frac{\langle \vec{u}, \vec{v} \rangle}{\|\vec{u}\| \times \|\vec{v}\|}$$
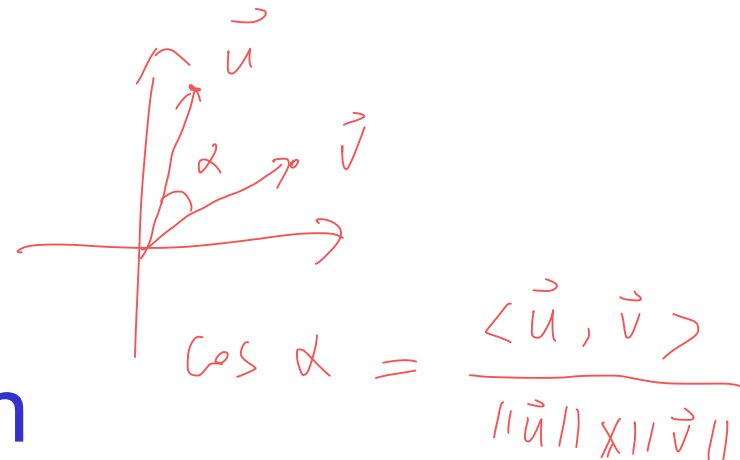
Distributional representation

Issues with one-hot vectors

- Most document vectors are orthogonal to each other and have 0 similarity
  - find articles related to "ham", while "egg" won't show up.
- WordNet's synset can help, but it is not a good solution: incompleteness.
- Extremely high dimensional

The corpus = {
   D1=[<s> I am Sam </s>]
   D2=[<s> Sam I am </s>]
   D3=[<s> I do not like green eggs and ham </s>]
}

Sam = [0 0 0 0 0 0 0 1 0 0]

I= [1 0 0 0 0 0 0 0 0 0]

ham = [0 0 0 0 0 0 0 0 0 1]

Vector dimension = number of words in vocabulary (e.g., 500,000)

$$D1 = [0\ 1\ 0\ 0\ 1 \ - - - \ |\ ,\ 0\ 0\ 0\ 1\ |\ 0]$$

$$D2 = [0\ 1\ 0\ 0\ 1 \ - - - \ |\ ,\ 0\ 0\ 0\ 1\ |\ 0]$$

$$\cos(D1, D2) = 1$$

maximum similarity

Represent word semantics by their context

A very successful principle in NLP

"You shall know a word by the company it keeps" (J. R. Firth 1957: 11)

Context of a word `W`:

• other words that appear closeby the word.

• reflect what `W` means.

• "closeby" can be defined by a window of 5 words, a sentence, a document.

The corpus = {
*   D1=[<s> I am Sam </s>]*
*   D2=[<s> Sam I am </s>]*
*   D3=[<s> I do not like green eggs and ham </s>]*
}

a small window:

      context("I") = {"am", "do"}

a larger window:

      context("I") = {"am", "Sam", "not"}

Represent word semantics by their context

A dense word vector of 50 ~ 300 dimensions:

- each dimension represents some meaning: capital city, is a number, is a noun, etc.

- two vectors are similar if they appear in each other's context.

- also called: word embedding, word representation.

- Why need more than two dimensions? Two words can be similar in different senses.

  - The following example shows a not-so-good word embedding.
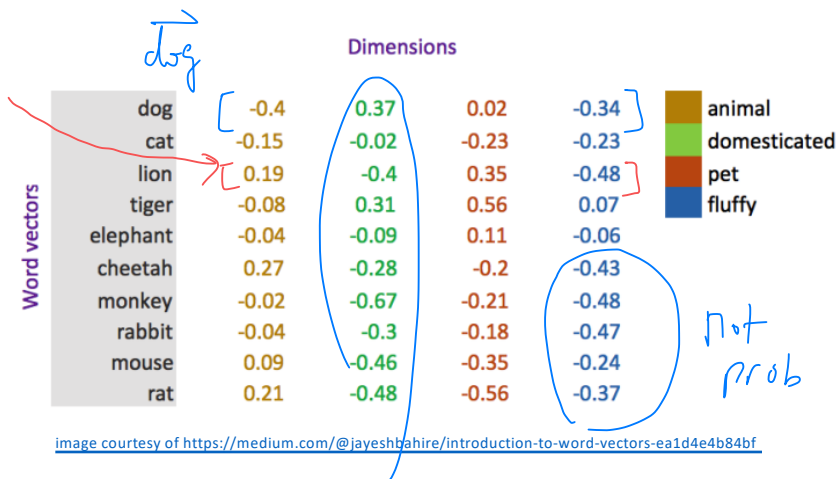

	Dimensions				
dog	-0.4	0.37	0.02	-0.34	animal
cat	-0.15	-0.02	-0.23	-0.23	domesticated
lion	0.19	-0.4	0.35	-0.48	pet
tiger	-0.08	0.31	0.56	0.07	fluffy
elephant	-0.04	-0.09	0.11	-0.06	
cheetah	0.27	-0.28	-0.2	-0.43	
monkey	-0.02	-0.67	-0.21	-0.48	
rabbit	-0.04	-0.3	-0.18	-0.47	
mouse	0.09	-0.46	-0.35	-0.24	
rat	0.21	-0.48	-0.56	-0.37	

Word vectors

image courtesy of https://medium.com/@jayeshbahire/introduction-to-word-vectors-ea1d4e4b84bf

(handwritten annotations): dog, lion, Tion, not prob

A single dimension will not describe a concept in full

Topic model (Bayesian Network)

$$\cos(\text{"dog"}, \text{"lion"}) = \frac{\langle \vec{dog}, \vec{lion} \rangle}{\|\vec{dog}\| \times \|\vec{lion}\|}$$
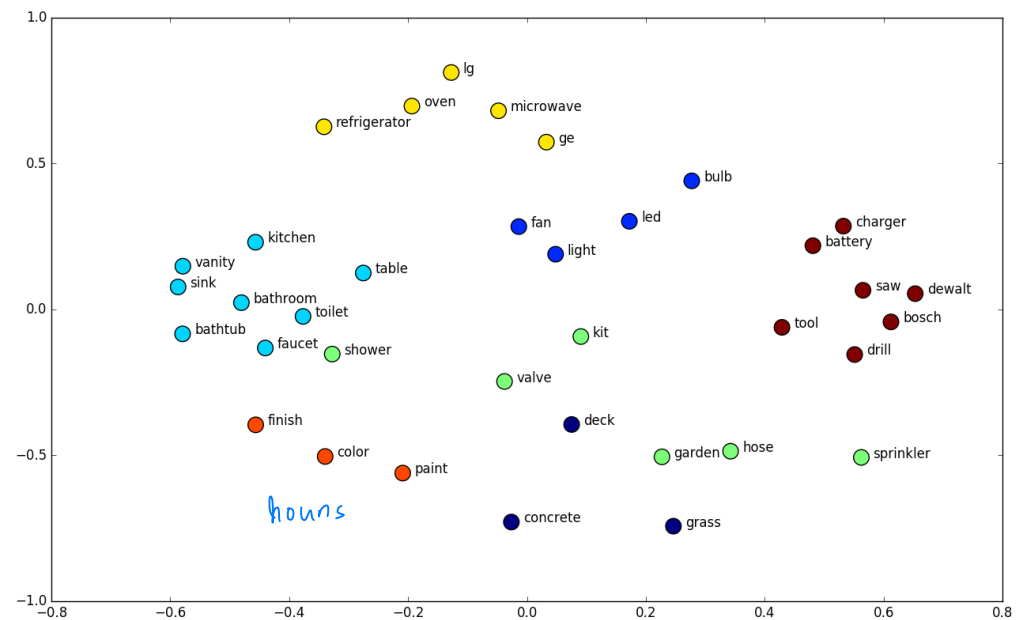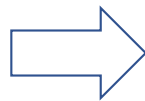
cat, dog, "mouse"
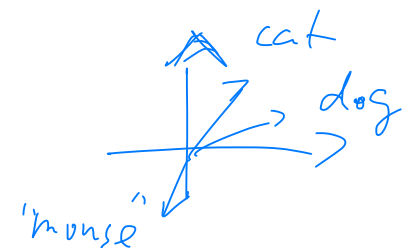
Visualization of word vectors

$t-SNE$

$$
fan \;=\; \begin{bmatrix}
0.286 \\
0.792 \\
-0.177 \\
-0.107 \\
0.109 \\
-0.542 \\
0.349 \\
0.271 \\
0.487
\end{bmatrix}
$$

Further
embedding

houns

image courtesy of https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

$$LM = \{ \text{unigram}, \text{bi-gram}, \text{tri-gram}, \text{word2vec}, \text{HMM}, \ldots \}$$

$$P(w) \qquad P(w_t \mid w_{t-1})$$

Word2Vec: overview

Tomas Mikolov, etc. **Efficient Estimation of Word Representations in Vector Space**, NIPS, 2013

Ideas:

- Learn vectors for many words (millions) from large corpus (billions).
- It is a neural language model: use neural networks to predict probabilities.
- Use center word to predict context word (Ski-gram) or the other way around (CBOW).
- Iterate through the corpus and adjust the word vectors to maximize the likelihood of word prediction.

Word2Vec: overview

Finding pairs of (center, context) words:

center word → context word in the window →

window size = 5

I do [not like green eggs and] ham

Use the center word to predict the context words:

- maximize the conditional probabilities.
- context words are assumed to be conditional independent.
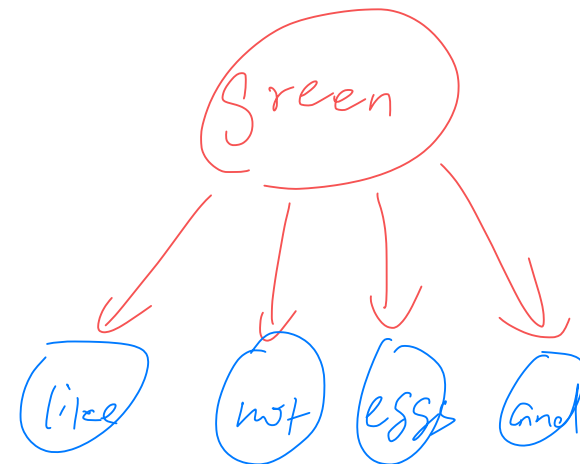
observe "like" given "green"

sufficient statistic

P(*like* | green)   P(*eggs* | green)

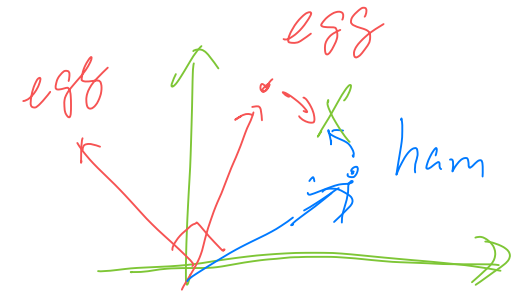P(*not* | green)   P(*and* | green)

green → like, not, eggs, and

Word2Vec: overview

Finding pairs of (center, context) words:

center word · context word in the window

I do not [like green **eggs** and ham]

$?\begin{bmatrix} \cdot \\ \cdot \\ \circ \end{bmatrix}$ $\begin{bmatrix} \cdot \\ \circ \end{bmatrix}?$

P(*like* | egg)   P(*ham* | egg)

P(*green* | egg)   P(*and* | egg)

$V_{"egg"} = \begin{bmatrix} \circ \\ \cdot \\ \circ \end{bmatrix}$ $\max_{U_{ham}} P(ham | egg) = \frac{\exp(U_{"ham"} V_{"egg"})}{\sum \exp(U_w V_{"egg"})}$

$U_{"ham"} = \begin{bmatrix} \circ \\ \vdots \\ \circ \end{bmatrix}$ V_{egg}

θ to be optimized

any w in vocab

Alg MLE

Input : $\mathcal{D}$ observed sufficient statistics

$P(\mathcal{D}|\theta)$

output : $\theta^* = \arg\max_\theta P(\mathcal{D}|\theta)$

Word2Vec: loss function

Maximum Likelihood Est.

(MLE)

For each position $t = 1, \ldots, T$

predict context words w_{t+j} within a window of fixed size m, given center word w_t

$m = 2$

Context

Likelihood:
$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{j=-m \\ j \neq 0}}^{m} P(w_{t+j}|w_t)$$

$\begin{bmatrix} w_{t-2} & w_{t-1} & w_t & w_{t+1} & w_{t+2} \end{bmatrix}$

Indexing

Indexing the center word   the context

Center $m = 2$

Loss function (negative log-likelihood, or cross entropy loss):

$$-\log L(\theta) = -\sum_{t=1}^{T} \sum_{j=-m, j \neq 0}^{m} \log P(w_{t+j}|w_t)$$

T in the scale of billions

with millions of word vectors

Word2Vec: parametrization

How to represent the probability using some parameters?

- In bi-gram model, we use a single number to represent a probability.

- Neural networks allow more complex parametrization of common

  probability distributions.

  - Logistic regression parametrized a binomial distribution.

  - Multi-layered Perceptron (MLP) can output a multinomial distribution.

Softmax as distribution over the vocab:

Let each word have two parameter vectors:

- $\mathbf{v}_w$ word vector of any w when used as a center word.

- $\mathbf{u}_w$ word vector of any w when used as a context word.

$$P(w_o|w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_w \exp(\mathbf{u}_w^\top \mathbf{v}_c)}$$

$\in Vocab$

$$\sum_{\substack{w_o \\ \in Vocab}} p(w_o|w_c) = 1$$

Word2Vec: parametrization

Softmax as distribution over the vocab:

$$P(w_o|w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_w \exp(\mathbf{u}_w^\top \mathbf{v}_c)}$$

- It is indeed a probability distribution: non-negative and sum-to-one.

- It is "max" since the exp enlarge the gap between different inner products.

- It is "soft" since it is does not assign all probability to the maximum.

Word2Vec: optimization

Want to minimize the loss

$$-\log L(\theta) = -\sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log P(w_{t+j}|w_t) \qquad P(w_o|w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_w \exp(\mathbf{u}_w^\top \mathbf{v}_c)}$$

- The parameter θ to be optimized is the (2 x V) d-dimensional word vectors.

- Stochastic gradient descent

  - sample a batch of (center, context) pairs.

  - compute the gradient of the loss with respect to the relevant vectors.

Alg "SGD"

Init randomly word vectors

for $i = 1, \cdots \infty$ // an epoch

    Sample a mini-batch of B observed (w_c, w_o) pairs

    eval $-\log L(\theta)$ using the mini-batch, and find grad to update the involved word vectors

Word2Vec: optimization

gradient descent on u_o:

$$\begin{cases} u_o \leftarrow u_o - \eta \dfrac{\partial J}{\partial u_o} \\[2mm] = u_o + \eta \underbrace{[1 - P(w_o|w_c)]}_{>0} v_c \end{cases}$$

$\nearrow$ learning rate >0

pushes u_o towards v_c $\Longleftarrow$ making them closer

Want to minimize the loss

$$-\log L(\theta) = -\sum_{t=1}^{T} \sum_{j=-m, j\neq 0}^{m} \log P(w_{t+j}|w_t) \qquad P(w_o|w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_w \exp(\mathbf{u}_w^\top \mathbf{v}_c)}$$

Let's focus on one (center, context) pair. The gradient w.r.t. the center word vector is:

$$J = -\log P(w_o|w_c) = -\log \exp(u_o^\top v_c) + \log \sum_w \exp(u_w^\top v_c)$$

$$\frac{\partial J}{\partial (u_o^\top v_c)} = -1 + \frac{1}{\sum_w \exp(u_w^\top v_c)} \cdot \exp(u_o^\top v_c) = P(w_o|w_c) - 1$$

$$\frac{\partial J}{\partial u_o} = \frac{\partial J}{\partial (u_o^\top v_c)} \cdot \frac{\partial (u_o^\top v_c)}{\partial u_o} = \Big[P(w_o|w_c) - 1\Big] \cdot v_c \qquad \text{see above}$$

Word2Vec: optimization

Optimizing the original loss is too expensive:

- Due to the denominator in the softmax.

- Negative sampling: reduce the number of summands in the denominator.

$$P(w_o | w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_w \exp(\mathbf{u}_w^\top \mathbf{v}_c)}$$

Stochastic gradient descent is more preferred:

- With billions of tokens and millions of words, computing the gradient using all training (center, context) pairs can be slow.

- Same a small batch (mini-batch) of training pairs and just optimized the vectors of the selected words => a very sparse gradient.

Word2Vec: negative sampling

Turn the problem into a binary classification problem

- Classify the context word from the words outside the window centered at w_c

$$k = 10$$

$$J(w_o, w_c, \theta) = -\log \sigma(\mathbf{v}_o | \mathbf{v}_c) - \sum_{k=1}^{K} \log(1 - \sigma(\mathbf{v}_k | \mathbf{v}_c))$$

- How to sample the non-context words? $\tilde{P}(w) = P(w)^{3/4}$
- How many to sample? K=10 is good enough.

$P(w)$: relative frequency