Lecture 2

After the introduction of word2vec，this lecture is about the part of **Optimization**

1. Gradient Descent

- so as said, there is a cost function $J(\theta)$ to be minimized, by calculating the gradient of $J(\theta)$ and moving θ in small steps to the direction of the gradient
- $\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$
- α is the learning rate/step size, which is a hyperparameter
- ∇ is the gradient

2. Stochastic Gradient Descent (SGD)

- obviously, the $J(\theta)$ in the gradient calculation is too massive, so for a single update, it would take a lot of time
- so this **SGD** method, repeatedly takes small random window size with **less center words**, and update the θ parameters in the same way
- **does a better job, and also in a quicker time**

3. Negative Sampling (skip Gram model)

- In **Softmax** the **denominator for the norm** consists of the huge sum of all the words in the vocabulary, which is very costly to compute

Regression (side quest)

Linear regression: is training a model in a form of $y = wx + b$, where w is the weight, and b is the bias, and learns w and b to predict y as close as possible to the real value of y

Logistic regression: here the trained model predicts the probability of y, which is in the form of $y = \frac{1}{1+e^{-wx}}$ using a sigmoid.

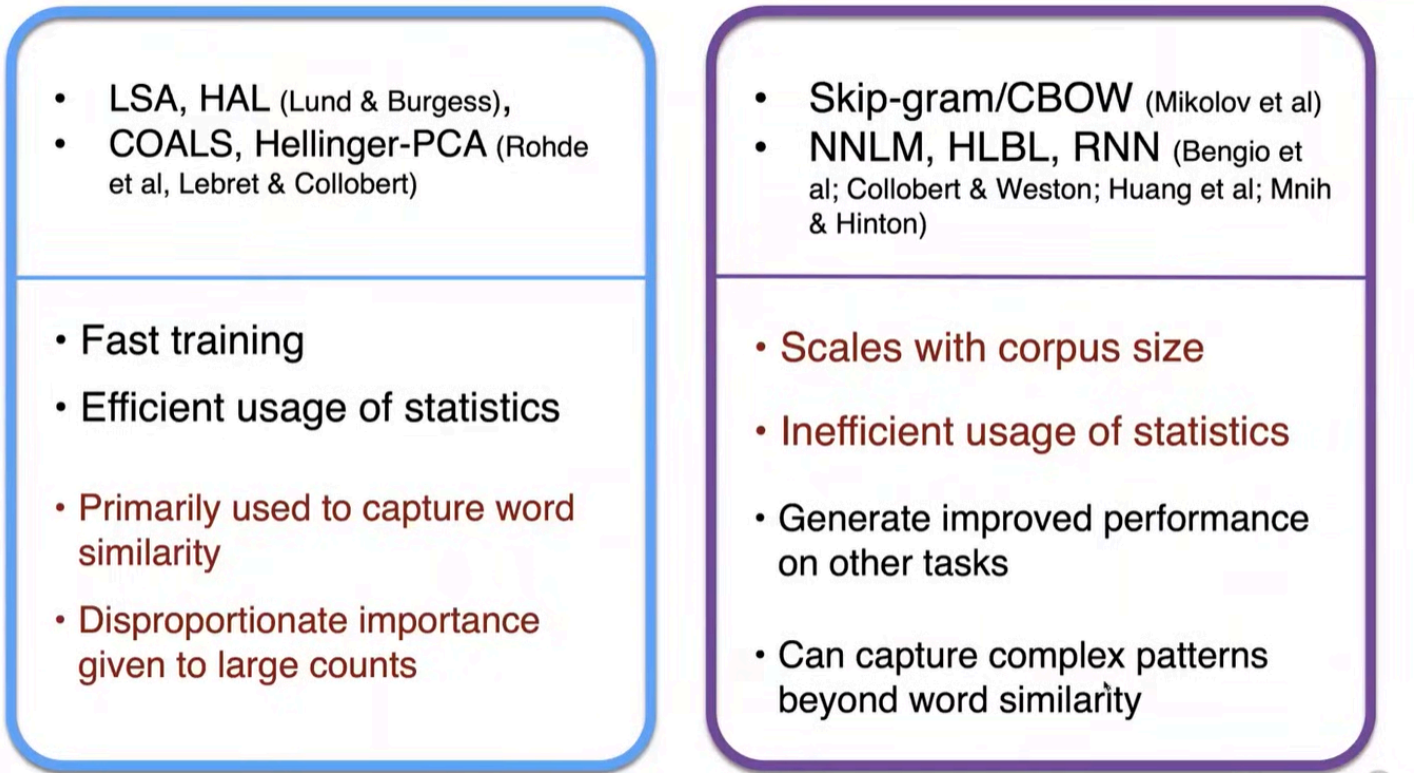
- so in the Skip Gram model, it uses logistic regression. And is trained in a way such that it uses a **true pair versus a noise pair**, (pair consists of a center word and a context word, and noise refers to a random context word for the same center word). so it gives the probability if it is true pair then P = 1, else P = 0, it is a **classifier**
- $J_t(\theta) = log(\sigma(u_o^T v_c)) + \Sigma_{i=1}^{k} E_{j\ inP(w)}[log(\sigma(-u_j^T v_c))]$
- σ as the sigmoid/logistic function, so the goal is to minimize the cost, here which means maximizing σ in which when the input dot product is large to infinity, output of σ is 1, and $log1 = 0$, similarly for noise pairs, the dot product should have negative infinity, and putting it into $\sigma(--inf)$ also gives out 0, so 0 + 0 = 0, (maximized).cuz everything else is negative.
- and negating works due to the sigmoid function is symmetric around 0.5
- And again, we are taking the **negative log likelihood**, because before 0 was max, still maximizing, but we want to minimize the cost function.
- $J_{neg-sample}(\theta) = -log(\sigma(u_o^T v_c)) - \Sigma_{i=1}^{k} E_{j\ inP(w)}[log(\sigma(-u_j^T v_c))]$
- maximize real pair, minimize random pair

4. Co-Occurence Matrix $\neq$ Word2vec

- it is a symmetric matrix, depending on the window size, it places the number of occurences of a word in the context window of another center word. like row for context, column for center word.
- **Reducing dimension on Co-occurence Matrix:** By decomposing the matrix through **SVD**, getting n singular values,
- Stores most information in a smaller number of dimensions.
- **Problems:** for processing the values:
  - As running raw counts on SVD, it wouldnt work well
- **Fix:**
  - make the maximum limit of 100
  - we can log the frequencies
  - also remove the frequently occuring function words like "the/he/has"
  - so you get more useful word vectors

Comparisons, introducing GloVe

LHS uses Linear algebra
RHS uses gradient neural models



- skip gram lacks statistics , because it only looks a one word at a time, unlike the co-occurence matrix which looks at the whole window

5. GloVe

Meaning Components: words co-occuring with eachother

- level of co-occurence represented with the **ratio of probabilities**.
- for example **ice to solid** rather than gas
-

	x = solid	x = gas	x = water	x = random
$P(x\|\text{ice})$	large	small	large	small
$P(x\|\text{steam})$	small	large	large	small
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	large	small	~1	~1

- so here, you can see that ice and steam is **both similar** to water, so the ratio is ~ 1, nice!! however for both random words to ice and steam with **no correlation**, the ratio is also ~1
- So for plotting we would normally try to find one that is related but the other one is not.

Capturing the ratio:

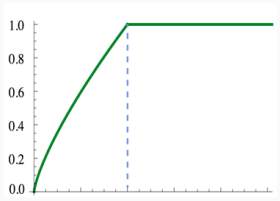
- using a log bilinear model
- The GloVe model takes both the co-occuring and neural model's proprties. (count based & dot product learning)

Q: How can we capture ratios of co-occurrence probabilities as linear meaning components in a word vector space?

A: Log-bilinear model: $w_i \cdot w_j = \log P(i|j)$

with vector differences $w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$

Loss: $J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$

$f \sim$ 

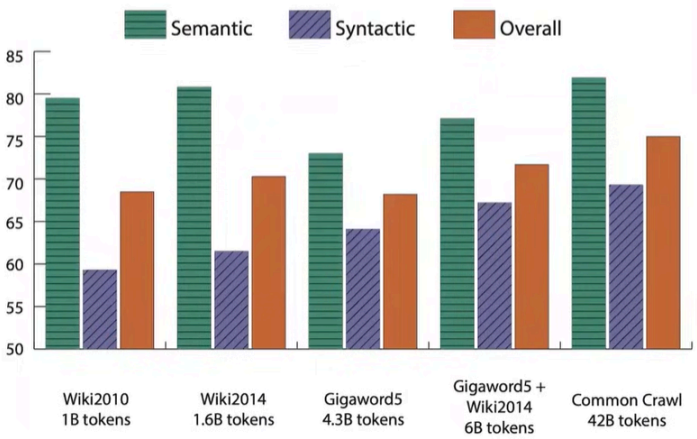
- Fast training
- Scalable to huge corpora

- the bilinear model is just the defining the dot product to be the log of the ratio of the probabilities
- For the **loss function** X_{ij} is form the co-occurence matrix, how many times the context j appeared in the window of center word i
- so then, the model needs to adjust the weight and biases so for every pair (i, j), you want the model's prediction $w_i^\top \tilde{w}_j + b_i + \tilde{b}_j$ to be close to the "target" value $\log X_{ij}$.
- $f(X_{ij})$ is for scaling the words that are more common

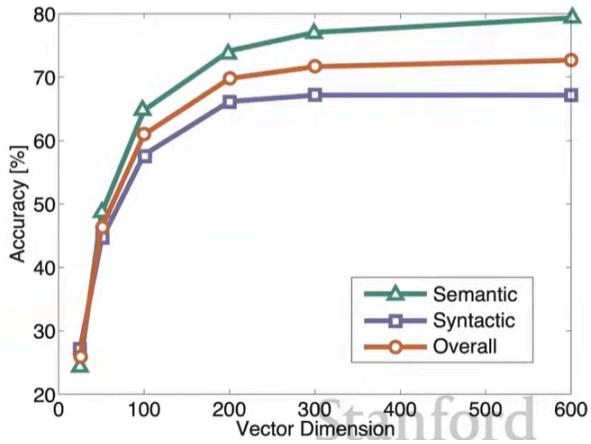
- **Intrinsic way of evaluation**: (局部子任务做起)
  - evaluate directly on a specific subtask
  - helps to understand the system
  - fast
  - not clear if there is correlation established between the real task
- **Extrinsic way of evaluation**:（全局的任务）
  - evaluate on a specific task (whole)
  - long time to evaluate
  - Unclear if the subsystem has problems

6. Results

- More data helps
- Wikipedia is better than news text!



- Dimensionality
- Good dimension is ~300



Word vector distances and their correlation with human judgments

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	72.7	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	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
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

On extrinsic task - In a real NLP task

Extrinsic evaluation of word vectors: All subsequent NLP tasks in this class. More examples soon.

One example where good word vectors should help directly: **named entity recognition**: identifying references to a person, organization or location

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

- **Extra:** words with multi meaning are made into multiple word vectors.
- and when computing they are made into 1 single vector through linear combination of the multi meaning vectors.
- however they could be seperated again due to the high dimensionality of the vectors and how sparse the vectors are.