COMP 5300 / COMP 4600
Deep Learning for NLP

Lecture

Plan for next week

- Homework #2
  - Will be posted tonight
  - Due next Tue 1/30/24 at midnight
  - Use colab to do the homework
  - Colab tutorial: https://neptune.ai/blog/how-to-use-google-colab-for-deep-learning-complete-tutorial
- Homework #1
  - Was due last night at midnight
  - Can submit by EOD today or tomorrow, 10% off each day
- Today's lecture
  - Text classification and lexical embeddings

Lecture outline

- Lexical embeddings (= word vectors)
  - – count-based (sparse, dense)
  - – prediction-based (neural embeddings)
  - – evaluating lexical embeddings
- Tokenization
- Text classification

Lexical embeddings

vector space representation for word-level semantics

Naive Representation of Text Documents

One-hot representation for each word:

[0 0 0 0 1 0 0 0 0 0]

– dimensionality is |V|, size of your chosen vocabulary

Compare two documents, e.g. for classification

[0 1 0 0 1 1 0 0 1 0]

[0 0 0 0 1 0 0 0 1 0]

– 1’s in positions corresponding to the words present in the document

– could be (normalized) counts instead.

Use e.g. cosine or set-membership similarity measures to compute “distance” between two documents

Count-based Vectors

Can represent a document as a “bag of words”

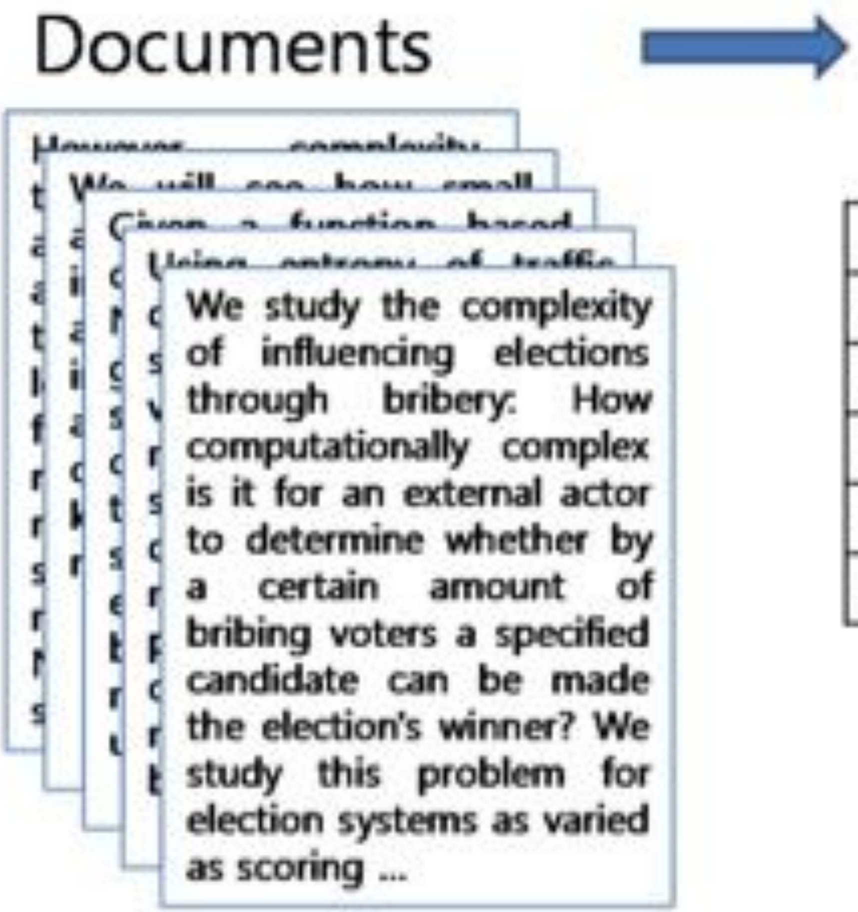
Turn your text into a vector of word counts

For example, a movie review:

> An unpleasant, humorless slog through the muck of low-budget January horror fodder that is neither frightening or particularly entertaining, blandly ambling from tired jump-scare to tired jump-scare."

$\langle$unpleasant: 1, the: 23, dog: 0, tired:3, ...$\rangle$

Bag-of-Words (BOW) Representation



Vector-space representation

	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Term-document matrix

SOURCE: http://mlg.postech.ac.kr/research/nmf

Bag-of-Words (BOW) Representation

Counts could be normalized by frequency

Probability (divide by corpus size)

$$\text{count}(w1)/N$$

Conditional probability (divide by co-occurrence count)

$$P(w1)/P(w1, w2)$$

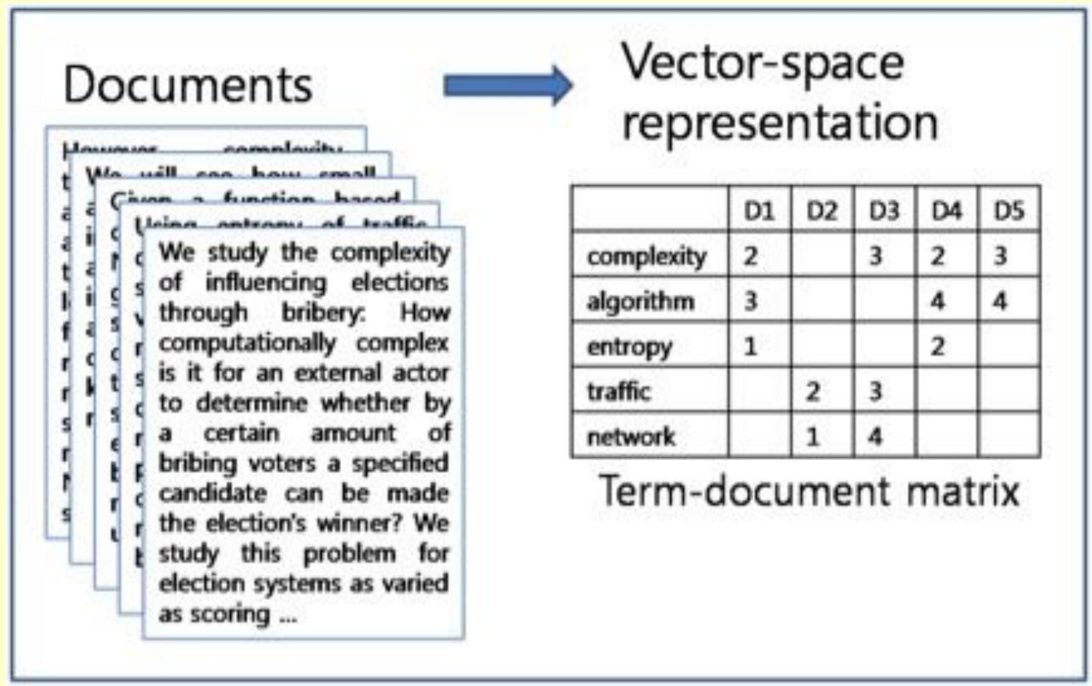
Pointwise mutual information

$$\log P(w1, w2)/P(w1)P(w2)$$

TF-IDF

$$\text{count}(w1)/\text{doc\_count}(w1)$$

BOW representation works well for tasks invariant to word order, such as text classification

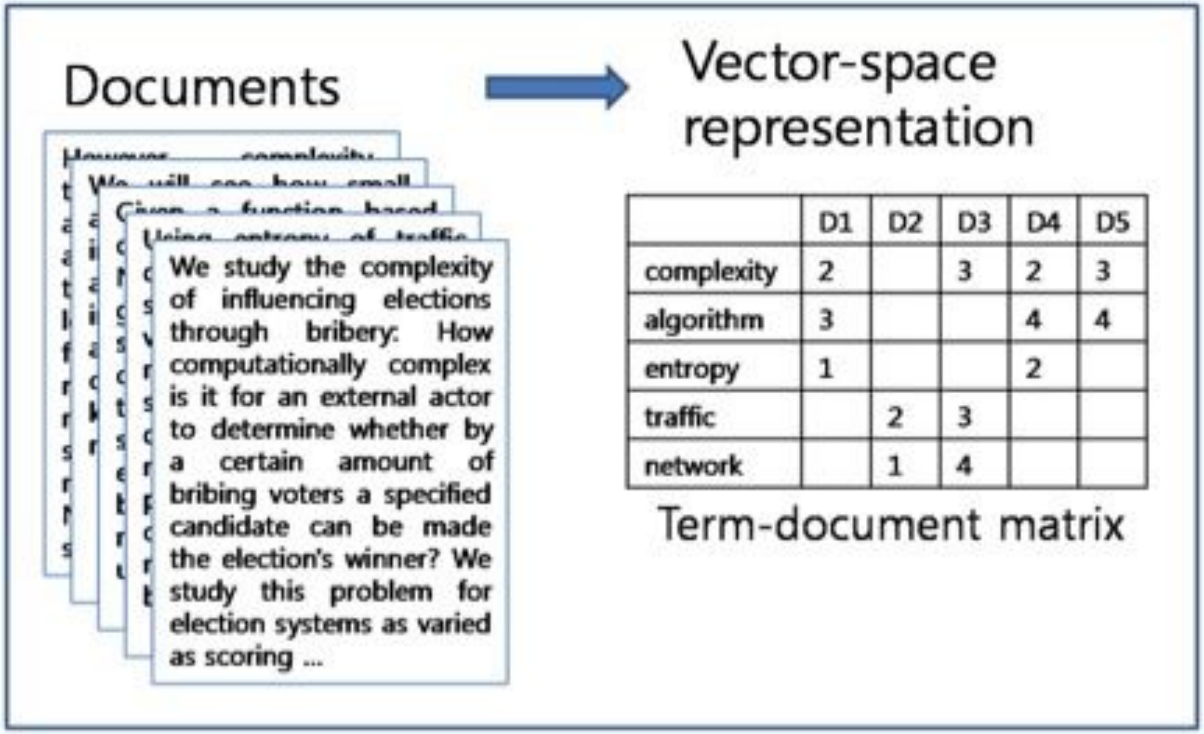


	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

TF-IDF Weighting

TF-IDF (x) = term frequency in document / # of documents in which the term occurs.

- IDF (“inverse document frequency”) penalizes counts (= reduces weights) of words that occur in many documents
- A document could be a sentence, a paragraph, a chapter, a movie/product review, etc.

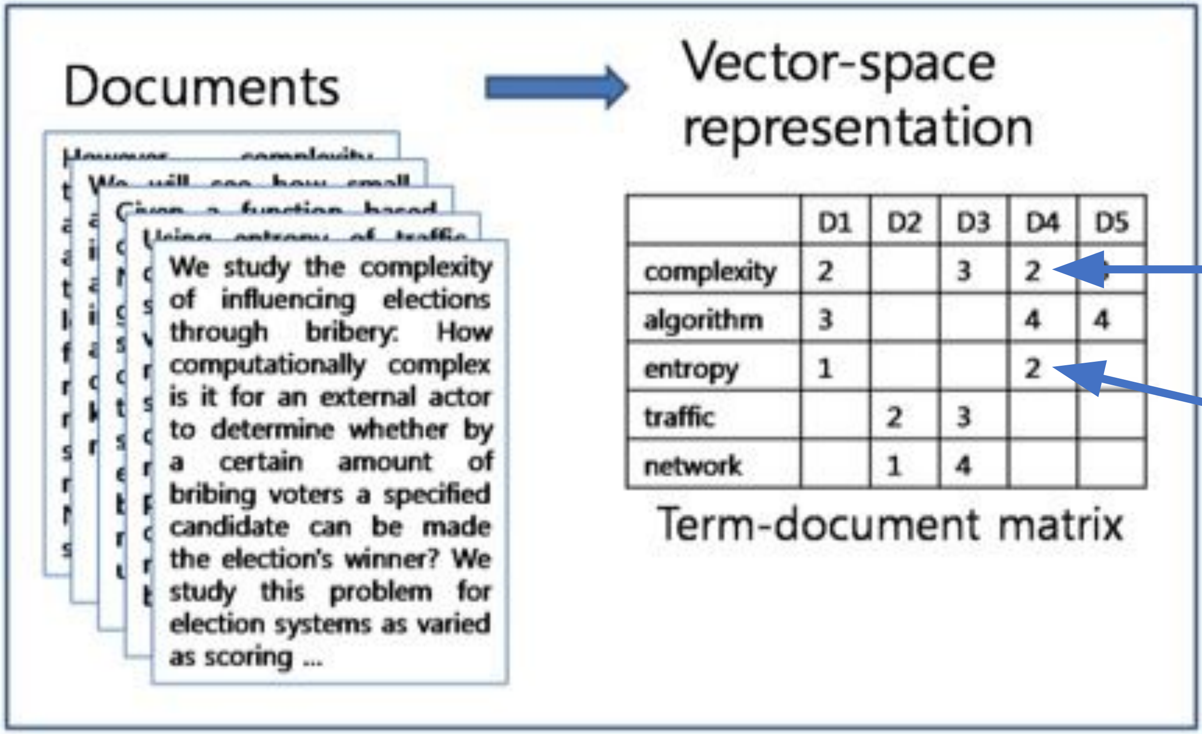


	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

TF-IDF Weighting

TF-IDF (x) = term frequency in document / # of documents in which the term occurs.

- IDF (“inverse document frequency”) penalizes counts (= reduces weights) of words that occur in many documents
- A document could be a sentence, a paragraph, a chapter, a movie/product review, etc.



	D1	D2	D3	D4	D5
complexity	2		3	2	
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

2/4 = 0.5

2/2 = 1

D4 is more about “entropy” than about “complexity”

Similarity Measures

$$Dice(A,B) = \frac{|A \cap B|}{\frac{1}{2}(|A|+|B|)}; \quad Dice^{\dagger}(\vec{X},\vec{Y}) = \frac{\sum_i \min(x_i,y_i)}{\frac{1}{2}(\sum_i x_i + \sum_i y_i)}$$

$$Jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|}; \quad Jaccard^{\dagger}(\vec{X},\vec{Y}) = \frac{\sum_i \min(x_i,y_i)}{\sum_i \max(x_i,y_i)}$$

$$\cos(\bar{X},\bar{Y}) = \frac{\vec{X} \cdot \vec{Y}}{|\vec{X}||\vec{Y}|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

$$Euclidean\text{-}Distance(\vec{X},\vec{Y}) = |\vec{X} - \vec{Y}| = \sqrt{\sum_i (x_i - y_i)^2}$$

$$L_1 \ norm = \sum_i |x_i - y_i| = 2\left(1 - \sum_i \min(x_i, y_i)\right)$$

$$D(p||q) = \sum_i p_i \log \frac{p_i}{q_i}$$

$$JS(p||q) = \frac{1}{2}\left[D(p||\tfrac{p+q}{2}) + D(q||\tfrac{p+q}{2})\right]$$

$$\alpha\text{-}skew(p,q) = D(p||\alpha \cdot q + (1-\alpha) \cdot p)$$

Text Classification

Text Classification Examples

- Identifying topics
  - sports / politics / finance / …
- Subject headings for books / articles
  - MeSH headings: Drug Therapy / Embryology / …
- Sentiment analysis
- Authorship identification
- Spam detection
- etc.

Text Classification

Input:

a document $d \in D$

a fixed set of classes $C = \{c_1 \ldots c_k\}$

Output:

a predicted class $c \in C$

a trained model → a learned classification function $f : D \rightarrow C$

Workflow

- Create a labeled corpus
  select texts, pick categories, assign labels
- Split it into test and training segments

Supervised Task

- Choose a representation, i.e. how each text will be represented
- Choose a classifier model
  Naive Bayes, Decision Tree, Logistic Regression, SVM, Neural Network

- Train the model on the training data
- Test model performance on the test data
- If satisfactory, use the model to process unseen text

Text Classification Models

Could be any of your favorite classifier models:

Logistic regression (LR), support vector machine (SVM), multi-layer perceptron (MLP), deep neural network (the last hidden state is used as input to a softmax layer that does classification), etc.

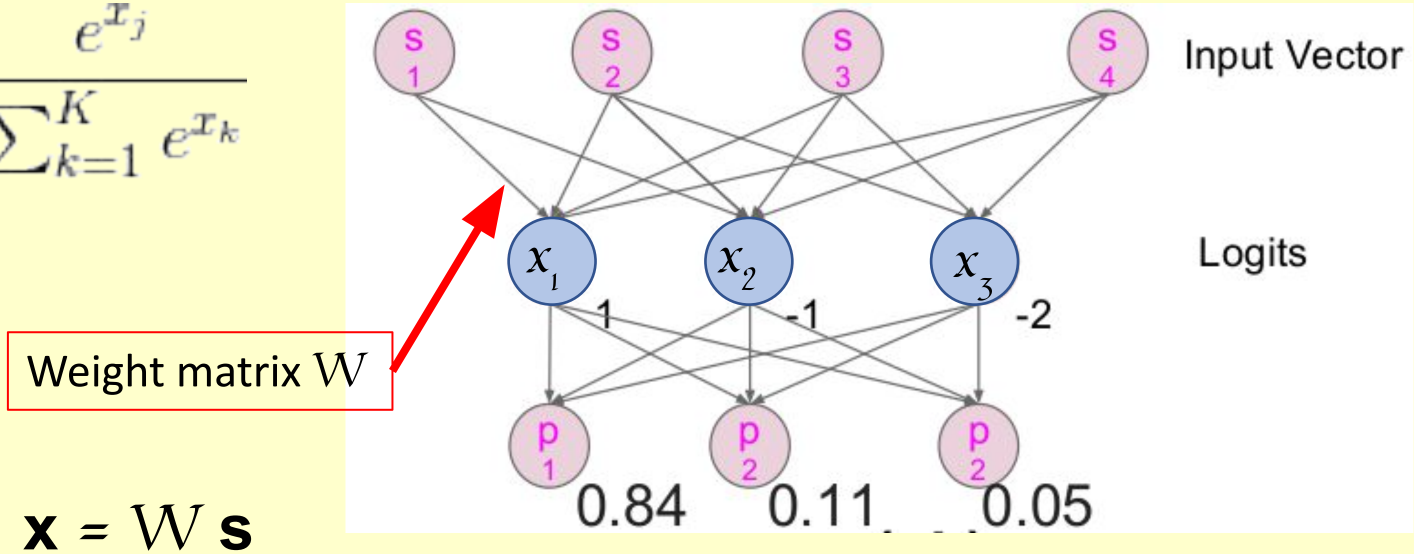
If text classification done by applying **logistic regression** (softmax) to vector representations of input text

- Project into an softmax layer of dimensionality |C|
- Apply softmax to find highest-probability category

Multiclass Logistic Regression

Vector representation of the document is used as input to a softmax layer that does classification, assigning a probability to each label y_i:

$$P(y=j|x) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$



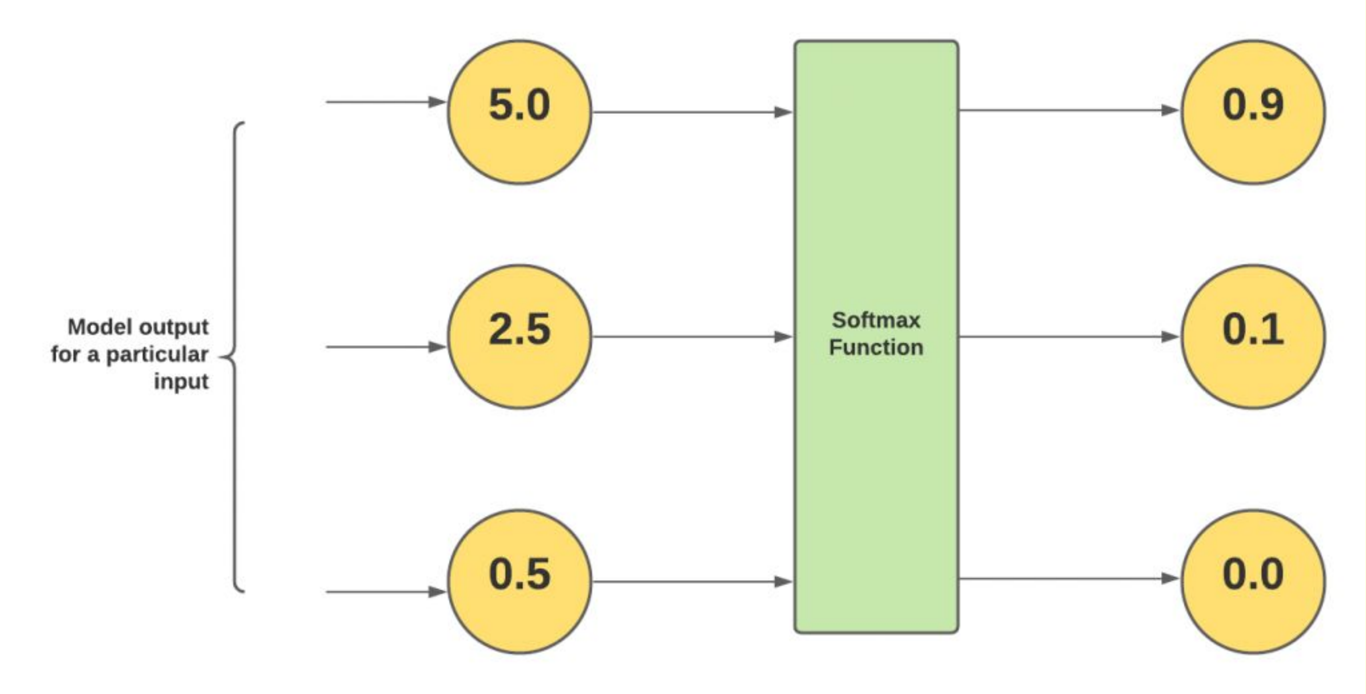
$\mathbf{x} = W\,\mathbf{s}$

$x_i = \mathbf{w}_j\,\mathbf{s}$

$x_i = \sum s_j w_{ij}$

Softmax operation

$$P(y=j|x) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$



SOURCE: https://vitalflux.com/what-softmax-function-why-needed-machine-learning/

Sample Classification Tasks

Sentiment

"This movie was actually neither that funny, nor super witty."

Labels: positive, negative, neutral

Entailment

"The maniac killed his victim" – "His victim died"

"The maniac killed his victim" – "His victim survived"

Labels: Entailment, Contradiction, Neutral

What about words?

One-hot representation for each word:

[0 0 0 0 1 0 0 0 0 0]

– dimensionality is |V|, size of your chosen vocabulary

Can we do better?

Representing words

One-hot representation for each word:

[0 0 0 0 1 0 0 0 0 0]

– dimensionality is |V|, size of your chosen vocabulary

Can we do better?

Can we make it so that similar words have similar representations?

Distributional hypothesis

- Meaning as an invariant of similar words (Mel’chuk)
- Similar words are used in similar contexts.
- This is known as the “distributional hypothesis” (Harris, 1985), or the “strong contextual hypothesis” (Miller and Charles, 1991), and related to the much-quoted remark by Firth (1957)

“You shall know a word by the company it keeps”

Meaning from context

Consider occurrence contexts of an unknown word, tezgüino

C1: A bottle of tezgüino is on the table

C2: Everybody likes tezgüino

C3: Don’t have tezgüino before you drive.

C4: We make tezgüino out of corn.

Distributional statistics for tezgüino:

	C1	C2	C3	C4	...
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
choices	0	1	0	0	
wine	1	1	1	0	

What about words?

Can we make it so that similar words have similar representations?

- Represent each word as a collection of contexts in which it has occurred in a corpus of texts

	C1	C2	C3	C4	...
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
choices	0	1	0	0	
wine	1	1	1	0	

Concordance for “showed”

KWIC concordance (Key Word In Context)

1	could find a target. The librarian	“showed	off” – running hither and thither w
2	elights in. The young lady teachers	“showed	off” – bending sweetly over pupils
3	ingly. The young gentlemen teachers	“showed	off” with small scoldings and other
4	seeming vexation). The little girls	“showed	off” in various ways, and the littl
5	n various ways, and the little boys	“showed	off” with such diligence that the a
6	t genuwyne?” Tom lifted his lip and	showed	the vacancy. “Well, all right,” sai
7	is little finger for a pen. Then he	showed	Huckleberry how to make an H and an
8	ow’s face was haggard, and his eyes	showed	the fear that was upon him. When he
9	not overlook the fact that Tom even	showed	a marked aversion to these inquests
10	own. Two or three glimmering lights	showed	where it lay, peacefully sleeping,
11	ird flash turned night into day and	showed	every little grass-blade, separate
12	that grew about their feet. And it	showed	three white, startled faces, too. A
13	he first thing his aunt said to him	showed	him that he had brought his sorrows
14	p from her lethargy of distress and	showed	good interest in the proceedings. S
15	ent a new burst of grief from Becky	showed	Tom that the thing in his mind had
16	shudder quiver all through him. He	showed	Huck the fragment of candle-wick pe

Concordance for “showed”

KWIC concordance (Key Word In Context)

1	could find a target. The librarian	“showed	off” - running hither and thither w
2	elights in. The young lady teachers	“showed	off” - bending sweetly over pupils
3	ingly. The young gentlemen teachers	“showed	off” with small scoldings and other
4	seeming vexation). The little girls	“showed	off” in various ways, and the littl
5	n various ways, and the little boys	“showed	off” with such diligence that the a
6	t genuwyne?” Tom lifted his lip and	showed	the vacancy. “Well, all right,” sai
7	is little finger for a pen. Then he	showed	Huckleberry how to make an H and an
8	ow’s face was haggard, and his eyes	showed	the fear that was upon him. When he
9	not overlook the fact that Tom even	showed	a marked aversion to these inquests
10	own. Two or three glimmering lights	showed	where it lay, peacefully sleeping,
11	ird flash turned night into day and	showed	every little grass-blade, separate
12	that grew about their feet. And it	showed	three white, startled faces, too. A
13	he first thing his aunt said to him	showed	him that he had brought his sorrows
14	p from her lethargy of distress and	showed	good interest in the proceedings. S
15	ent a new burst of grief from Becky	showed	Tom that the thing in his mind had
16	shudder quiver all through him. He	showed	Huck the fragment of candle-wick pe

Concordance for “showed”

KWIC concordance (Key Word In Context)

1	could find a target. The librarian	“showed	off” - running hither and thither w
2	elights in. The young lady teachers	“showed	off” - bending sweetly over pupils
3	ingly. The young gentlemen teachers	“showed	off” with small scoldings and other
4	seeming vexation). The little girls	“showed	off” in various ways, and the littl
5	n various ways, and the little boys	“showed	off” with such diligence that the a
6	t genuwyne?” Tom lifted his lip and	showed	the vacancy. “Well, all right,” sai
7	is little finger for a pen. Then he	showed	Huckleberry how to make an H and an
8	ow’s face was haggard, and his eyes	showed	the fear that was upon him. When he
9	not overlook the fact that Tom even	showed	a marked aversion to these inquests
10	own. Two or three glimmering lights	showed	where it lay, peacefully sleeping,
11	ird flash turned night into day and	showed	every little grass-blade, separate
12	that grew about their feet. And it	showed	three white, startled faces, too. A
13	he first thing his aunt said to him	showed	him that he had brought his sorrows
14	p from her lethargy of distress and	showed	good interest in the proceedings. S
15	ent a new burst of grief from Becky	showed	Tom that the thing in his mind had
16	shudder quiver all through him. He	showed	Huck the fragment of candle-wick pe

Context definition

Example: “was haggard and his eyes **showed** the fear that was upon”

Window BOW context (+5/-5):

{was, haggard, and, his, eyes, the, fear, that, was, upon}

Structured context (+5/-5):

{(was, −5), (haggard, −4), (and, -3), (his, -2), (eyes, -1), (the, +1), (fear, +2), (that, +3), (was, +4), (upon, +5)}

Dependency context:

{(eyes, NSUBJ), (fear, DOBJ), … }

develop **bnc freq = 23388**

object	14323	5.5	subject	4512	3.5	modifier	3153	2.4	and/or	960	0.5
skill	432	38.26	compact	10	19.46	further	192	46.65	grow	45	27.52
technique	219	30.08	patient	58	18.43	rapidly	97	40.95	implement	26	26.95
strategy	164	27.61	situation	44	16.53	jointly	51	39.89	maintain	39	25.87
model	195	24.98	capitalism	14	16.51	fully	116	38.68	refine	12	24.94
idea	223	24.14	embryo	11	16.43	originally	71	37.65	expand	21	24.78
method	168	23.77	scientist	22	16.11	well	132	31.64	market	15	24.29
theory	158	23.45	Freud	10	15.47	independently	28	30.91	test	19	21.03
system	439	23.39	relationship	42	15.2	subsequently	39	30.3	sustain	14	20.7
understanding	103	23.24	Scotvec	8	15.17	specially	31	29.26	research	10	20.11
technology	139	22.69	Ici	9	14.72	later	59	29.21	explore	15	19.84
expertise	50	21.69	Aea	6	14.33	also	184	28.96	extend	20	19.28
methodology	29	21.33	Ibm	14	14.19	highly	59	28.58	improve	17	17.74
concept	89	21.17	company	61	12.94	quickly	51	27.9	promote	14	17.42
relationship	144	20.67	friendship	10	12.38	gradually	31	27.33	train	12	16.86

Bag-of-Words (BOW) Representation

Counts could be normalized:

Count (w) = how many times a term w occurs in our corpus; normalize by corpus size N to get $p(w)$.

Conditional probability (divide by co-occurrence count)

Conditional_Probability$(w|c)$ = divide the number of times our target word w coccurs with the context word c in our corpus, i.e. *Count*(w, c) by the total occurrence count of c, *Count*(c).

TF-IDF

TF-IDF (w) = term frequency in document / # of documents in which the term occurs.

Pointwise mutual information

$$\operatorname{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}$$

develop **bnc freq = 23388**

object	14323	5.5	subject	4512	3.5	modifier	3153	2.4	and/or	960	0.5
skill	432	38.26	compact	10	19.46	further	192	46.65	grow	45	27.52
technique	219	30.08	patient	58	18.43	rapidly	97	40.95	implement	26	26.95
strategy	164	27.61	situation	44	16.53	jointly	51	39.89	maintain	39	25.87
model	195	24.98	capitalism	14	16.51	fully	116	38.68	refine	12	24.94
idea	223	24.14	embryo	11	16.43	originally	71	37.65	expand	21	24.78
method	168	23.77	scientist	22	16.11	well	132	31.64	market	15	24.29
theory	158	23.45	Freud	10	15.47	independently	28	30.91	test	19	21.03
system	439	23.39	relationship	42	15.2	subsequently	39	30.3	sustain	14	20.7
understanding	103	23.24	Scotvec	8	15.17	specially	31	29.26	research	10	20.11
technology	139	22.69	Ici	9	14.72	later	59	29.21	explore	15	19.84
expertise	50	21.69	Aea	6	14.33	also	184	28.96	extend	20	19.28
methodology	29	21.33	Ibm	14	14.19	highly	59	28.58	improve	17	17.74
concept	89	21.17	company	61	12.94	quickly	51	27.9	promote	14	17.42
relationship	144	20.67	friendship	10	12.38	gradually	31	27.33	train	12	16.86

Word meaning represented by word embeddings

Count-based frequency vectors

- Dimensionality of vocabulary |V|, 100K
- Sparse

Prediction-based continuous "dense" vectors

- Lower dimensionality (commonly 200-1000)
- Continuous

Dense word embeddings

Dense Word Embeddings

1. Reduce dimensionality of count-based representation
   - Principal Component Analysis (PCA), singular value decoposition (SVD) (also “Latent Semantic Analysis”, LSA).
2. Learn embeddings as parameters in a learning task, where a cost function is tied to the context
   - Different approximations to the log likelihood of the full corpus

Singular Value Decomposition

Mathematical apparatus:

- $rank(A)$
  - number of linearly independent columns in $A_{m \times n}$: $R^n \rightarrow R^m$
  - linear transform of rank r maps the basis vectors of the pre-image into r linearly independent basis vectors of the image
- Singular Value Decomposition
  - Matrix A can be represented as
    - » $A = U \Sigma V^T$ where

      columns of U and V are left and right eigenvectors of $A A^T$

      U and V orthogonal ($VV^T=I$) , and Σ is a diagonal matrix:

      $\Sigma = \mathbf{diag}(\sigma_1, \ldots, \sigma_n)$, where σ_i are nonnegative square roots of the r eigenvalues of $A A^T$, SVD produces a k-rank approximation Â to matrix A minimizing the "distance" between the two matrices in the form of Frobenius norm (aka 2-norm, Euclidean norm) is minimized:

T

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{i\,j}|^2}$$

Minimize the objective:

Min $\|A - U\,\Sigma\,V^T\|_F$

As linear regression can be interpreted as collapsing a two-dimensional space onto a one-dimensional line, SVD can be thought of as projecting an n-dimensional space onto a k-dimensional space where n >> k.

Folding new count-based vectors into the reduced space:

$A = U \sum V^T$

$\rightarrow U^T A = U^T U \sum V^T$

$\rightarrow U^T A = \sum V^T$

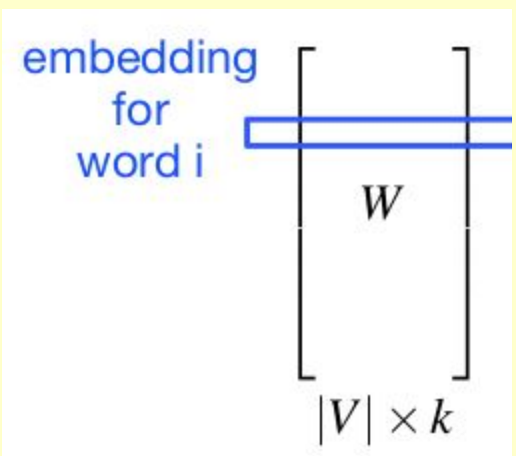
U, V are orthonormal (column vectors are unit length and orthogonal, so $U^T U = I$)

Project a new count-based vector into k-dimensional space

$a_k = U^T a_{new}$

U = (W)ords, V = (C)ontexts

$$\underset{A}{\underset{|V| \times |V|}{\begin{bmatrix} \\ \\ X \\ \\ \\ \end{bmatrix}}} = \underset{U}{\underset{|V| \times k}{\begin{bmatrix} \\ \\ W \\ \\ \\ \end{bmatrix}}} \underset{\Sigma}{\underset{k \times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix}}} \underset{V}{\underset{k \times |V|}{\begin{bmatrix} C \end{bmatrix}}}$$



$$a_k = W^T a_{new}$$

Slide credit: Dan Jurafsky

Vector Space Embeddings

"Count-based" vectors

- Variously normalized word count-based representations
- Vocabulary-size dimensionality reduced via PCA, SVD, etc.

Learned vectors ("prediction-based")

- Learn embeddings as parameters in a learning task
- Where the cost function e.g. maximizes the probability of your training text

Neural word embeddings

Embedding Models

Learning count based vectors produces an **embedding matrix** in $R^{|vocab| \times |context|}$:

$$\mathbf{E} = \begin{array}{c} \\ \text{kitten} \\ \text{cat} \\ \text{dog} \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccc} \text{bit} & \text{cute} & \text{furry} & \dots \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & \dots \\ 0 & 1 & 1 & \dots \\ 1 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \end{array}$$

Rows are word vectors, so we can retrieve them with **one hot vectors** in $\{0,1\}^{|vocab|}$:

$$onehot_{cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \mathbf{cat} = onehot_{cat}^{\top}\mathbf{E}$$

Symbols = unique vectors. Representation = embedding symbols with **E**.

Slide credit: EdGrefenstette

word2vec (Mikolov et al 2013)

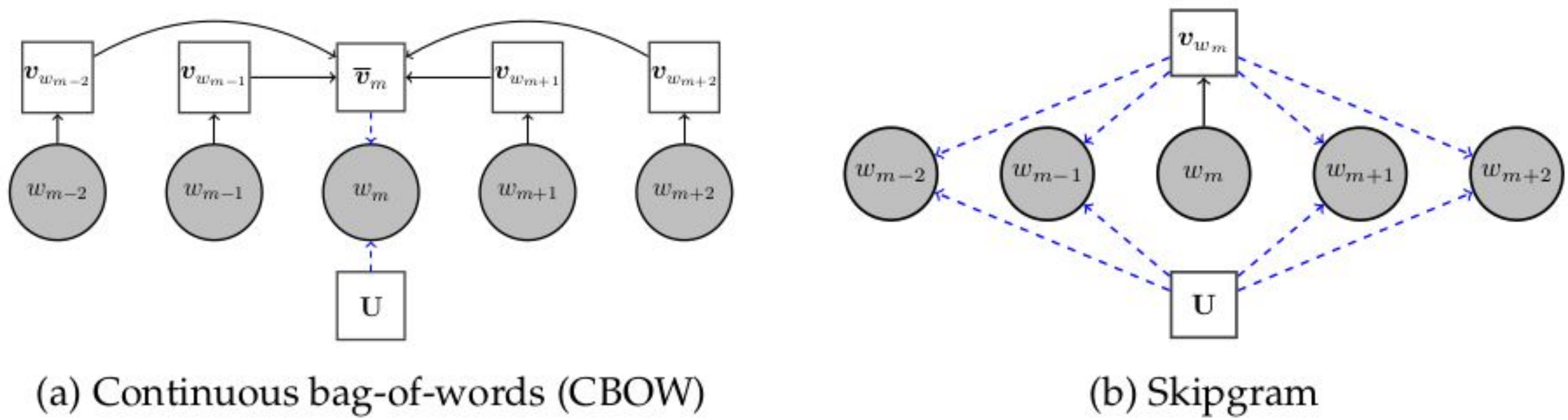


Figure 13.3: The CBOW and skipgram variants of WORD2VEC. The parameter $\mathbf{U}$ is the matrix of word embeddings, and each $\boldsymbol{v}_m$ is the context embedding for word w_m.

word2vec (Mikolov et al 2013)

Let vector u_i = the k-dimensional embedding for word i

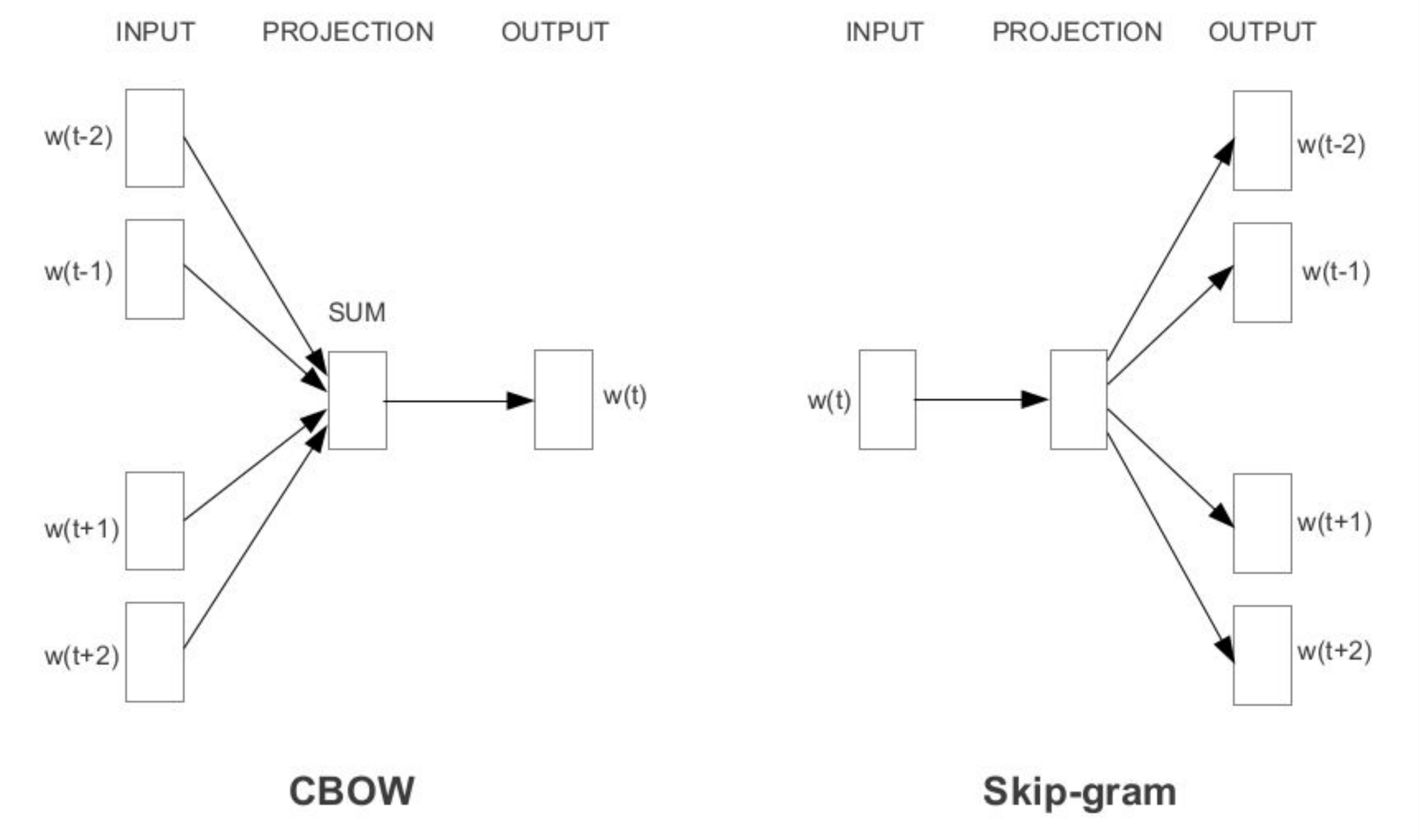
Let v_j = k-dimensional embedding for context j.

The inner product $u_i \cdot v_j$ represents the compatibility between word i and context j.

By incorporating this inner product into an approximation to the log-likelihood of a corpus, it is possible to estimate both parameters by backpropagation.

$$\begin{aligned} P(w_1^t) &= P(w_1)P(w_2|w_1)P(w_3)|w_1w_2)\ldots P(w_t|w_1^{t-1}) \\ &= \prod_{i=1}^{t} P(w_i|w_1^{i-1}) \end{aligned}$$

word2vec includes two such approximations: continuous bag-of-words (CBOW) and skip-gram.



INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW
INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram

Word2vec CBOW

BOW b/c order of words doesn't matter

h determines window size

Local context is computed as an average of embeddings for words in the immediate neighborhood of m:

$$\mathrm{m}-\mathrm{h}, \mathrm{m}-\mathrm{h}+1, \ldots, \mathrm{m}+\mathrm{h}-1, \mathrm{m}+\mathrm{h}$$

$$\tilde{v}_m = \frac{1}{2h}\sum_{n=1}^{h}(v_{m+n}+v_{m-n})$$

Word2vec CBOW

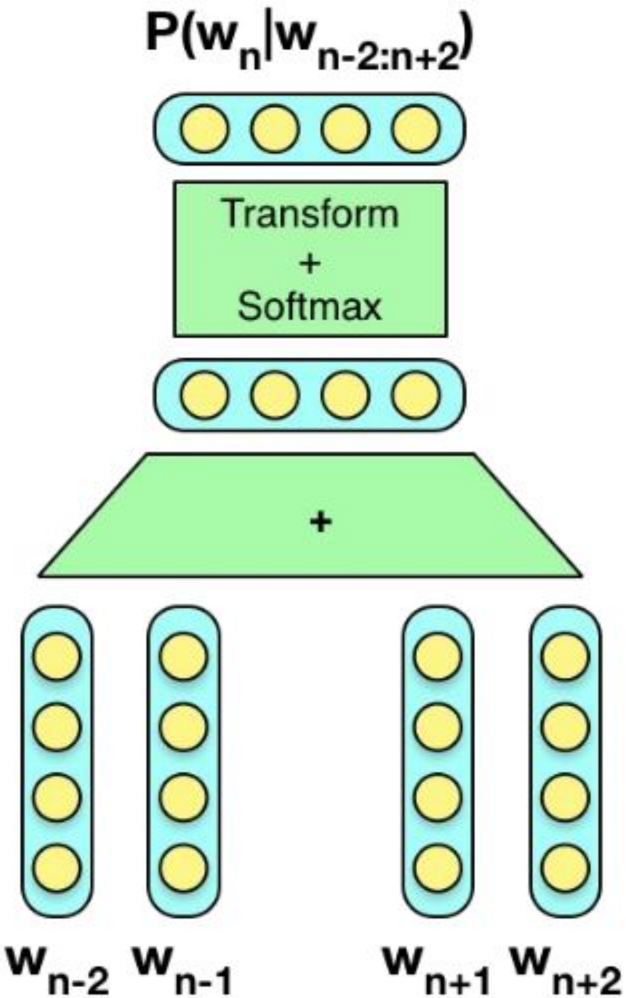
Words are predicted from context

Optimizes approximation to corpus log likelihood

$$
\begin{aligned}
\log \mathrm{p}(\boldsymbol{w}) &\approx \sum_{m=1}^{M} \log \mathrm{p}(w_m \mid w_{m-h}, w_{m-h+1}, \ldots, w_{m+h-1}, w_{m+h}) \\
&= \sum_{m=1}^{M} \log \frac{\exp\left(\boldsymbol{u}_{w_m} \cdot \overline{\boldsymbol{v}}_m\right)}{\sum_{j=1}^{V} \exp_{j}\left(\boldsymbol{u}_i \cdot \overline{\boldsymbol{v}}_m\right)} \\
&= \sum_{m=1}^{M} \boldsymbol{u}_{w_m} \cdot \overline{\boldsymbol{v}}_m - \log \sum_{j=1}^{V} \exp_{j}\left(\boldsymbol{u}_i \cdot \overline{\boldsymbol{v}}_m\right).
\end{aligned}
$$

M is the size of the corpus

word2vec – continuous bag of words (CBOW)



Embed context words. Add them.

Project back to vocabulary size. Softmax.

$$softmax(\mathbf{l})_i = \frac{e^{l_i}}{\sum_j e^{l_j}}$$

$$P(t_i|context(t_i)) = softmax\left(\sum_{t_j \in context(t_i)} onehot_{t_j}^{\top}\ \mathbf{E}\ W_v\right)$$

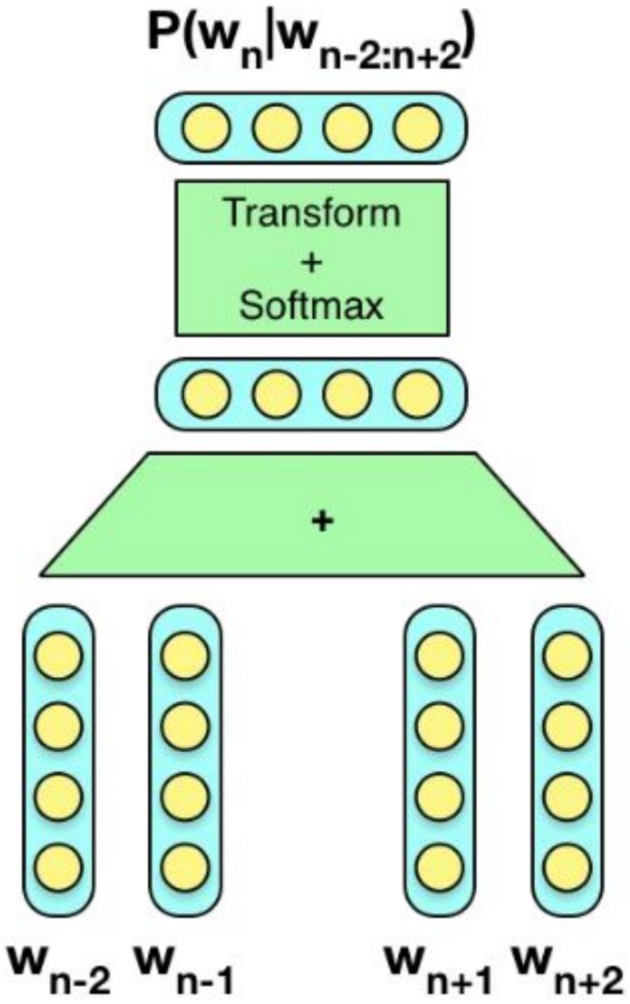
$$= softmax\left(\left(\sum_{t_j \in context(t_i)} onchot_{t_j}^{\top}\ \mathbf{E}\right) W_v\right)$$

Minimize Negative Log Likelihood:

$$L_{data} = -\sum_{t_i \in data} logP(t_i|context(t_i))$$

Slide credit: Ed Grefenstette

word2vec – continuous bag of words (CBOW)



All linear, so very fast. Basically a cheap way of applying one matrix to all inputs.

Historically, negative sampling used instead of expensive softmax.

NLL minimisation is more stable and is fast enough today.

Variants: position specific matrix per input (Ling et al. 2015).

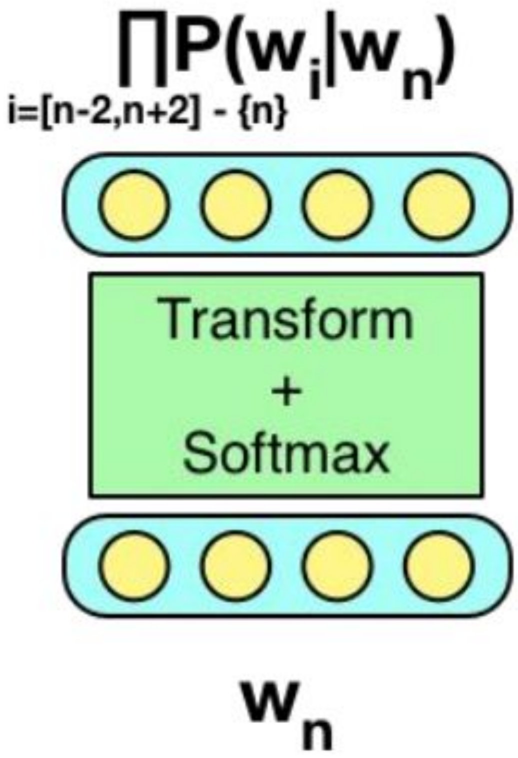
Slide credit: Ed Grefenstette

Word2vec Skip-Gram

Contexts are predicted from words

$$
\begin{aligned}
\log \mathrm{p}(\boldsymbol{w}) &\approx \sum_{m=1}^{M} \sum_{n=1}^{h_m} \log \mathrm{p}(w_{m-n} \mid w_m) + \log \mathrm{p}(w_{m+n} \mid w_m) \\
&= \sum_{m=1}^{M} \sum_{n=1}^{h_m} \log \frac{\exp(\boldsymbol{u}_{w_{m-n}} \cdot \boldsymbol{v}_{w_m})}{\sum_{j=1}^{V} \exp(\boldsymbol{u}_i \cdot \boldsymbol{v}_{w_m})} + \log \frac{\exp(\boldsymbol{u}_{w_{m+n}} \cdot \boldsymbol{v}_{w_m})}{\sum_{j=1}^{V} \exp(\boldsymbol{u}_i \cdot \boldsymbol{v}_{w_m})} \\
&= \sum_{m=1}^{M} \sum_{n=1}^{h_m} \boldsymbol{u}_{w_{m-n}} \cdot \boldsymbol{v}_{w_m} + \boldsymbol{u}_{w_{m+n}} \cdot \boldsymbol{v}_{w_m} - 2 \log \sum_{j=1}^{V} \exp\left(\boldsymbol{u}_i \cdot \boldsymbol{v}_{w_m}\right).
\end{aligned}
$$

word2vec – Skip-gram



Target word predicts context words.

Embed target word.

Project into vocabulary. Softmax.

$$P(t_j|t_i) = softmax(onehot_{t_i}^{\top}\ \mathbf{E}\ W_v)$$

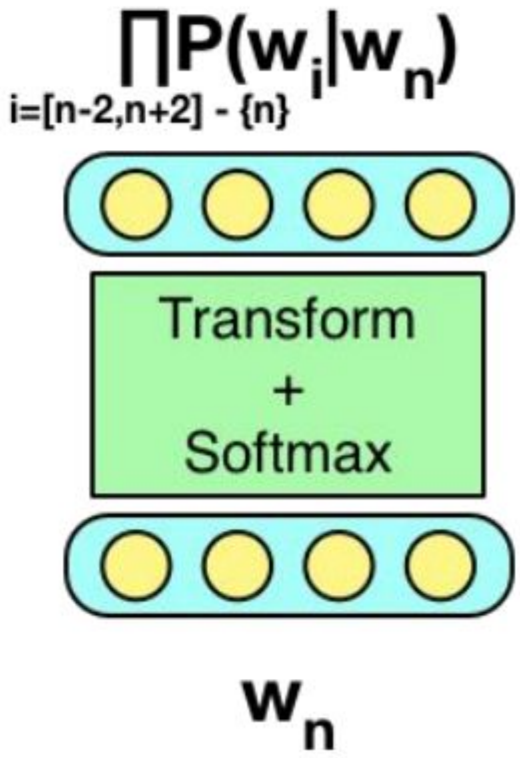
Learn to estimate likelihood of context words.

$$-logP(context(t_i)|t_i) = -log \prod_{t_j \in context(t_i)} P(t_j|t_i)$$

$$- \sum_{t_j \in context(t_i)} logP(t_j|t_i)$$

Slides from Ed Grefenstette

word2vec – Skip-gram



Fast: One embedding versus |C| embeddings.

Just read off probabilities from softmax.

– probabilities of the context words, that is

Similar variants to CBoW possible: position specific projections.

Trade off between efficiency and more structured notion of context.

Time Complexity

CBOW and skipgram have a linear time complexity in the size of the word and context representations.

But! they compute a normalized probability over word tokens – a naïve implementation requires summing over the entire vocabulary.

The time complexity of this sum is $O(V \times K)$ for k-dimensional embeddings – which dominates all other computational costs.

One solution is negative **negative sampling:**

Negative sampling is an approximation that eliminates the dependence on vocabulary size!

Negative sampling

Likelihood-based methods are expensive b/c each probability must be normalized over the vocabulary

These probabilities are based on similarity scores between words and contexts for each word in each context.

Can we define an alternative objective based on the same word-context co-occurrence scores $\psi(w, c)$?

Negative Sampling

Seek word embeddings that *maximize the difference* between the score for the word observed in context, and the scores for several randomly selected "negative samples" (words that did not occur in that context):

- where $\psi(i, j)$ is the score for word i in context j, W_{neg} is the set of negative samples.

1 – σ (x) = σ(–x) i.e. probability that x did not occur (cf. σ(x) graph)

The objective is to maximize the sum over the corpus:

$$\sum \psi(w_m , c_m)$$

i.e. log product probability of each encountered context in corpus by probability that negative samples for that context didn't occur

Negative Sampling

The set of negative samples W_{neg} is obtained by sampling from a unigram language model.

Unigram language model constructed by exponentiating the empirical word probabilities, setting $\hat{p}(i) \propto (count(i))^{(3/4)}$.

This has the effect of redistributing probability mass from common to rare words.

The number of negative samples increases the time complexity of training by a constant factor.

5-20 negative samples works for small training sets, and that two to five samples suffice for larger corpora.

Word Embeddings as Matrix factorization

The negative sampling objective is linked to the matrix factorization objective employed in latent semantic analysis.

For a matrix of word-context pairs in which all counts are non-zero, negative sampling is equivalent to factorization of the matrix M,

where $M_{ij} = PMI(i, j) - \log k$

- each cell in the matrix is equal to the pointwise mutual information of the word and context, shifted by log k, with k equal to the number of negative samples (Levy and Goldberg, 2014)
- k is the number of negative samples in SGNS

Word embeddings are obtained by factoring this matrix with truncated singular value decomposition.

GloVe Embeddings (Stanford)

Another matrix factorization approach. The matrix to be factored is constructed from log co-occurrence counts, $M_{i\,j}$ = count(i, j).

Weighted least squares is the objective:

$$\min_{\boldsymbol{u},\boldsymbol{v},b,\tilde{b}} \quad \sum_{j=1}^{V} \sum_{j \in \mathcal{C}} f(M_{ij}) \left(\widehat{\log M_{ij}} - \log M_{ij} \right)^2$$

$$s.t. \quad \widehat{\log M_{ij}} = \boldsymbol{u}_i \cdot \boldsymbol{v}_j + b_i + \tilde{b}_j,$$

where b_i and b_j are biases for word i and context j, which are estimated jointly with the word embedding **u** and context embedding **v**.

The weighting function $f(M_{i\,j})$ is set to 0 at $M_{ij} = 0$

To avoid overweighing frequent context-word pairs:

$f(M_{ij}) = (M_{ij} / \text{threshold})^{3/4}$ if $M_{ij} <$ threshold, 1 otherwise

Evaluating lexical embeddings

(vector space representation for word-level semantics)

How good are these representations?

Evaluation of word embeddings

- WordSim-353 (Finkelstein et al. 2003)

  contains two sets of English word pairs along with human-assigned similarity judgements
- SimLex-999 (Hill et al. 2016, but has been around since 2014)

Pair	Simlex-999 rating	WordSim-353 rating
coast - shore	9.00	9.10
clothes - closet	1.96	8.00

- Word analogy task (Mikolov et al. 2013), queen = king - man + woman.
- Embedding visualization (nearest neighbors, T-SNE projection)

T-SNE – dimensionality reduction technique

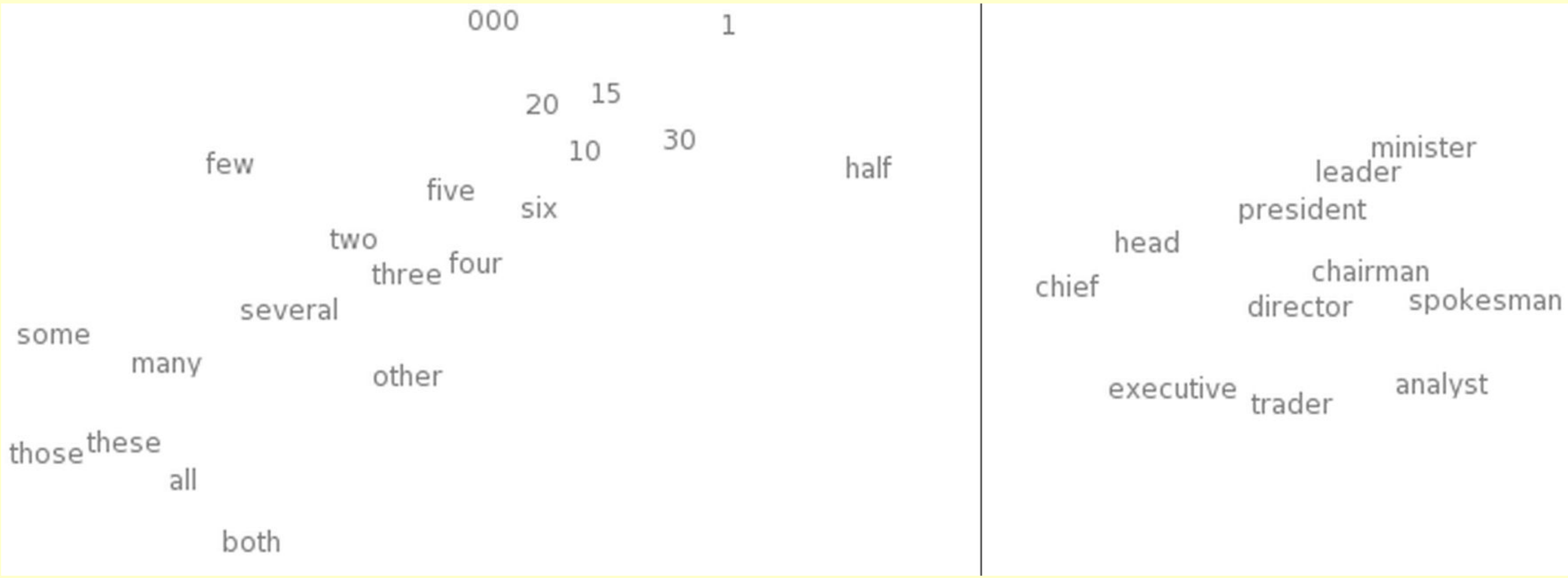
"Distributed stochastic neighbor embedding" (Maaten & Hinton 2008)

Projects points into 2d space by minimizing KL-divergence between two probability distributions between pairs of objects in high-dimensional space, and pairs of objects in low-dimensional space

similar pairs (e.g. via Eucledian distance) have high probability, dissimilar pairs have low probability

Evaluation of word embeddings

Nearest neighbors using t-SNE visualization technique



From: http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

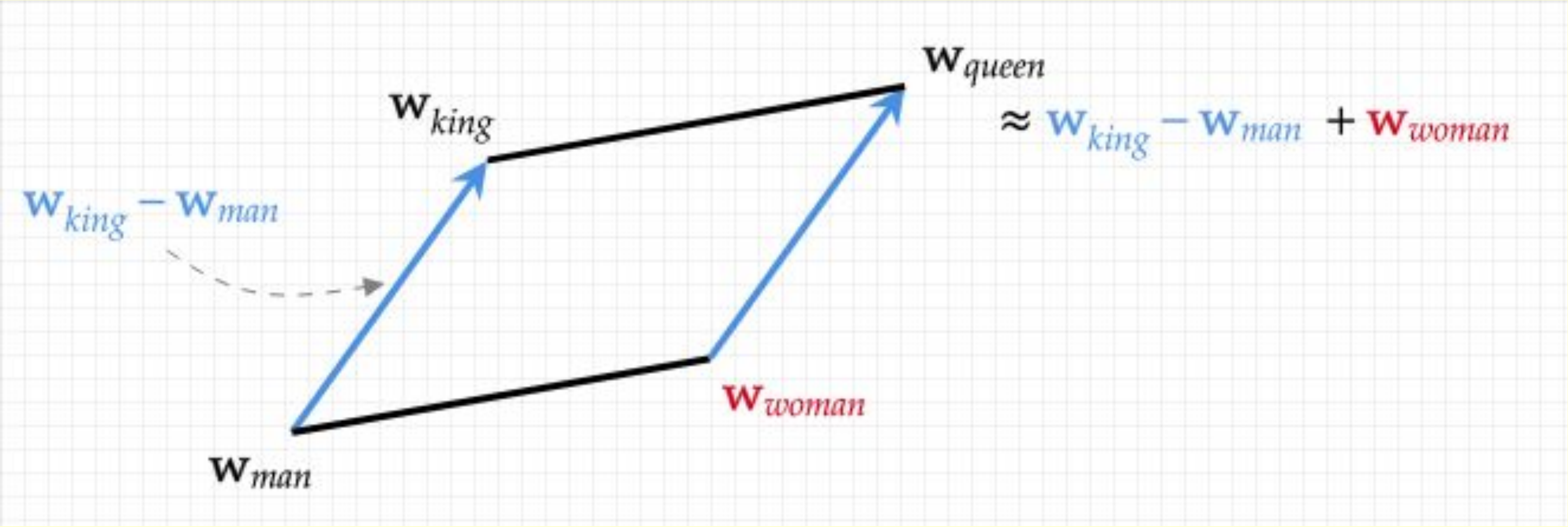
Word Similarity / Relatedness

	Target word	GloVe	SVD
1	phone	telephone	mobile
2	coffee	tea	drinks
3	grammar	vocabulary	grammatical
4	cohesiveness	cohesion	inclusiveness

Table 1: Examples of nearest neighbors in GloVe and SVD

Evaluation of word embeddings

Analogies



SOURCE:
https://www.ed.ac.uk/informatics/news-events/stories/2019/king-man-wo

Different Ways to do Analogies with the Same Embedding Scheme

ANALOGY

a to b is as a' to b'

TASK: Given a, b, and a', find b'

METHOD 1 (Mikolov et al 2013)

$b' = \text{argmax}_V (\cos b', b - a + a')$

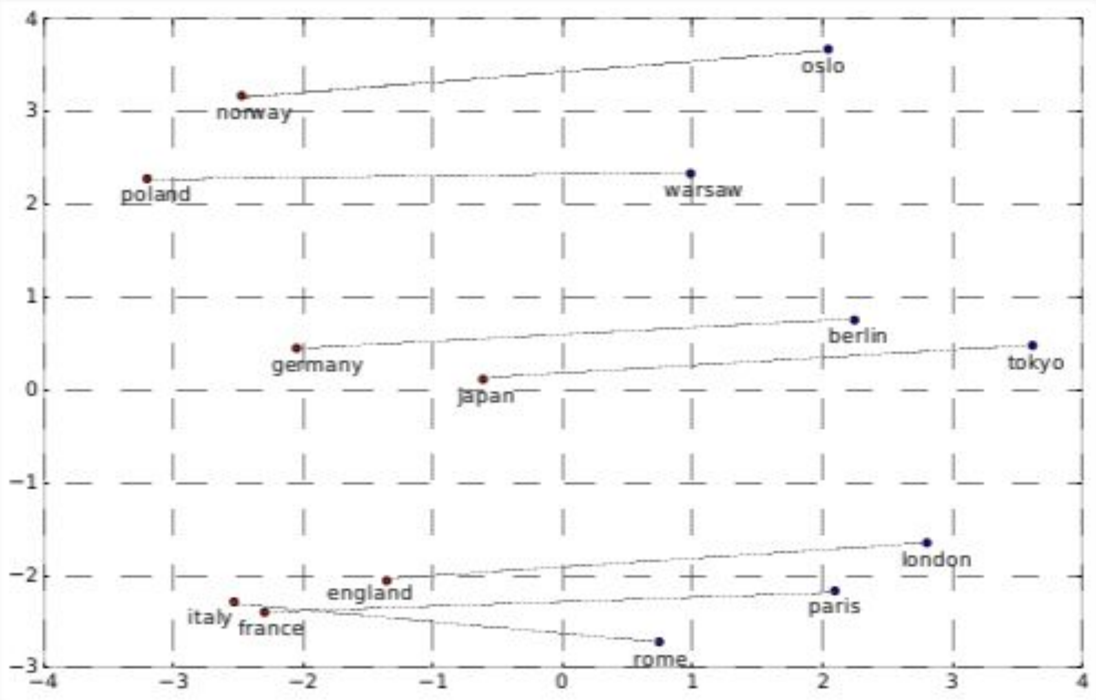
METHOD 2 (Levy & Goldberg 2014)

$b' = \text{argmax}_V (\cos b' - b, a' - a)$

The Analogy Test

- Task: given one word, identify its pair with a given linguistic relation.
- Method: solving word analogies: $\overrightarrow{Berlin}$ - $\overrightarrow{Germany}$ + $\overrightarrow{Japan}$ = $\overrightarrow{Tokyo}$ (Mikolov et al. 2013)
- GloVe: 80% accuracy!

Linear relations between countries and capitals in GloVe



Google Analogy Test (Mikolov et al 2013)

9 morphological categories: adjective-to-adverb, comparatives, superlatives, verb:present-participle, country-nationality, verb:past-tense, verb:3PsSg-plural, opposites.

5 semantic categories: common countries and capitals, countries and capitals of the world, city-in-state, country-and-currency, male:female.

20-70 unique word pairs per category.

8,869 semantic and 10,675 morphological questions in total.

Bigger Analogy Test (Gladkova et al 2016)

Inflectional morphology	Nouns	regular plurals *(student:students)*, plurals with orthographic changes *(wife:wives)*
	Adjectives	comparative degree *(strong:stronger)*, superlative degree *(strong:strongest)*
	Verbs	infinitive: 3Ps.Sg *(follow:follows)*, infinitive: participle *(follow:following)*, infinitive: past *(follow:followed)*, participle: 3Ps.Sg *(following:follows)*, participle: past *(following:followed)*, 3Ps.Sg : past *(follows:followed)*
Derivational morphology	Stem change	verb+er *(bake:baker)*, verb+able *(edit:editable)*, verb+ation *(continue:continuation)*, verb+ment *(argue:argument)*
	No stem change	re+verb *(create:recreate)*, noun+less *(home:homeless)*, adj.+ness *(mad:madness)*, un+adj. *(able:unable)*, adj.+ly *(usual:usually)*, over+adj. *(used:overused)*
Lexicographic semantics	Hypernyms	animals *(turtle:reptile)*, miscellaneous *(peach:fruit)*
	Hyponyms	miscellaneous *(color:white)*
	Meronyms	part-whole *(car:engine)*, substance *(sea:water)*, member *(player:team)*,
	Antonyms	opposites *(up:down)*, gradable *(clean:dirty)*
	Synonyms	exact *(sofa:couch)*, intensity *(cry:scream)*
Encyclopedic semantics	Animals	the young *(cat:kitten)*, sounds *(dog:bark)*, shelter *fox:den*
	Geography	capitals *(Athens:Greece)*, languages *(Peru:Spanish)*, UK city:county *York:Yorkshire*
	People	occupation *(Lincoln:president)*, nationalities *(Lincoln:American)*
	Other	thing:color *(blood:red)*, male:female *(actor:actress)*

Analogies?

Why should different linguistic relations translate to exactly the same vector offsets for all words?

Other Methods of Evaluation

<u>Extrinsic evaluation via performance on downstream tasks</u> (POS-tagging, chunking, NER, sentiment polarity, NLI)

<u>Behavioral evaluation:</u> correlation with similarity judgments, intrusion, N400 effect, fMRI scans, eye-tracking, and semantic priming data.

Benefits of Neural Approaches

Easy to learn, especially with good linear algebra libraries.

Highly parallel problem: minibatching, GPUs, distributed models.

Can predict other discrete aspects of context (dependencies, POS tags, etc). Can estimate these probabilities with counts, but sparsity quickly becomes a problem.

Can predict/condition on continuous contexts: e.g. images.

Comparison with count-based methods

Count based and objective-based models: same general idea.

Word2Vec == PMI matrix factorization of count based models (Levy and Goldberg, 2014)

Count-based and most neural models have equivalent performance when properly hyper parameters are properly optimized (Levy et al. 2015)

Tokenization

Remember tokenization?

Tokenization is splitting text into meaningful units that form your vocabulary

- words, punctuation
- can use white-space segmentation to get words (roughly)

Tokens are not just words

Word-internal punctuation: Ph.D., AT&T, Google.com, 555,500.50

Expanding clitics: I'm → I am

Multiword tokens: New York, Rock 'n' roll

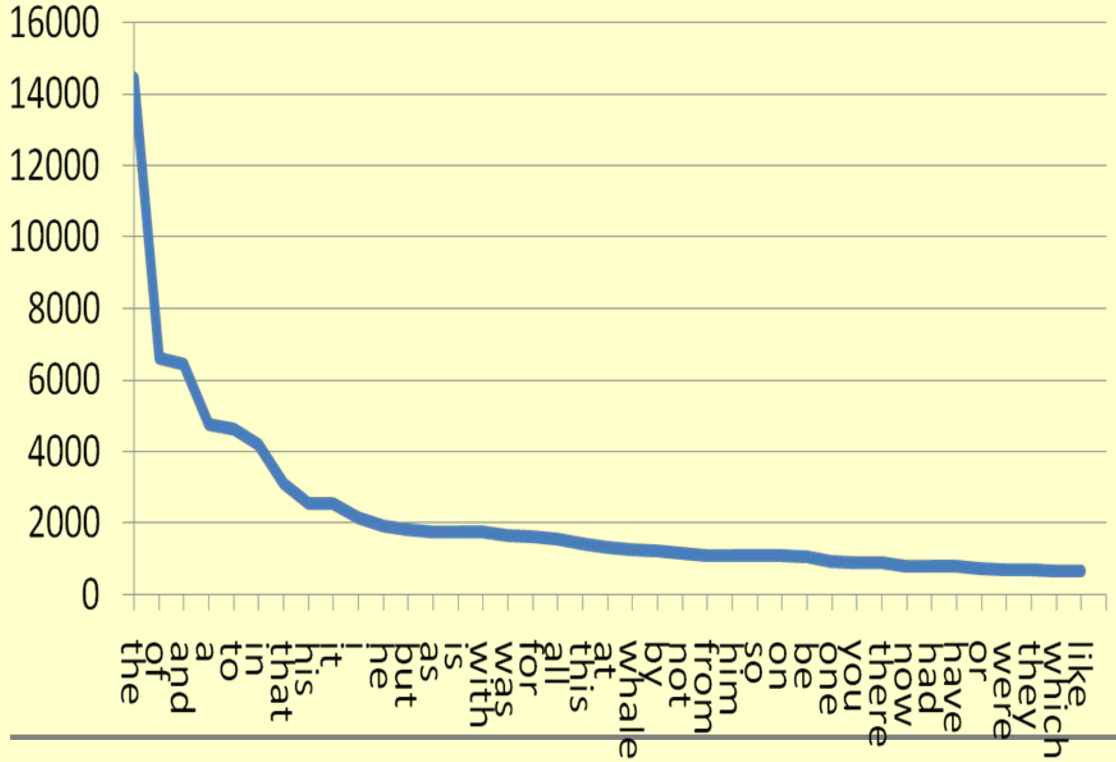
Word vectors are really token vectors!

Zipf's law – representations are always sparse

Zipf’s Law (long tail phenomenon):

The frequency of any word is inversely proportional to its rank in the frequency table

A large number of events occur with low frequency. You might have to wait an arbitrarily long time to get valid statistics on low frequency events



Words with frequency of less than one in 50,000 make up 20-30% of newswire reports (Dunning, 1993)

Out-of-vocabulary words

Texts are sparse!

Many words (tokens) never seen even in large texts.

- What if your test data contains words that are not in your training data?
- You get the so-called "out of vocabulary" words (OOV)
- People would add a special "<unknown>" token to their vocabulary
- During training, use this token for unseen words in validation data.

A better solution – subword tokenization

- Can we construct a vocabulary of meaningful subwords?
- Use the data to tell us how to tokenize!
- Tokens could be subwords, i.e. variable-length character ngrams
- That way, out-of-vocabulary words can sometimes be represented reasonably.

Subword tokenization

- Byte-Pair Encoding (Sennrich et al, 2016)
- WordPiece (Schuster and Nakajima, 2012)

Two parts:

Token learner takes a raw training corpus and induces a vocabulary (a set of tokens)

Token segmenter takes a raw test corpus and tokenizes it according to the vocabulary

Byte Pair Encoding (BPE) Tokenization

- Let the initial vocabulary be the the set of individual characters = {A, B, C, D, … , a, b, c, ….}

Repeat

- Choose two symbols that are most often adjacent in the training corpus (e.g. “t” and “h”)
- Add a new merged symbol “th” to the vocabulary
- Replace every adjacent “t” and “h” with “th” in the training corpus

Until k merged have been done.

Byte Pair Encoding (BPE) Tokenization

- BPE will often deduce frequent subwords:

  morphemes like –est, –er, un–, etc.

- Most subword algorithms are run inside space-separated tokens, adding a special end-of-word character before each space.
- That way, word-final combinations of letters get a different treatment than word-internal combinations

BPE Example

Training text: “This runner jumped higher”

- Initial vocabulary:

  t, h, i, s, r, u, n, e, j, u, m, d, j, i, g, <eow>

- Initial tokenization

  t h i s <eow> r u n n e r <eow> j u m p e d <eow> h i g h e r <eow>

- Merge “e” and “r”:

  T h i s <eow> r u n er <eow> j u m p e d <eow> h i g h er <eow>

- Merge “er” and “<eow>

  T h i s <eow> r u n n er<eow> j u m p e d <eow> h i g h er<eow>

WordPiece

- Popularized by BERT, which was the first pretrained transformer encoder.
- Instead of relying on the frequency of token pairs during the merge, at each merge step:
  - Train a language model at each step
  - Merge the token that maximizes the likelihood of the training data, i.e.
    - $p(t1\ t2) > p(t1)\ p(t2)$
    - $MI(t1, t2)$ is greater than for any other pair

Lab : Homework 2