CSE 4392 Special Topics
Natural Language Processing

Word Embedding

2024 Spring

How to Represent Words?

- N-gram language models:

It is 76 F and ______.

$$P(w \mid \text{it is 76 F and})$$

[0.0001, 0.1, 0, 0, 0.002, ..., 0.3, ..., 0]

red sunny

- Text classification:

$$P(y = 1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

I like this movie. 👍 $\boldsymbol{x}^{(1)}$ [0, 1, 0, 0, 0, ..., 1, ..., 1]

I don't like this movie. 👎 $\boldsymbol{x}^{(2)}$ [0, 1, 0, 1, 0, ..., 1, ..., 1]

don't

Feature: don't

Representing Words as Discrete Symbols

- In traditional NLP, we regard words as discrete symbols: hotel, conference, motel — a localist representation

one 1, the rest 0's

- Words can be represented by one-hot vectors:

hotel = [000000000010000]

motel = [000100000000000]

- Vector dimension = num of words in vocabulary (e.g., 500,000)
- No way to encode similarity between words!

Quiz: One-hot vectors

- Using one-hot vectors to represent words, why is there no way to encode similarity between the words?

Represent Words by their Context

- **Distributional hypothesis**: words that occur in similar contexts tend to have similar meanings

J.R. Firth 1957
"*You shall know a word by the company it keeps*"
One of the most successful ideas of modern statistical NLP!

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent *banking*.

Distributional Hypothesis

"tejuino"

- C1: A bottle of ___ is on the table.
- C2: Everybody likes ___.
- C3: Don't have ___ before you drive.
- C4: We make ___ out of corn.

Distributional Hypothesis

C1: A bottle of ___ is on the table.

C2: Everybody likes ___.

C3: Don't have ___ before you drive.

C4: We make ___ out of corn.

	C1	C2	C3	C4
tejuino	1	1	1	1
loud	0	0	0	0
motor-oil	1	0	0	0
tortillas	0	1	0	1
choices	0	1	0	0
wine	1	1	1	0

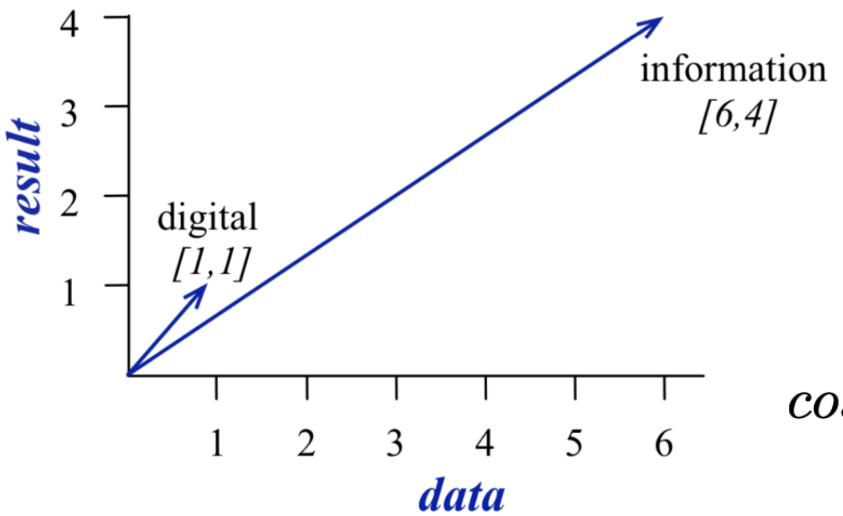
"words that occur in similar contexts tend to have similar meanings"

Words as Vectors

- We'll build a new model of meaning focusing on similarity
  - Each word is a vector
  - Similar words are "nearby in space"
- A first solution: we can just use context vectors to represent the meaning of words!
  - Word-word co-occurrence matrix:

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Words as Vectors



$$cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

$$cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{V} u_i v_i}{\sqrt{\sum_{i=1}^{V} u_i^2}\sqrt{\sum_{i=1}^{V} v_i^2}}$$

The range of cos (.) is [0, 1]

Quiz: Compute the cosine similarity between "digital" [1, 1] and "information" [6, 4].

WORDS AS VECTORS

- Problem: using raw frequency counts is not always very good..
  - Solution: let's weight the counts!
  - PPMI = Positive Pointwise Mutual Information

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

w and *c* are two words in the same text window

	computer	data	result	pie	sugar
cherry	2	8	9	442	25
strawberry	0	0	1	60	19
digital	1670	1683	85	5	4
information	3325	3982	378	5	13

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

https://gyan-mittal.com/nlp-ai-ml/nlp-word-embedding-svd-based-methods/

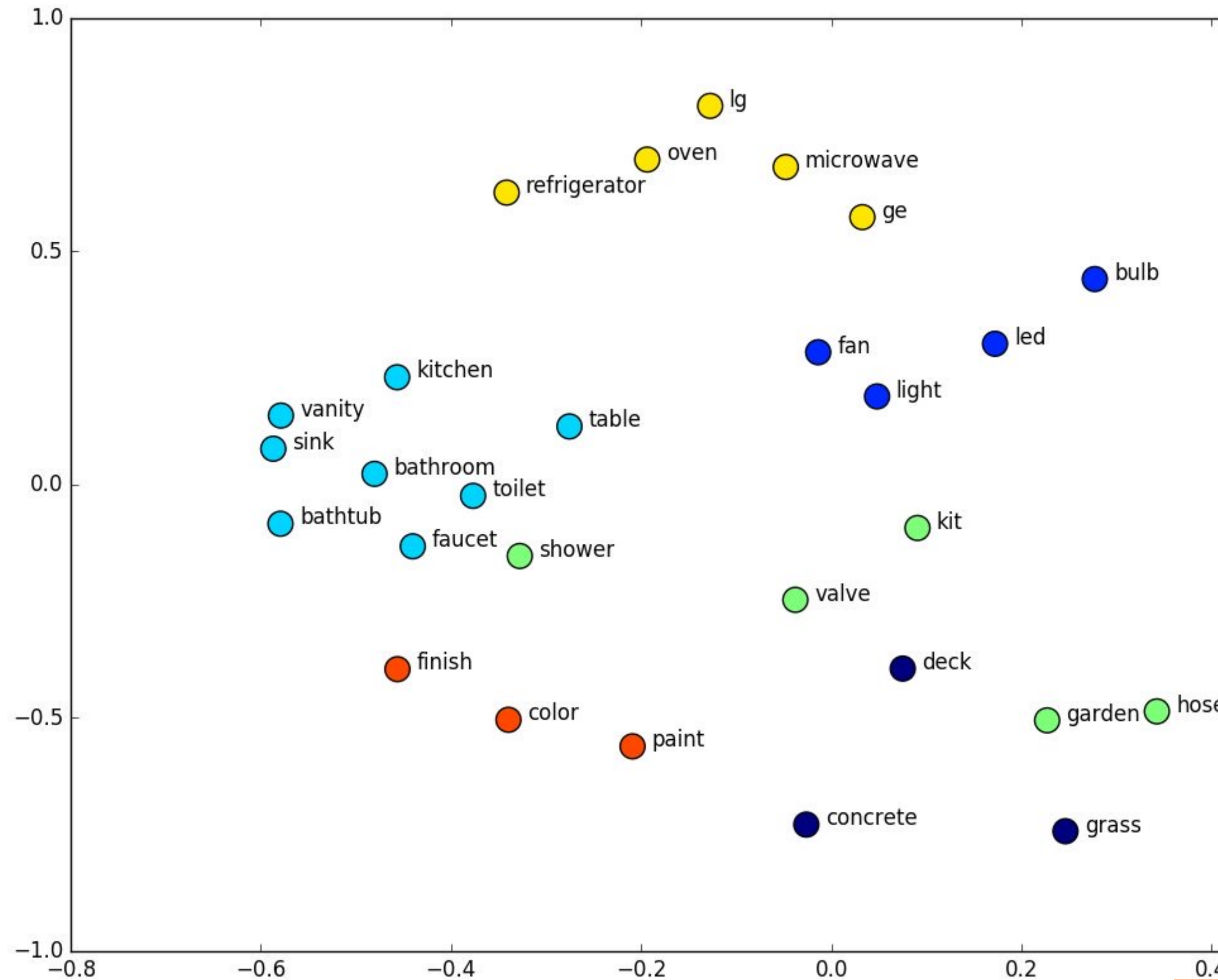
Sparse vs. Dense Vectors

- Still, the vectors we get from word-word occurrence matrix are sparse (most are 0's) & long (vocabulary size)

- Alternative: we want to represent words as **short** (50-300 dimensional) & **dense** (real-valued) vectors
  - The focus of this lecture
  - The basis of all the modern NLP systems

Dense Vectors

$$\text{faucet}=\begin{pmatrix} 0.246 \\ 0.793 \\ -0.177 \\ -2.212 \\ 12.322 \\ 3.292 \\ -0.768 \\ 3.211 \end{pmatrix}$$



Why Dense Vectors?

- Short vectors are easier to use as features in ML systems
- Dense vectors may generalize better than storing explicit counts
- They do better at capturing synonymy
  - w_1 co-occurs with “car”, w_2 co-occurs with “automobile” → w_1 and w_2 are synonyms

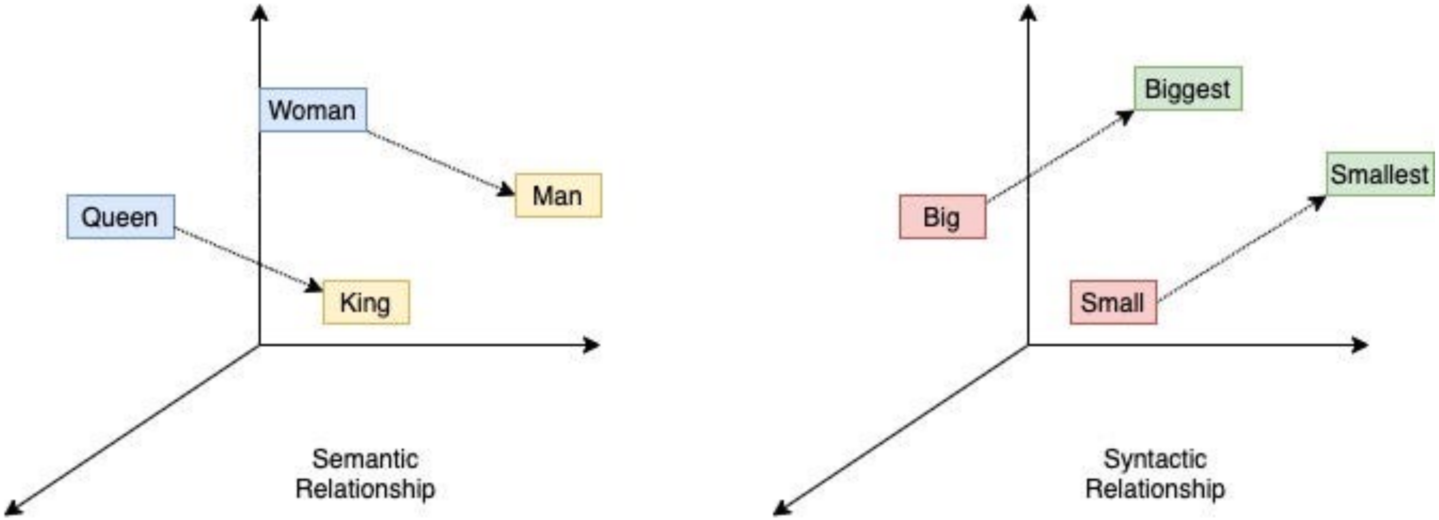
Different Methods for Getting Dense Vectors

- Singular value decomposition (SVD)

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

  - Limitations:
    - Vocab usually big, so matrix is big and difficult to train
    - Imbalance due to high frequency words
    - As new words are added to the vocab, matrix size changes, need to re-train!
  - https://gyan-mittal.com/nlp-ai-ml/nlp-word-embedding-svd-based-methods/
- word2vec and friends: “learn” the vectors!
  - Much more elegant than SVD

WORD2VEC AND FRIENDS



- (Mikolov et al, 2013): Distributed Representations of Words and Phrases and their Compositionality

Word2vec

- Input: a large text corpus, V, d
  - V: a pre-defined vocabulary
  - d: dimension of word vectors (e.g. 300)
  - Text corpora:
    - Wikipedia + Gigaword 5: 6B
    - Twitter: 27B
    - Common Crawl: 840B
- Output:

$$f : V \to \mathbb{R}^d$$

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \qquad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

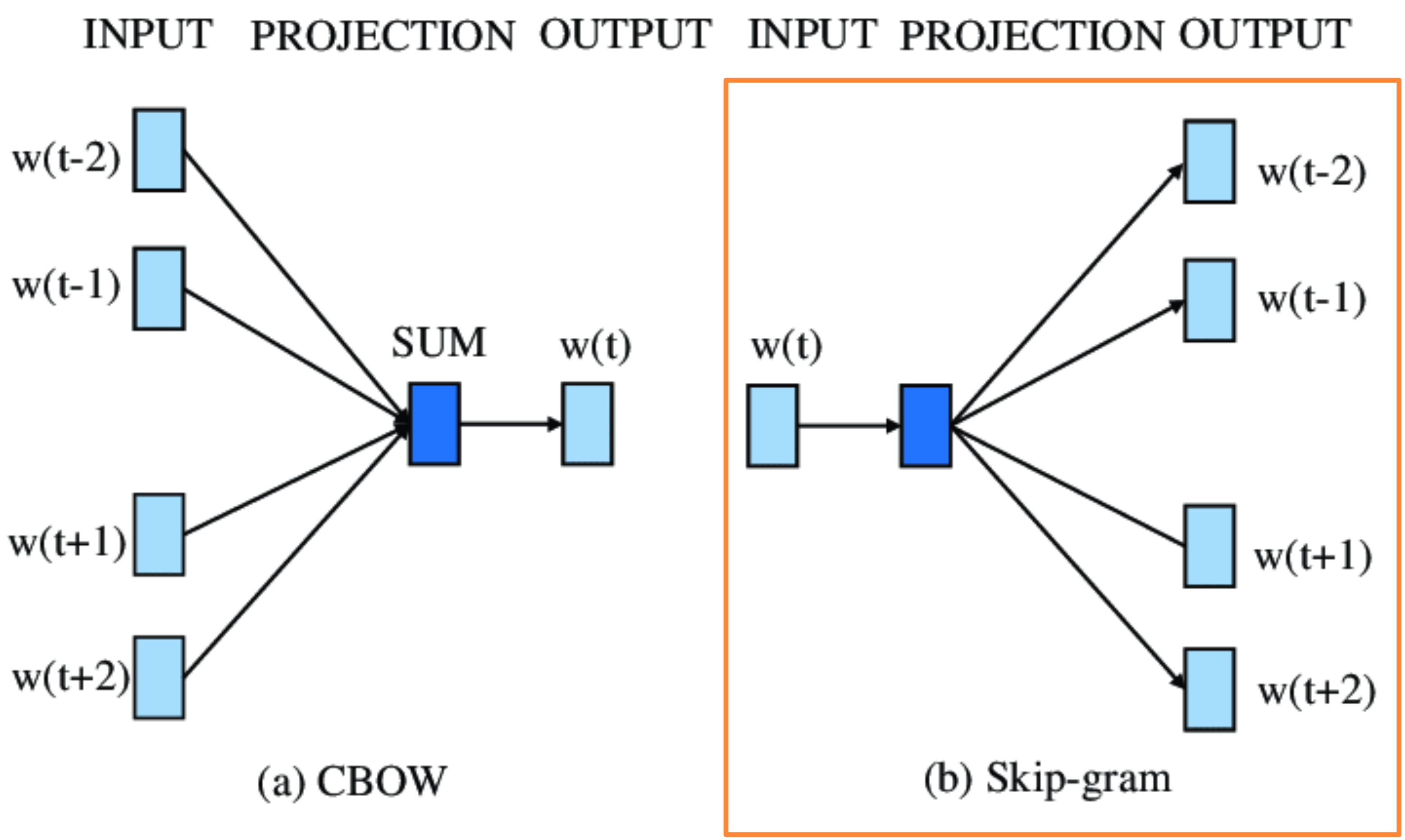
$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \qquad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Word2vec

- Word = "sweden"

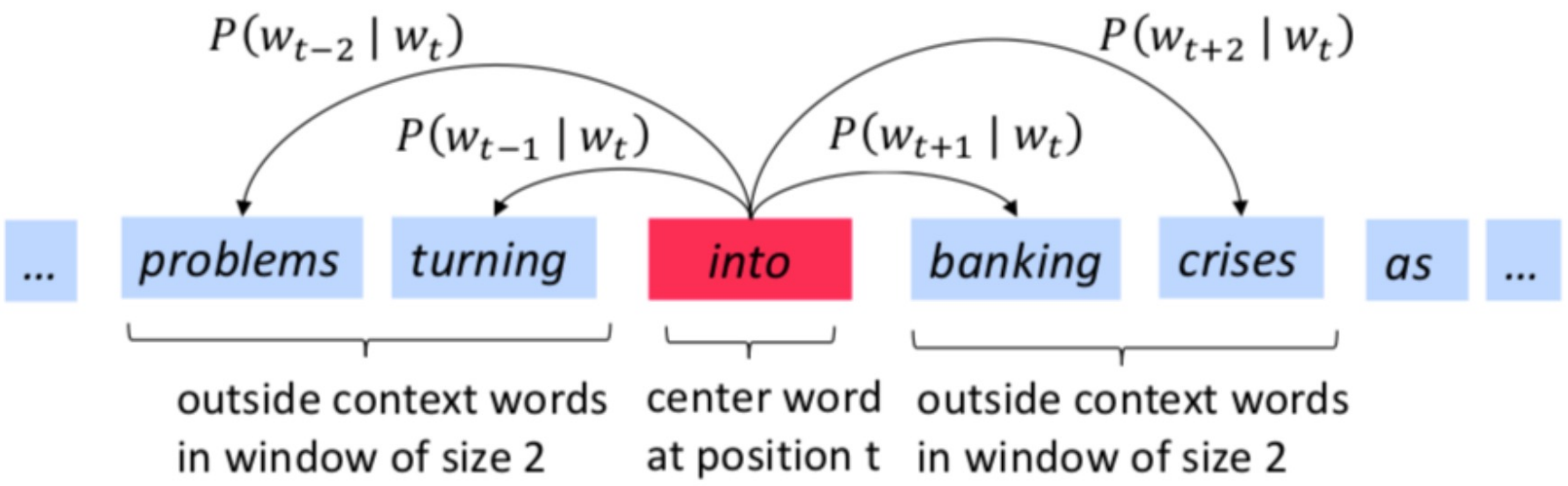
Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

Word2vec


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

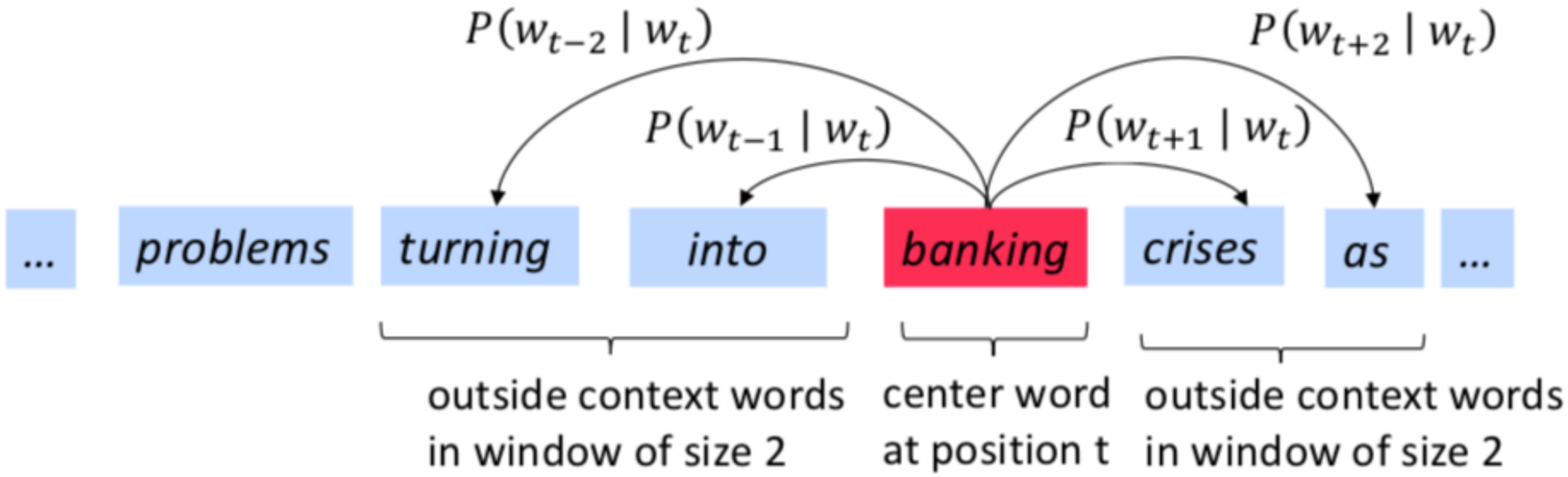
Skip-gram

- Idea: Use the *center* word to predict its *context* words.
- Context: a fixed window of $2m$



Skip-gram

- Next time step t:



Skip-gram: Objective function

- For each position t = 1, 2, …, T, predict context words within context size m, given center word w_j:

all the parameters to be optimized

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} \mid w_t; \theta)$$

- The objective function *J(θ)* is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log \mathcal{L}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} \mid w_t; \theta)$$

How to Define: $P(w_{T+J} \mid w_T; \Theta)$?

- There are two word embeddings (vectors) for each word in the vocabulary:

$$\mathbf{u}_i \in \mathbb{R}^d \text{ : embedding for target word } i$$

$$\mathbf{v}_{i'} \in \mathbb{R}^d \text{ : embedding for context word } i'$$

- Use inner product $\boldsymbol{u}_i \cdot \boldsymbol{v}_{i'}$ to measure how likely a word *i* appears with context word *i'*, the larger the better.

"softmax" we learned last time!

$$P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$$

$\theta = \{\{\mathbf{u}_k\}, \{\mathbf{v}_k\}\}$ are all the parameters in this model!

How to Train the Model

- Calculate all the gradients together!

$$\theta = \{\{\mathbf{u}_k\}, \{\mathbf{v}_k\}\}$$

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m \le j \le m, j \ne 0} \log P(w_{t+j} \mid w_t; \theta) \qquad \nabla_\theta J(\theta) = ?$$

Quiz: Suppose the vocab is V in the corpus, how many parameters in θ to train in total?

- We can apply stochastic gradient descent (SGD)!

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta J(\theta)$$

- Let's go through the math.

Warm-up

$$f(x) = \exp(x) \qquad \frac{df}{dx} = \exp(x)$$

$$f(x) = \log(x) \qquad \frac{df}{dx} = \frac{1}{x}$$

chain rule:

$$f(x) = f_1(f_2(x)) \qquad \frac{df}{dx} = \frac{df_1(z)}{dz}\frac{df_2(x)}{dx} \qquad z = f_2(x)$$

$$f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{a} \qquad \frac{\partial f}{\partial \mathbf{x}} = \mathbf{a}$$

$$\frac{\partial f}{\partial \mathbf{x}} = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}]$$

Computing the Gradients

- Consider one pair of target/context words (t, c):

$$y = -\log\left(\frac{\exp(\mathbf{u}_t \cdot \mathbf{v}_c)}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}\right)$$

$$\begin{aligned}
\frac{\partial y}{\partial \mathbf{u}_t} &= \frac{\partial\left(-\mathbf{u}_t \cdot \mathbf{v}_c + \log\left(\sum_{k \in V} \exp\left(\mathbf{u}_t \cdot \mathbf{v}_k\right)\right)\right)}{\partial \mathbf{u}_t} \\
&= -\mathbf{v}_c + \frac{\sum_{k \in V} \frac{\partial \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}{\partial \mathbf{u}_t}}{\sum_{k \in V} \exp\left(\mathbf{u}_t \cdot \mathbf{v}_k\right)} \\
&= -\mathbf{v}_c + \frac{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)\mathbf{v}_k}{\sum_{k \in V} \exp\left(\mathbf{u}_t \cdot \mathbf{v}_k\right)} \\
&= -\mathbf{v}_c + \sum_{k \in V} P(k \mid t)\mathbf{v}_k
\end{aligned}$$

$$\frac{\partial y}{\partial \mathbf{v}_k} = -1(k = c)\mathbf{u}_t + P(k \mid t)\mathbf{u}_t$$

(Try to derive this!)

Putting it Together

- Input: text corpus, context size m, embedding size d, vocab $\boldsymbol{V}$
- Initialize $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ randomly
- Work through the training corpus and collection training data (t, c):
  - Update:

$$\mathbf{u}_t \leftarrow \mathbf{u}_t - \eta \frac{\partial y}{\partial \mathbf{u_t}}$$
$$\mathbf{v}_k \leftarrow \mathbf{v}_k - \eta \frac{\partial y}{\partial \mathbf{v_k}}, \forall k \in V$$

  - Any problem here?

Skip-gram with Negative Sampling (SGNS)

- Problem: for each training data pair (t, c), you need to update the embedding of every word in the vocab. That's too expensive!

$$\frac{\partial y}{\partial \mathbf{u}_t} = -\mathbf{v}_c + \sum_{k \in V} P(k \mid t)\mathbf{v}_k$$

$$\frac{\partial y}{\partial \mathbf{v}_k} = -1(k = c)\mathbf{u}_t + P(k \mid t)\mathbf{u}_t$$

- *Negative Sampling*: instead of considering all the words in V, let's randomly sample K (5-20) negative examples.

softmax: $$y = -\log\left(\frac{\exp(\mathbf{u}_t \cdot \mathbf{v}_c)}{\sum_{k \in V} \exp(\mathbf{u}_t \cdot \mathbf{v}_k)}\right)$$

NS: $$y = -\log(\sigma(\mathbf{u}_t \cdot \mathbf{v}_c)) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log(\sigma(-\mathbf{u}_t \cdot \mathbf{v}_j))$$

Skip-gram with negative Sampling (SGNS)

$$y = -\log(\sigma(\mathbf{u}_t \cdot \mathbf{v}_c)) - \sum_{i=1}^{K} \mathbb{E}_{j \sim P(w)} \log(\sigma(-\mathbf{u}_t \cdot \mathbf{v}_j))$$

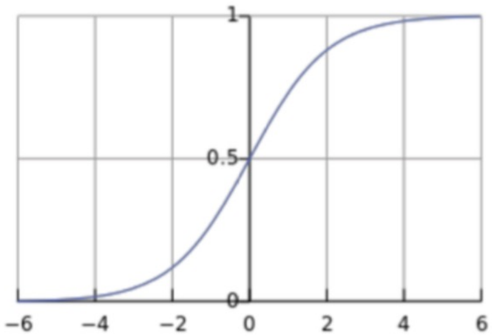
positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

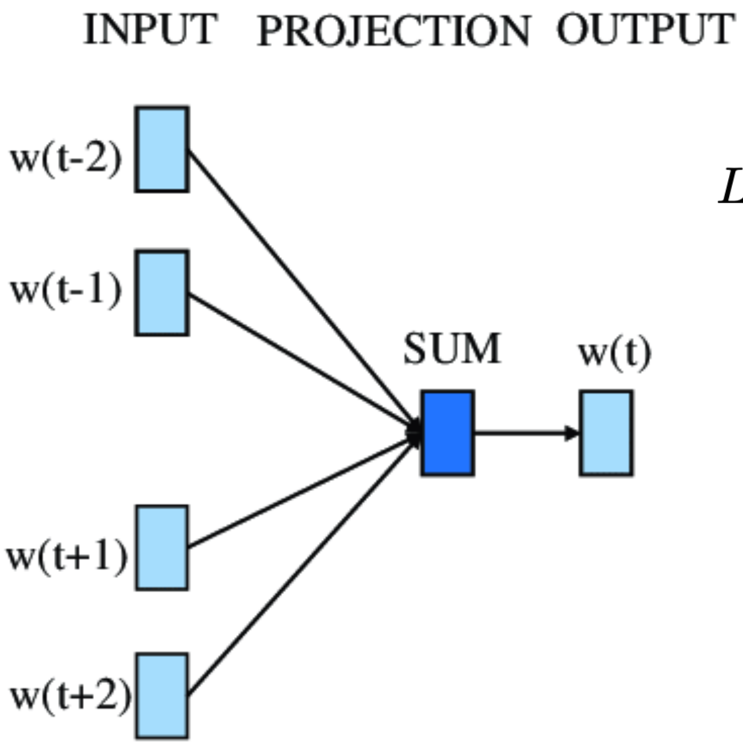


- Same as training a **logistic regression** for binary classification!

$$P(D = 1 \mid t, c) = \sigma(\mathbf{u}_t \cdot \mathbf{v}_c)$$

- Compute the gradient in assignment!

Continous Bag of Words (CBOW)



$$L(\theta) = \prod_{t=1}^{T} P\left(w_t \mid \{w_{t+j}\}, -m \leq j \leq m, j \neq 0\right)$$

$$\bar{\mathbf{v}}_t = \frac{1}{2m} \sum_{-m \leq j \leq m, j \neq 0} \mathbf{v}_{t+j}$$

(average context vector for w_t)

$$P(w_t \mid \{w_{t+j}\}) = \frac{\exp(\mathbf{u}_{w_t} \cdot \bar{\mathbf{v}}_t)}{\sum_{k \in V} \exp(\mathbf{u}_k \cdot \bar{\mathbf{v}}_t)}$$

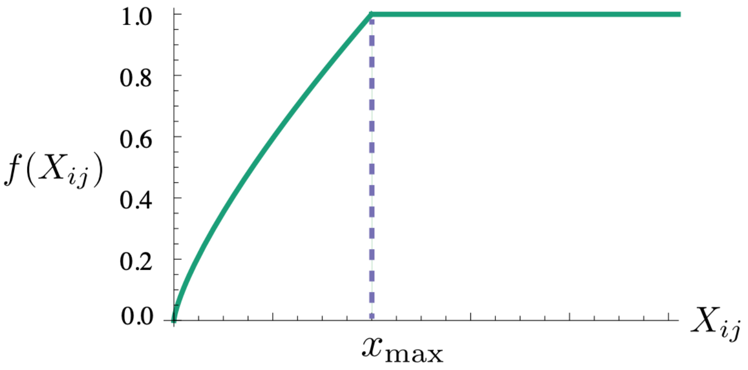
GloVe: Global Vectors

- Skip-gram and CBoW uses local context
  - Slow to train when the corpus is very large
- Let's take the global co-occurrence statistics X_{ij}:

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

  - X_{ij} tabulate the number of times word j occurs in the context of word i.

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} .$$



GloVe: Global Vectors

- Nearest word to frog:
  - frogs
  - toad
  - litoria
  - leptodactylidae
  - rana
  - lizard
  - eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

(Pennington et al, 2014): GloVe: Global Vectors for Word Representation

FASTTEXT: SUB-WORD EMBEDDINGS

- Similar as Skip-gram, but break words into n-grams with n = 3 to 6

  where: 3-grams: <wh, whe, her, ere, re>

  4-grams: <whe, wher, here, ere>

  5-grams: <wher, where, here>

  6-grams: <where, where>

- Replace $\boldsymbol{u}_i \cdot \boldsymbol{v}_j$ by

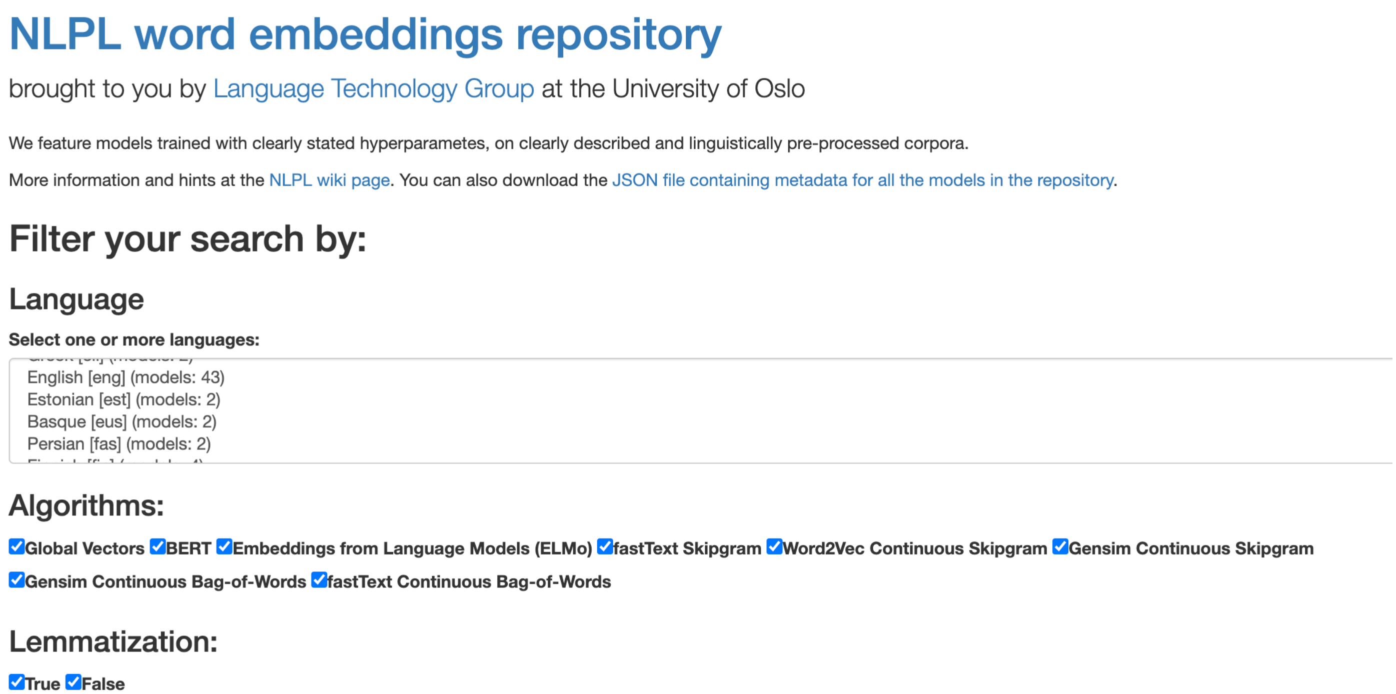
$$\sum_{g \in ngrams(w_i)} \boldsymbol{u}_g \cdot \boldsymbol{v}_j$$

- More to come: contextualized word embeddings

(Bojanowski et al, 2017): Enriching Word Vectors with Subword Information

Pre-trained Word Embeddings Available

- word2vec: https://code.google.com/archive/p/word2vec/
- GloVe: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/

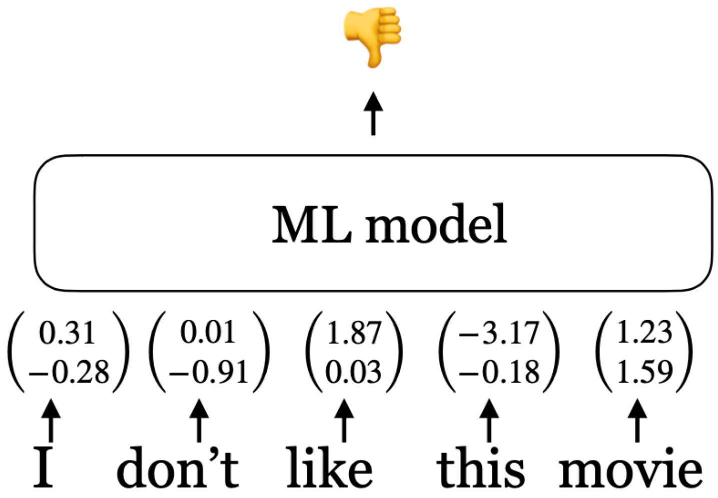


- Differ in algorithms, text corpora, dimensions, cased/uncased…

Evaluating Word Embeddings

Extrinsic vs Intrinsic Evaluation

- Extrinsic evaluation
  - Plug these word embeddings into a real NLP system and see if this improves the performance
  - Could take a long time but still the most important evaluation metric
- Intrinsic evaluation
  - Evaluate on a specific/intermediate subtask
  - Fast to compute
  - Not clear if it really helps the downstream task



Intrinsic Evaluation

- Word similarity task
  - Example dataset: WordSim-353
    353 pairs of words with human judgement
  - http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Cosine similarity:

$$\cos(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i \cdot \boldsymbol{u}_j}{||\boldsymbol{u}_i||_2 \times ||\boldsymbol{u}_j||_2}.$$

Metric:
Spearman Rank Correlation

Intrinsic Evaluation

- Word Similarity Results (Spearman correlations):

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

Intrinsic Evaluation

- Word analogy

  man:woman ≈ king: ?

$$\arg\max_i \left(\cos(\mathbf{u}_i, \mathbf{u}_b - \mathbf{u}_a + \mathbf{u}_c)\right)$$

Semantic

Austin:Texas ≈ ? :California

Syntactic

bad:worst ≈ hot: ?

More examples at:

http://download.tensorflow.org/data/questions-words.txt