Glove: Global Vectors

The ratios of co-occurrence probabilities matters

- The contrast of two probabilities remove the less salient contexts.
- It is not sensitive to the scale of the probabilities.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

image courtesy of Stanford NLP group

vocabulary $\left\{ \begin{array}{l} w_1 \rightarrow [\quad\quad] \\ \vdots \\ w_{|V|} \rightarrow [\,.\ .\ .\ .\ .\ .\,] \in \mathbb{R}^d \end{array} \right.$ $d=300$

window

$[\ w_j \quad w_i\ ]$

Glove: Global Vectors

The ratios of co-occurrence probabilities can be predicted by a neural network

context $\quad$ center

- context-center words are symmetric: $\mathbf{v}_i^\top \mathbf{v}_j = \log P(w_i|w_j)$

$\vec{v}_i = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \vec{v}_j = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

- $\mathbf{v}_k^\top(\mathbf{v}_i - \mathbf{v}_j) = \log \dfrac{P(w_k|w_i)}{P(w_k|w_j)}$

$\vec{v}_i^\top v_j = 1\times(-1) + 2\times 2 = 3$

- Let X_{ij} be the number of times context word j co-occurs with the center word i.

$$\mathbf{v}_k^\top \mathbf{v}_i = \log X_{ik} - \log X_i$$

$$\tilde{\mathbf{v}}_k^\top \mathbf{v}_i + b_i = \log X_{ik}$$

$$\tilde{\mathbf{v}}_k^\top \mathbf{v}_i + b_i + \tilde{b}_k = \log X_{ik}$$

$P(w_k | w_i) = \dfrac{X_{ik}}{X_i}$ = MLE of Bigram

where X_{ik} = frequency that w_k happens in context of w_i

$X_i = \Sigma_k X_{ik}$ = frequency of w_i

Glove: weighting

It is common in machine learning to stress certain observations.

- Focus on context words that are closer than those farther away.
  - count co-occurrences distance = 2

I do not [like green eggs and ham]

distance = 1

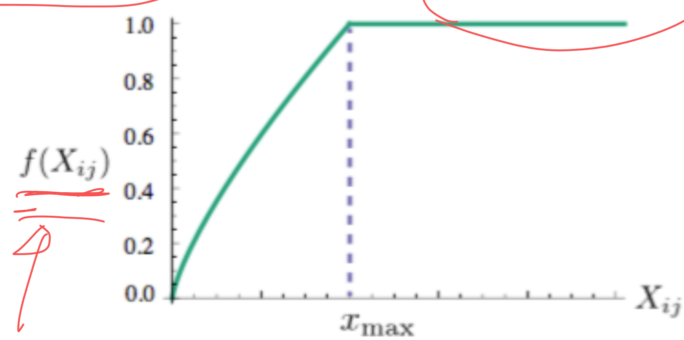
(1+0)

$X_{egg,\ and}$ $+= 1/1$ $X_{egg,\ green}$ $+= 1/1$

$X_{egg,\ ham}$ $+= 1/2$ $X_{egg,\ like}$ $+= 1/2$

(1+1)

- But don't over emphasize frequent co-occurrences:
  - This can happen for common words, not just stop-words. "a", "the", "is", "are" --
  - Idea: cap the X values.



See the Glove paper

Glove: windows

Different design of the context window lead to different results

- Larger windows capture more semantic information 7~11
  - e.g., good ~ great, king ~ queen
- Smaller windows capture more synatic information 3~5
  - walking vs. dancing: their closest contexts are quite similar. present tense
- Symmetric vs. Asymmetric windows
  - Symmetric ones capture semantics
  - Asymmetric ones find syntactic structures: syntaxes has orderings.

shared context

is walking

is dancing

2 words 2 words

[← w_i →]

center

[w_i - - - - - -]

center

Project 1 will adopt this schema.

Word embedding evaluation of a language model

Word analogies

- a is to b as c is to ?
  - Semantics: "Athens is to Greece as Berlin is to ?"
  - Syntactics: "dance is to dancing as fly is to ?"

$$w_b - w_a + w_c = ?$$

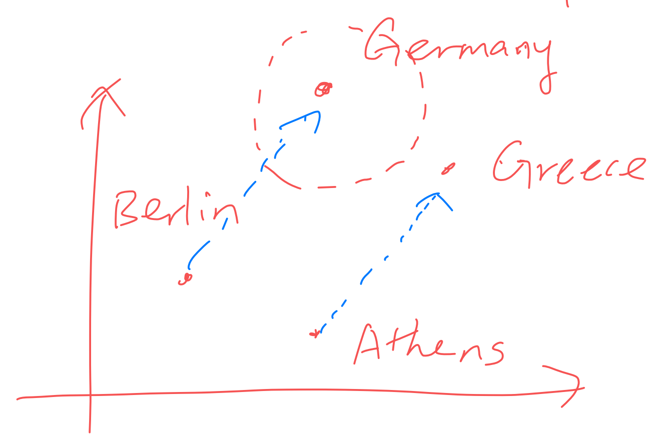
Word similarity

- Human-compiled pairs of similar words.
- http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Helping the end-task?

- Named entity recognition (sequence-to-sequence model): CoNLL-2003
- Combine continuous word vectors with 437,905 discrete features

word vector space



Word embedding evaluation

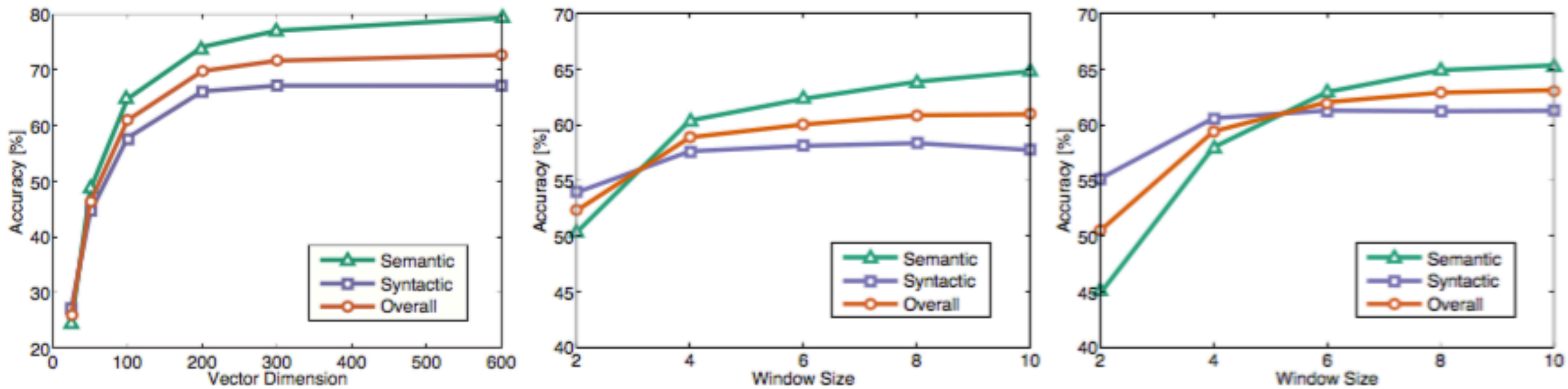
Table 3: Spearman rank correlation on word similarity tasks. All vectors are 300-dimensional. The CBOW* vectors are from the `word2vec` website and differ in that they contain phrase vectors.

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	<u>72.7</u>	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	<u>58.1</u>	37.2
GloVe	6B	<u>65.8</u>	<u>72.7</u>	<u>77.8</u>	53.9	<u>38.1</u>
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Observations on measuring word similarity:

- The more training data the better for Glove.
- Skip-gram is better than CBOW.
- CBOW can’t benefit from 100B tokens.
- SVD, if scaled properly, is a strong baseline

Word embedding evaluation

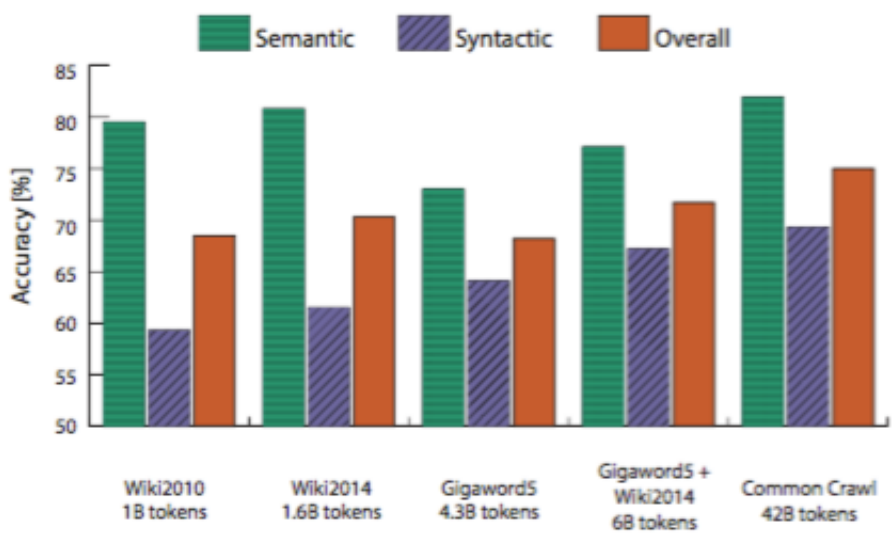


(a) Symmetric context (b) Symmetric context (c) Asymmetric context

Observations on the word analogy task:

- Larger dimensionality is better and does not hurt performance.
- Larger windows are good for measuring semantics.
- Smaller windows are good enough for measuring syntactics.
- Asymmetric windows work better than symmetric ones in capturing syntactics.

Word embedding evaluation

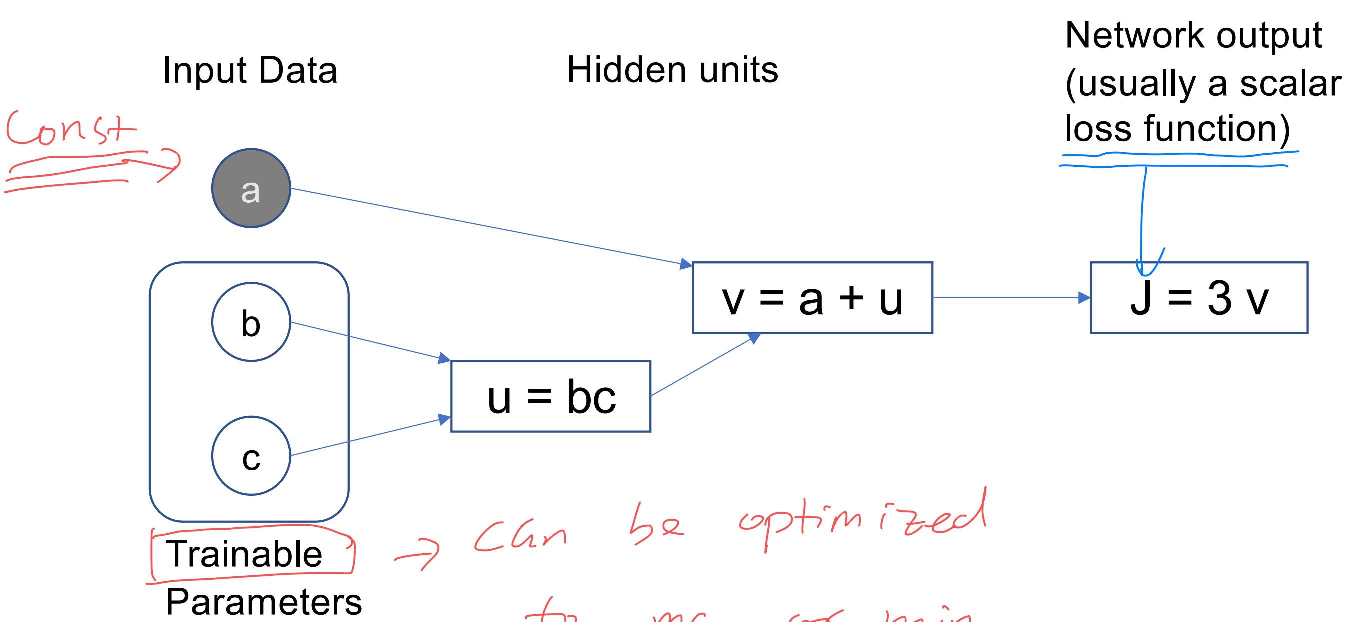


Observations on measuring word similarity:

- The larger the corpus the better.
- Depending on the task, need to select the right corpus: wikipedia is more comprehensive than news.
- Combining multiple corpora can hurt semantic measuring, but can help learning synatics (which is more general across corpora).

Neural networks (simplified)

Forward computation on a computation graph



→ can be optimized to max or min some objective function

Leaves

Forward Propagation

① $u = b \times c$

② $v = a + u$

$= a + bc$

③ $\boxed{\begin{aligned} J &= 3v \\ &= 3(a+bc) \end{aligned}}$

what the computation graph implements

$\boxed{\text{Alg:}}$ Step 1: forward propagation

Take two words that co-occured

compute $f(x_{ij})(w_i^T \tilde{w}_j - \log X_{ij})^2 = J(w_i, \tilde{w}_j)$

"Stochastic gradient descent"

Neural networks

Step 2: Backward Propagation

find $\frac{\partial J}{\partial w_i}$, $\frac{\partial J}{\partial \tilde{w}_j}$

Step 3: Take the gradient descent

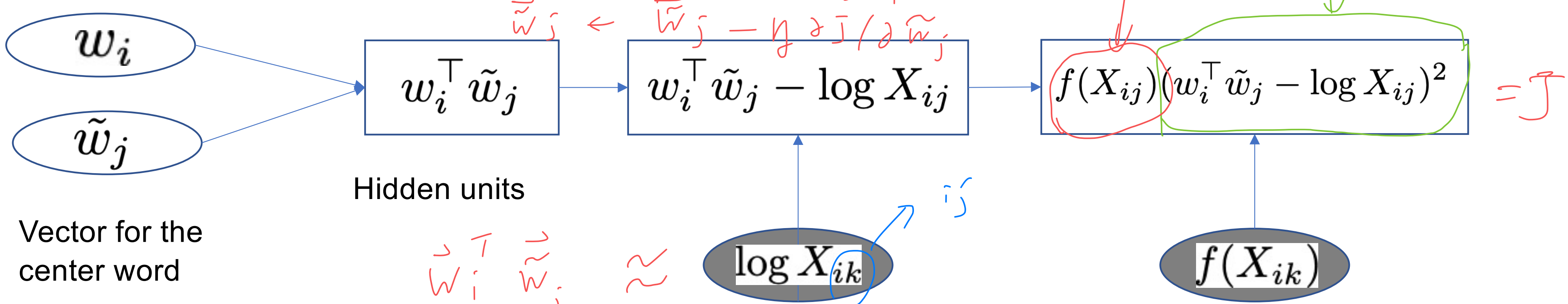
Computation graph for Glove

$\vec{w}_i \leftarrow \vec{w}_i - \eta \frac{\partial J}{\partial w_i}$

$\vec{\tilde{w}}_j \leftarrow \vec{\tilde{w}}_j - \eta\, \partial J / \partial \tilde{w}_j$

Vector for the context word

Network output: weighted square error.



Hidden units

$\vec{w}_i^T \vec{\tilde{w}}_j \approx$

Vector for the center word

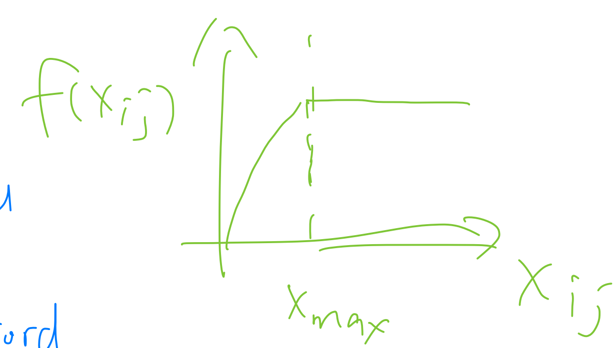
Log of co-occurrence frequency

Weight of the co-occurrence

Each word has two word vectors

$\vec{w}_i$: when w_i used as a center word

$\vec{\tilde{w}}_j$: when w_j used as a context word.



$$b \leftarrow b - \eta \boxed{\frac{\partial J}{\partial b}}$$

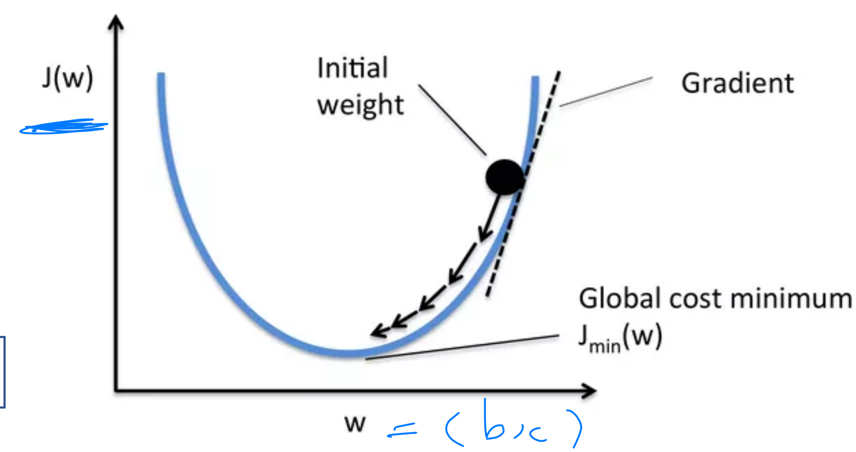
$$c \leftarrow c - \eta \boxed{\frac{\partial J}{\partial c}}$$

Fixed points $\stackrel{?}{=}$ (b_0, c_0) where $\left.\frac{\partial J}{\partial b}\right|_{b=b_0}=0$, $\left.\frac{\partial J}{\partial c}\right|_{c=c_0}=0$

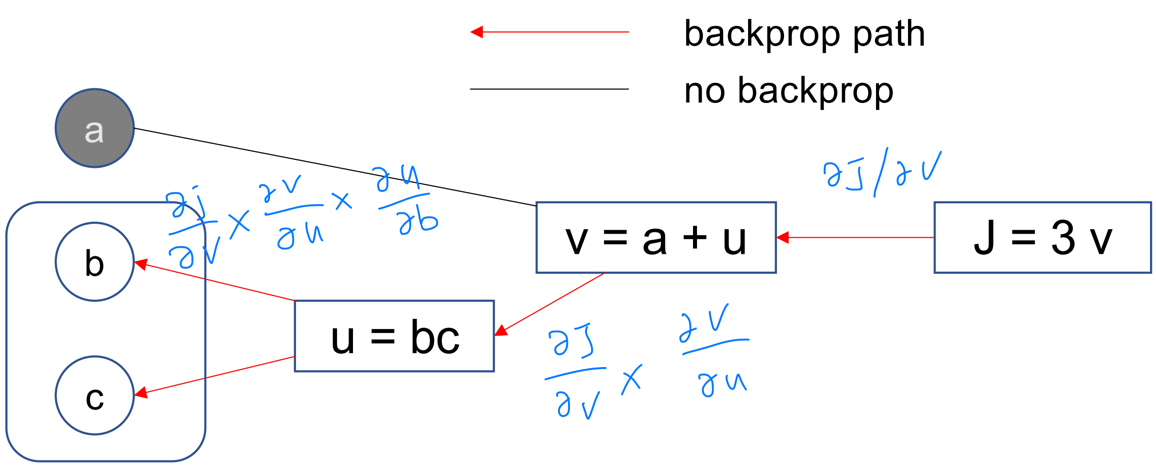
Gradient descent

Neural networks (simplified)

Training error



Backward computation for training the network.



Backward Propagation

$$\boxed{\frac{\partial J}{\partial b}} = \boxed{\left(\frac{\partial J}{\partial v} \times \frac{\partial v}{\partial u}\right)} \times \frac{\partial u}{\partial b}$$

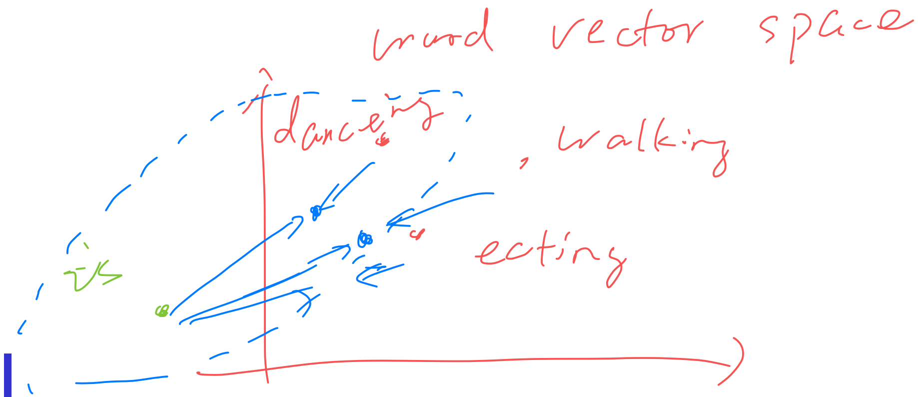
$$= \frac{\partial J}{\partial u} \times \frac{\partial u}{\partial b}$$

$$= \frac{\partial J}{\partial b}$$

A computation graph is a differentiable system.

$$\boxed{\frac{\partial J}{\partial c}} = \boxed{\left(\frac{\partial J}{\partial v} \times \frac{\partial v}{\partial u}\right)} \times \frac{\partial u}{\partial c}$$

Implement the chain rule in Calculus



Training Glove model

Loss function $$\hat{J} = \sum_{i,j} f(X_{ij})(w_i^T \tilde{w}_j - \log X_{ij})^2$$

1. gradient descent of the loss function is slow since there are many pairs of co-occurred words.
2. stochastic gradient descent computes the loss and gradient on one randomly selected pair: fast to compute but can be noisy. $\partial J / \partial \vec{W}_{\text{"is"}}$ is noisy
3. Mini-batch gradient descent: in the middle of the above. Use multiple pairs to reduce noise in the gradient
   - Seeing pairs [(*is*, *dancing*), (*is*, *working*), …, (*is*, *eating*)] all at once is better than seeing just (*is*, *dancing*) in learning the vector for "*is*".

PyTorch

What Pytorch offers

1. Construct computation graph in a declarative way.
2. Autograd allows you to find gradients without manual calculation.
3. Sophisticated optimizers that control how to make gradient descent work.
4. GPU computing and memory management API's

You still need to:

1. design the network architecture (the graph);
2. prepare training data.
3. monitor training and evaluate models.

Now walk through Project 1 in PyTorch in Colab.

Natural Language Processing
CSE 325/425

Sihong Xie

Lecture 5:

- Part-of-Speech (POS)
- POS tagging
- Hidden Markov Models (HMM)

Part-of-Speech

English word classes: cover a few common classes that will be used in tagging.

1. Nouns: pronouns (she, he, I, who, others), proper nouns (Russia), countable nouns (desk), mass noun (air)
2. Verbs: participles (paced), gerund (pacing), auxiliaries (be, do, have, can, may, should)
3. Adjectives: comparative, superlative. → describe nouns
4. Adverbs: I went *Church* *yesterday* → describe verbs

small class of words ←

5. Prepositions: in, on, over, ...
6. Particles: phrasal verb like "go *over*". "over" adds additional meaning to "go"
   - Easy to be confused with prepositions.
   - Combination of verb and particle does not have their meanings combined simply.
7. Determiners: a, the, an
8. Conjunctions: and, but, that, when
9. Other smaller classes.

Part-of-Speech

Syntatic information: how words are ordered in a sentence.

- noun-verb
- determiner-noun   a coffee mug
- adjective-noun   cold coffee
- verb-adverb   I drink slowly
- preposition-noun   12 students on zoom today
  prep   noun

Useful for grammar checking: go to (a?the?) hospital

Semantic information: meaning of a word in a context. Useful for:

- machine translation (building a building): building -> (建 vs. 楼)
- question-answering, need to understand the semantics of what a person is asking.
- relation extraction (Bill Gates founded MS): gates (verb vs. noun)
- event extraction (They went to a concert): concert (verb vs. noun)
- entity extraction (I will visit DC): DC?