CS 546 – Advanced Topics in NLP

Dilek Hakkani-Tür

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

CONVAI

CDS

Siebel School of Computing and Data Science

Topics for Today

- Distributional Similarity
- Sparse Word Representations
- Word Embeddings and Word2Vec
- Language Modeling
- Unexpected things we learn with word embeddings

What is Distributional Meaning?

- Typically, words are treated as discrete, arbitrary symbols in NLP systems.
- But words have lots of interesting relationships to each other!
- A lot of the previous work examined how to represent these:
  - Manually-built resources like WordNet provide one way to define words and their similarities
  - Distributional representations and word embeddings are another way
  - And they can be learned automatically from large text collections

Distributional View of Lexical Categories

- Characterizes knowledge of a word in terms of “the company it keeps” (Firth, 1957)
- Word categories can be defined by the context in which they appear
- Such characterization is the idea behind static word representations estimated from data (Mikolov et al., 2013) and modern large language models.

- For a word w, find all the contexts $w_1 w w_2$ in which w appears
- Find all words w' that share many frequent contexts

- Use a LARGE text collection to get good counts

A model of meaning focusing on similarity

- Representing discrete types (i.e., words) as vectors
- Each word = a vector
- Similar words are "nearby in space"

Word as a Vector

- Called an "embedding" because it's embedded into a space
- The standard way to represent meaning in NLP
- Fine-grained model of meaning for similarity
  - NLP tasks like sentiment analysis
    - With words, requires **same** word to be in training and test
    - With embeddings: ok if **similar** words occur.
  - Question answering, conversational agents, etc.

Two kinds of Embeddings

- **Sparse Vectors, formed by counting**
  - Term frequency (TF), term frequency, inverse document frequency (TF.IDF)
  - A common baseline model
  - Words are represented by a simple function of the counts of nearby words

- **Dense, learned word embeddings**
  - E.g. Word2Vec, Representation is created by training a classifier to distinguish nearby and far-away words

Topics for Today

- Distributional Similarity
- Sparse Word Representations
- Word Embeddings and Word2Vec
- Language Modeling
- Unexpected things we learn with word embeddings

Word-word Matrix

- Two **words** are similar in meaning if their context vectors are similar

sugar, a sliced lemon, a tablespoonful of	**apricot**	jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

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

How to represent words with counts?

- Count how often “apple” occurs close to other words in a large text collection (corpus):

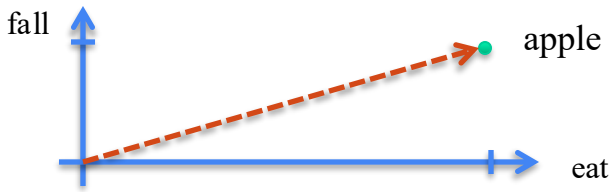
eat	fall	ripe	slice	peel	tree	throw	fruit	pie	bite	crab
794	244	47	221	208	160	145	156	109	104	88

- Do the same for “orange”:

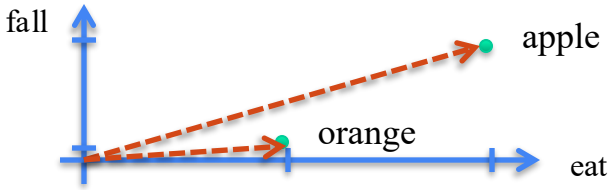
eat	fall	ripe	slice	peel	tree	throw	fruit	pie	bite	crab
265	22	25	62	220	64	74	111	4	4	8

- Interpret counts as coordinates.

Every context word becomes a dimension.



Visualization of Word Vectors

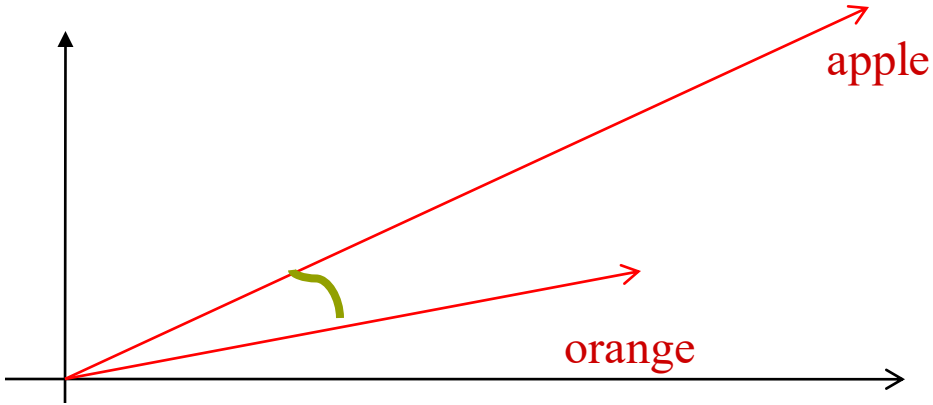


Similarity between two words as proximity in space

How to use word vectors to measure similarity?

Cosine similarity:

$$cos(\vec{p}, \vec{q}) = \frac{\sum_{i=1}^{n} p_i \cdot q_i}{\sqrt{\sum_{i=1}^{n} p_i^2} \cdot \sqrt{\sum_{i=1}^{n} q_i^2}}$$



Use the angle between vectors instead of point distance to get around word frequency issues

Some words are more informative than others

- Function words co-occur frequently with all words
  - That makes them less informative
- They have much higher co-occurrence counts than content words
  - They can “drown out” more informative contexts
- Some counts for “letter” in “Pride and Prejudice”.

the	to	of	and	a	her	she	his	is	was	in	that
102	75	72	56	52	50	41	36	35	34	34	33

had	i	from	you	as	this	mr	for	not	on	be	he
32	28	28	25	23	23	22	21	21	20	18	17

but	elizabeth	with	him	which	by	when	jane
17	17	16	16	16	15	14	12

All the most frequent co-occurring words are function words.

TF.IDF: combines two factors

- **tf: term frequency**. frequency count (usually log-transformed):

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Idf: inverse document frequency:**

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

Total # of docs in collection

# of docs that have word i

Words like "the" or "good" have very low idf

tf-idf value for word t in document d:

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

Alternative to tf.idf:
pointwise mutual information

TF.IDF Alternative: Pointwise Mutual Information (PMI)

- Whether a context word is **particularly informative** about the target word.

- Degree of association between target and context, rather than co-occurrence:
  - High association: high co-occurrence with specific words, lower with everything else
  - Low association: lots of co-occurrence with all words

Pointwise Mutual Information

Pointwise mutual information:

Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

PMI between two words: (Church & Hanks 1989)

Do words x and y co-occur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Sparse Vectors

- TF.IDF and PMI vectors are
  - **long** (length |V|= 20,000 to 50,000)
  - **sparse** (most elements are zero)

Topics for Today

- Distributional Similarity
- Sparse Word Representations
- Word Embeddings and Word2Vec
- Language Modeling
- Unexpected things we learn with word embeddings

Word Embeddings

- **AIM:** learn vectors to represent the semantics of a word from its contextual uses (just like distributional measure)
  - Words that have
    similar meanings -> close to each other in the vector space
    different meanings -> far from each other
- Dense vectors
  - **short** (length 50-1000)
  - **dense** (most elements are non-zero)

- Why dense vectors?
- How are they built?
- What are they good for?
- How are they used?
- Why are they an improvement over distributional semantics precursor?

Sparse versus Dense Vectors

- Why dense vectors?
  - Short vectors may be easier to use as **features** in machine learning (less weights to tune)
  - Dense vectors may **generalize** better than storing explicit counts
  - They may do better at capturing synonymy:
    - *car* and *automobile* are synonyms; but are distinct dimensions
      - a word with *car* as a neighbor and a word with *automobile* as a neighbor should be similar, but aren't
  - **In practice, they work better**

Learning Word Embeddings

- Train with just words (i.e., unlabeled data), in a supervised fashion, by constructing auxiliary tasks:
  - **Language modeling:** Given a sequence of words, predict the next word:

    Given $w_1,\ldots,w_{n-1}$, predict w_n
  - Given lexical context (i.e., previous and following words), predict the missing one (**CBOW model**):

    Given $w_1,\ldots,w_{n-1}$, and $w_{n+1},\ldots,w_{n+k}$, predict w_n
  - Given a word, predict words that occur within a window (independent of their position) (**skip-gram model**):

    Given w_n, predict $w_1,\ldots,w_{n-1}$, and $w_{n+1},\ldots,w_{n+k}$

Approaches to Learn Word Embeddings

- Word2Vec
- Stanford's GLoVe
- Facebook's FastText

- Typical format of word embeddings:

dog	-1.242	-0.360	0.573	0.367	0.600	-0.189	1.273	...
cat	-0.964	-0.610	0.674	0.351	0.413	-0.212	1.380	...

Word2Vec

- Idea: **predict** rather than **count**
- **Word2vec** (Mikolov et al., 2013)
- https://code.google.com/archive/p/word2vec/
- Instead of **counting** how often each word *w* occurs near "*apricot*"
  - Train a classifier on a binary **prediction** task:
    - Is *w* likely to show up near "*apricot*"?
  - Fake task: We don't actually care about this task
    - But we'll take the learned classifier weights as the word embeddings

Word2Vec – Skip-Gram Model

- Goal: predict surrounding words within a window of each word
- Objective function: maximize the probability of any context word given the current center word

$$w_1, w_2, \cdots, \underbrace{w_{t-m}, \cdots, w_{t-1}}, \underbrace{w_t}, \underbrace{w_{t+1}, \cdots, w_{t+m}}, \cdots, w_{T-1}, w_T$$

w_I w_O context window

$$p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} \mid w_I) = \prod_{c=1}^{C} p(w_{O,c} \mid w_I)$$

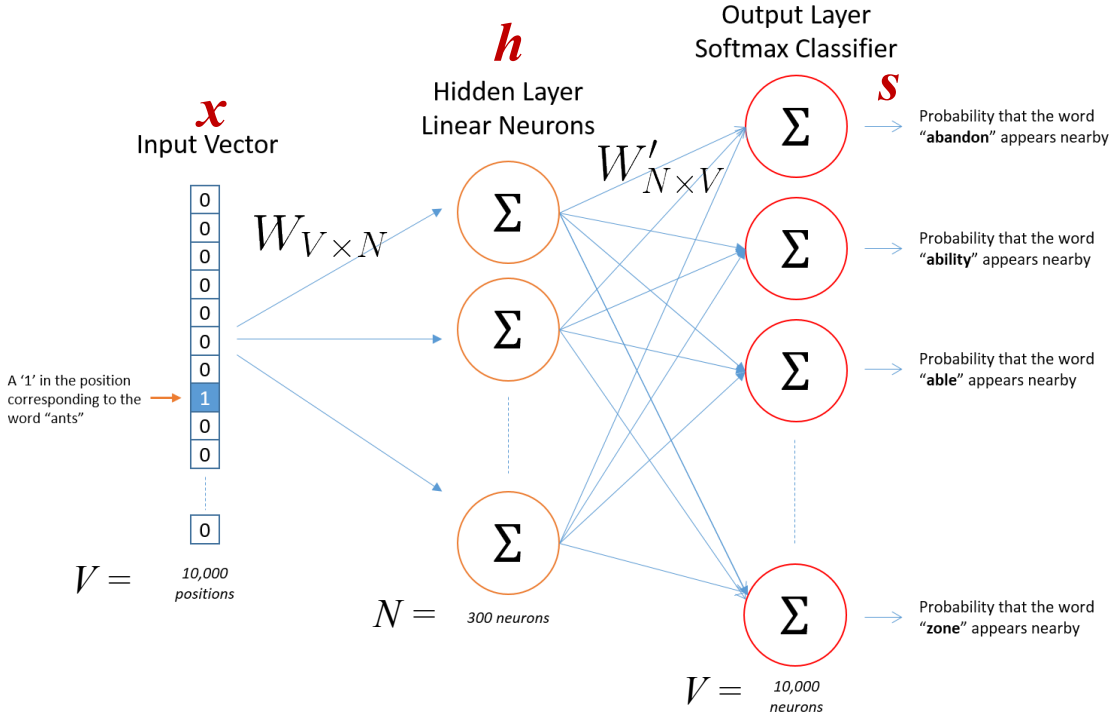
$$C(\theta) = -\sum_{w_I} \sum_{c=1}^{C} \log p(w_{O,c} \mid w_I)$$

$$p(w_O \mid w_I) = \frac{\exp(v'^T_{w_O} v_{w_I})}{\sum_j \exp(v'^T_{w_j} v_{w_I})}$$

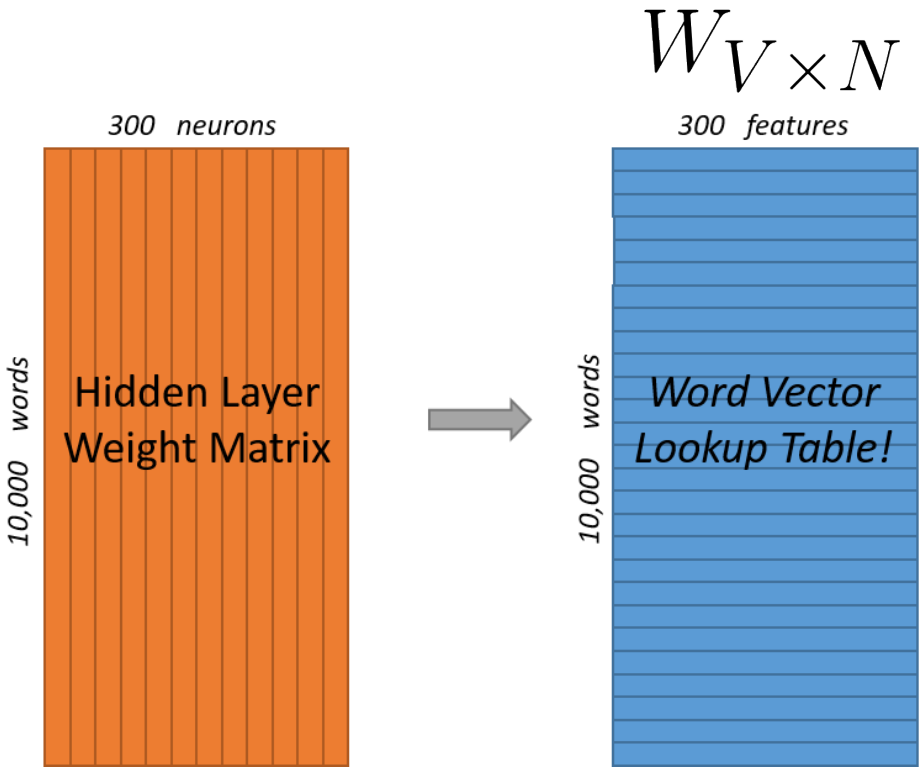
Benefit: fast, easy to incorporate a new sentence/document or add a word to vocab

Word2Vec Skip-Gram Illustration

- Goal: predict surrounding words within a window of each word



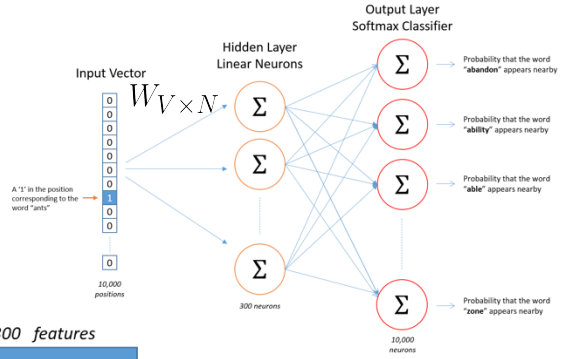
Hidden Layer Weight Matrix → Word Embedding Matrix



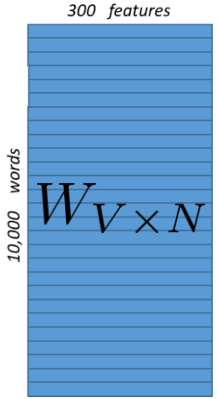
Weight Matrix Relation

- Hidden layer weight matrix = word vector lookup

$$h = x^T W = W_{(k,.)} := v_{w_I}$$



$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

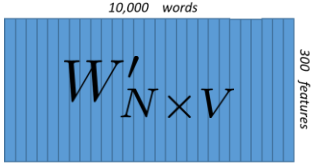


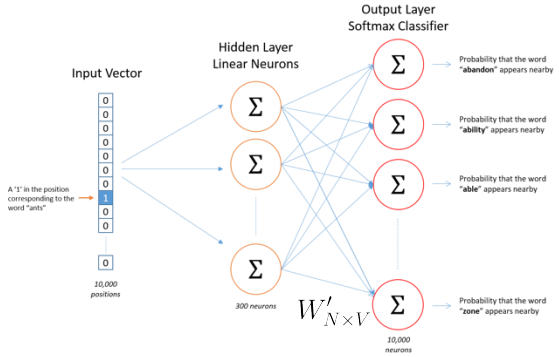
Each vocabulary entry has two vectors: as a **target** word and as a **context** word

Weight Matrix Relation

- Output layer weight matrix = weighted sum as final score

$$s_j = hv'_{w_j}$$



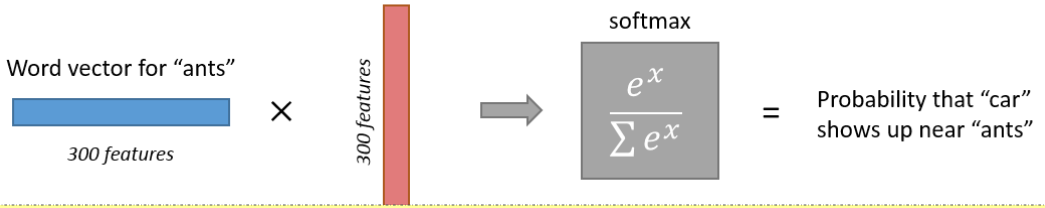


$$p(w_j = w_{O,c} \mid w_I) = y_{j_c} = \frac{\exp(s_{j_c})}{\sum_{j'=1}^{V} \exp(s_{j'})}$$

softmax

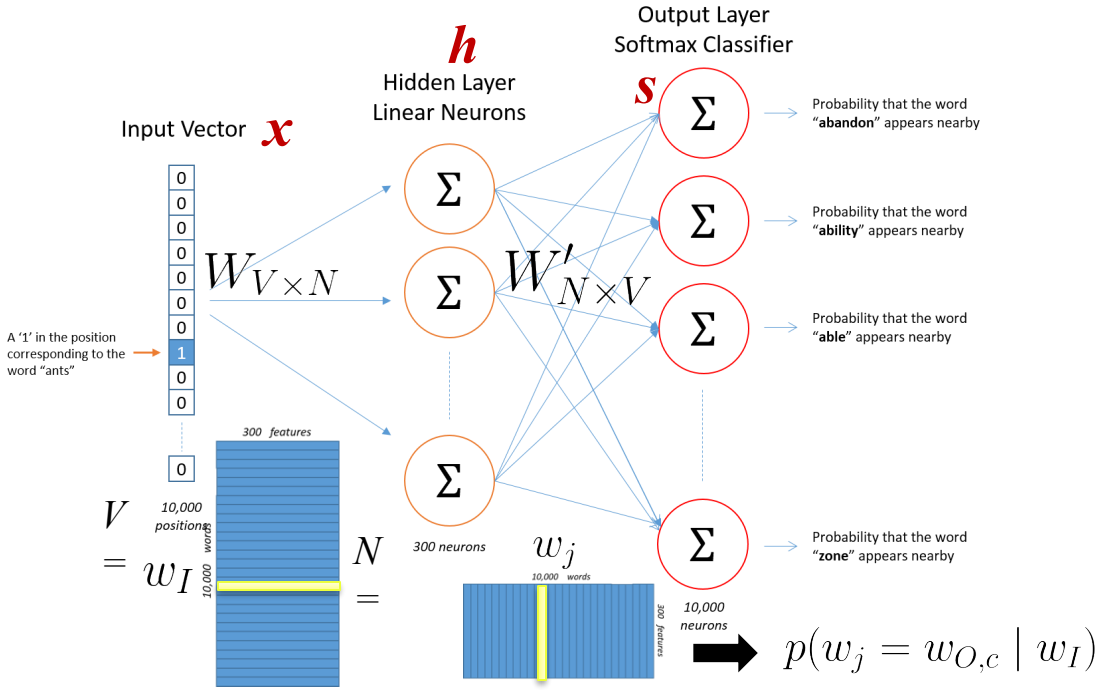
within the context window

Output weights for "car"



Each vocabulary entry has two vectors: as a **target** word and as a **context** word

Word2Vec Skip-Gram Illustration



Skip-Gram Training

- Training sentence:
- ... lemon, a tablespoon of **apricot** preserves or pinch ...
- c1 c2 t c3 c4

positive examples +	
t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

- For each positive example, create *k* negative examples.
- Using *noise* words
- Any random word that isn't *t*

Skip-Gram Training (cont.)

- Training sentence:
- ... lemon, a tablespoon of **apricot** preserves or pinch ...
- c1 c2 t c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

negative examples -

t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever

Negative Sampling

- Could pick w according to their unigram frequency P(w)
- More common to chose them according to $p_\alpha(w)$

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$$

- $\alpha = ¾$ works well because it gives rare noise words slightly higher probability
- To show this, imagine two events p(a)=.99 and p(b) = .01:

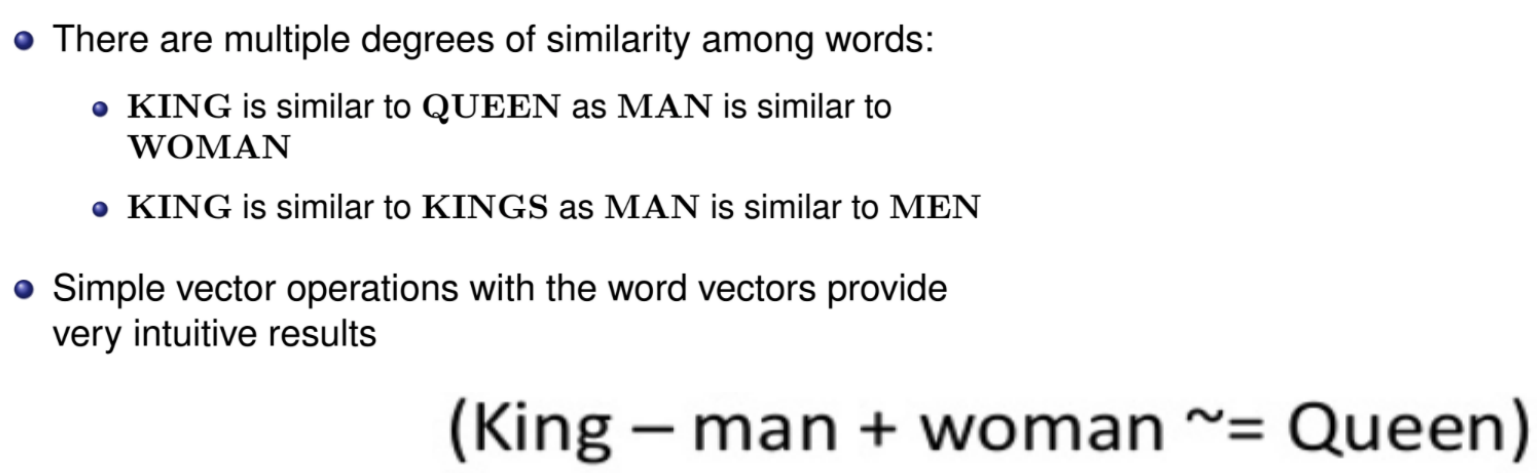
$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$

$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

Idea: less frequent words sampled more often.

What are Word Embeddings Good For?

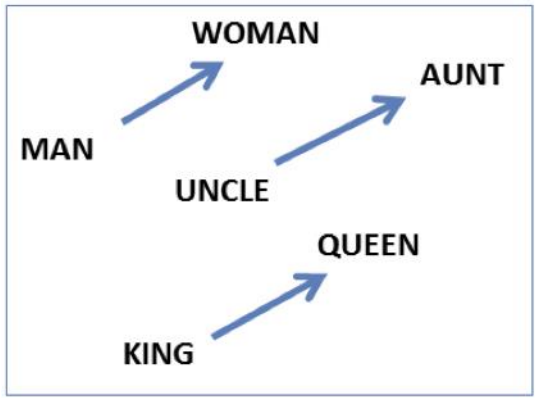
- Relations between words, captured by vector operations
- Lots of online lectures/tutorial
- https://www.slideshare.net/mlprague/tom-mikolov-distributed-representations-for-nlp

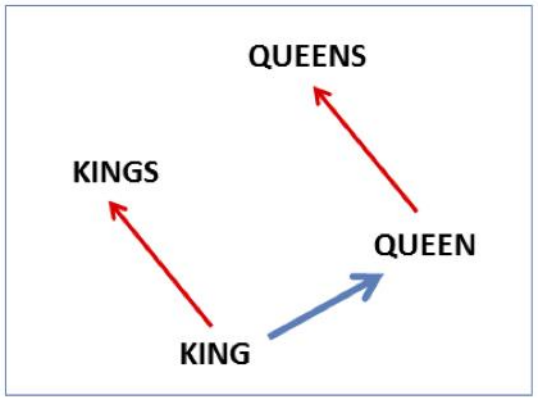


What are Word Embeddings Good For?

Linguistic Regularities: Distance in Vector Space

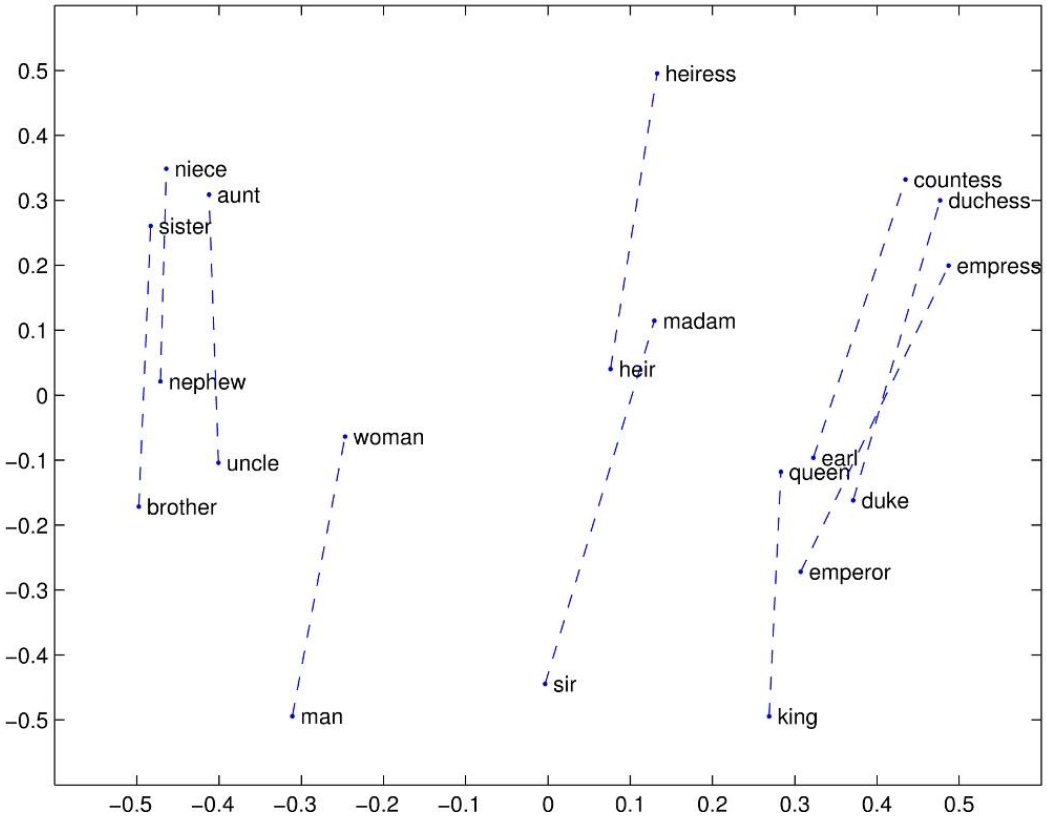
vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')
vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')



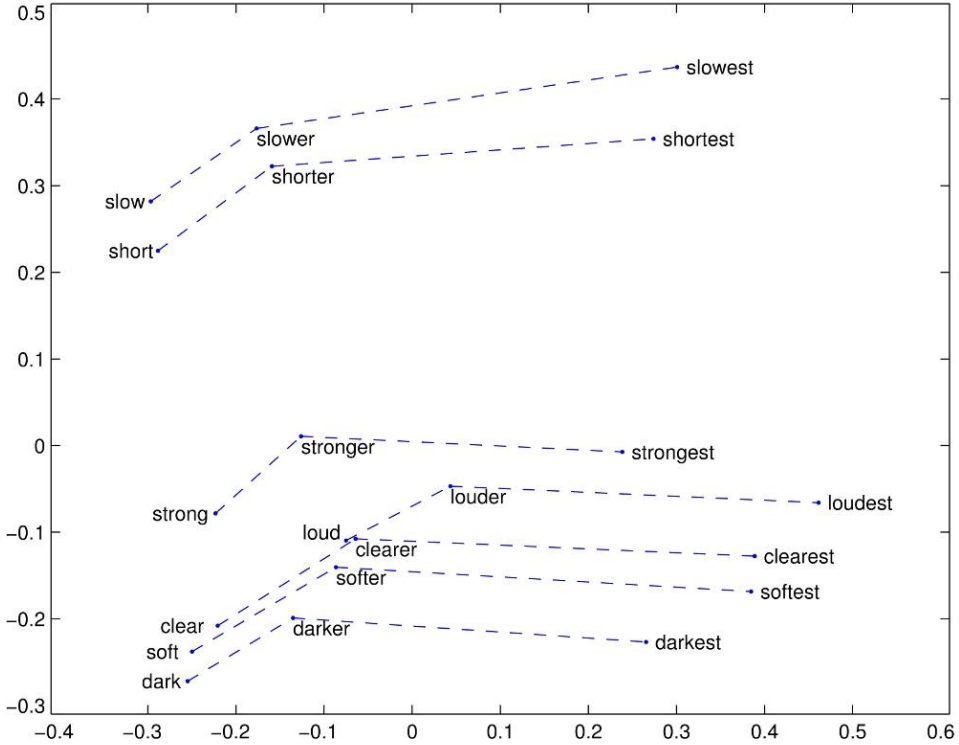


From Presentation by Mikolov

What are Word Embeddings Good For?



What are Word Embeddings Good For?



What are Word Embeddings Good For?

Linguistic Regularities: use the same vector

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

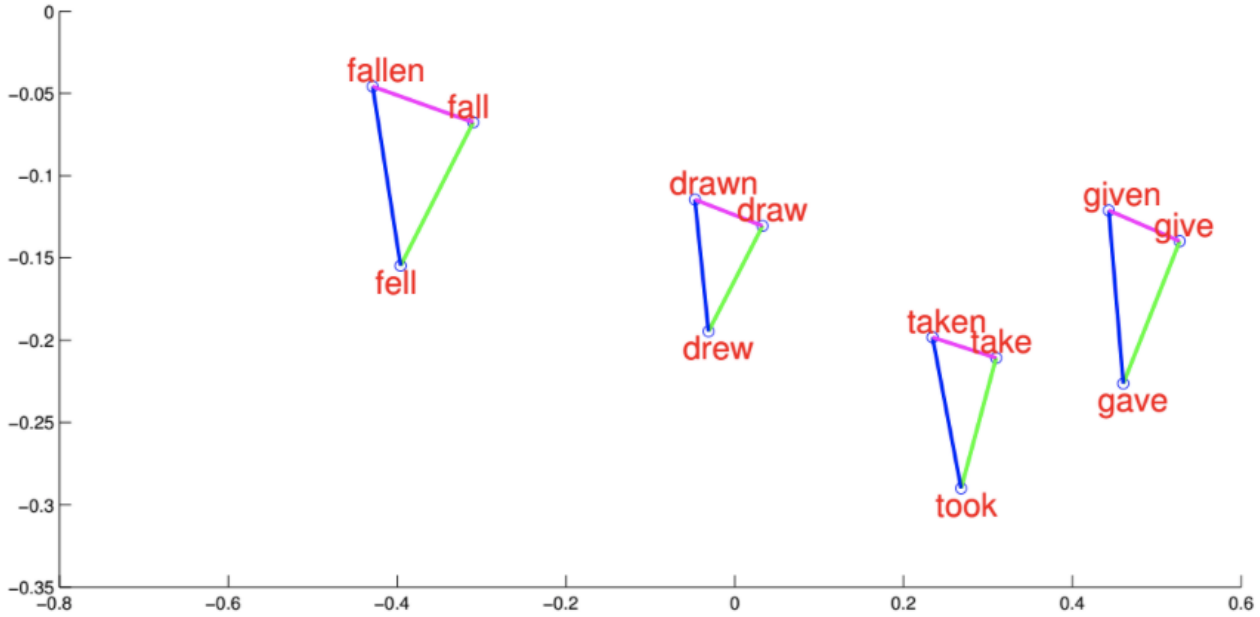
From Presentation by Mikolov

Vector Addition on Word Embeddings

Expression	*Nearest tokens*
Czech + currency	koruna, Czech crown, Polish zloty, CTK
Vietnam + capital	Hanoi, Ho Chi Minh City, Viet Nam, Vietnamese
German + airlines	airline Lufthansa, carrier Lufthansa, flag carrier Lufthansa
Russian + river	Moscow, Volga River, upriver, Russia
French + actress	Juliette Binoche, Vanessa Paradis, Charlotte Gainsbourg

From Presentation by Mikolov

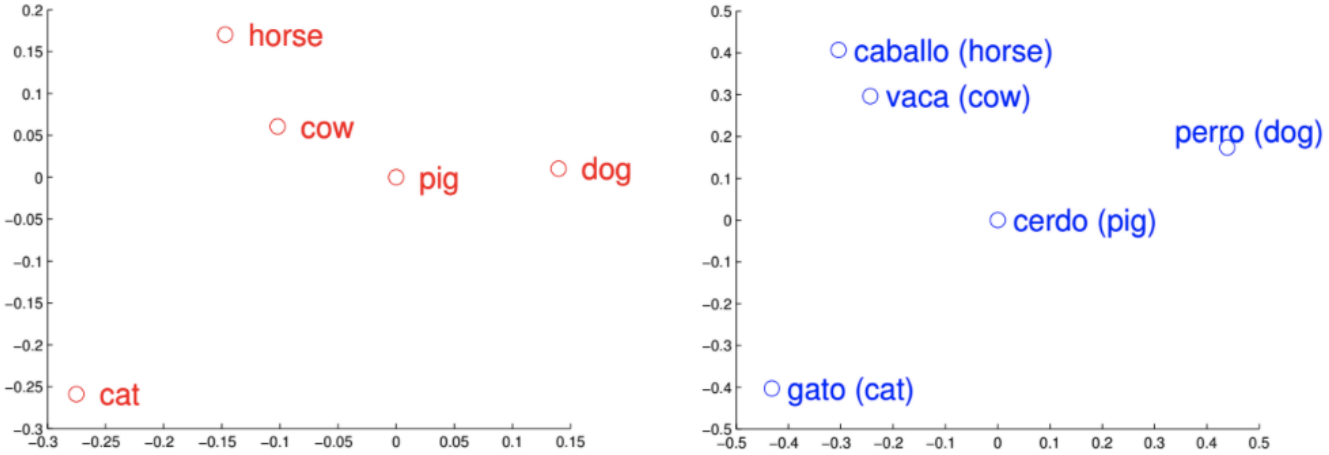
Visualization in the Vector Space



Applications to Machine Translation

- Typical Challenge in MT
  - large amounts of monolingual data for source and target language
  - Small amount of bilingual data
- Word Vectors have similar structures in two languages
- Need to learn projection from one language to another
- Can then translate any word seen in the monolingual data

Applications to Machine Translation



Similar idea works for other NLP tasks and transferring models for NLP tasks too. For example, spoken language understanding (Upadhyay et al, IEEE ICASSP, 2018).

Evaluating Embeddings – Intrinsic Evaluation

- Compare to human scores on word similarity-type tasks:
  - WordSim-353 (Finkelstein et al., 2002)
  - SimLex-999 (Hill et al., 2015)
  - Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
  - TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

Evaluating Embeddings – Extrinsic Evaluation, Subsequent Task

- Goal: use word vectors in neural net models built for subsequent tasks
- Benefit
  - Ability to also classify words accurately
    - Ex. countries cluster together a classifying location words should be possible with word vectors
  - Incorporate any information into them other tasks
    - Ex. project sentiment into words to find most positive/negative words in corpus

Other Word Embeddings – GloVe

- Global Vectors for Word Representation (GloVe)

- Methods like skip-gram do good on the analogy task, but…
- They poorly utilize the statistics of the corpus, since they train on separate local context windows instead of on global co-occurrence counts.

- A specific weighted least squares model that trains on global word-word co-occurrence counts and thus makes efficient use of statistics.

- GloVe learns based on a co-occurrence matrix and trains word vectors so their differences predict co-occurrence ratios.

- Source code for the model as well as trained word vectors at https://nlp.stanford.edu/projects/glove/

- Pennington et al., 2014
- Newer Version: Carlson et al., 2025

Other Word Embeddings – GloVe (cont.)

- Idea: ratio of co-occurrence probability can encode meaning
- Co-occurrence Probability, P_{ij} is the probability that word w_j appears in the context of word w_i

$$P_{ij} = P(w_j \mid w_i) = X_{ij}/X_i$$

- Relationship between the words w_i and w_j

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

The relationship of w_i and w_j can be examined by studying the ratio of their co-occurrence probabilities with various probe words

Other Word Embeddings – GloVe (cont.)

- The model enforces this property by learning embeddings $w_i, \tilde{w}_j$ such that:

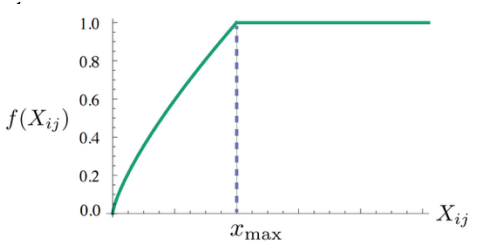
$$w_i^T \tilde{w}_j + b_i + \tilde{b}_j \approx \log(X_{ij})$$

- New, weighted least squares regression:

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

Weighting function

$$P_{ij} = X_{ij}/X_i$$



- f(x) is a weighting function that down-weights very rare and very frequent co-occurrences (so common words like “the” don’t dominate).

GloVe vs. Word2Vec

- Word2Vec uses local context (a sliding window over a corpus), whereas GloVe uses global co-occurrence statistics.

- GloVe needs big co-occurrence matrix upfront

Topics for Today

- Distributional Similarity
- Sparse Word Representations
- Word Embeddings and Word2Vec
- Language Modeling
- Unexpected things we learn with word embeddings

Language Modeling

- Goal: estimate the probability of a word sequence

$$P(w_1, \cdots, w_m)$$

- Example task: determine whether a sequence is grammatical or makes more sense

recognize speech
or
wreck a nice
beach

If P(recognize speech)
> P(wreck a nice beach)

Output = “recognize speech”

N-gram Language Modeling

- Goal: estimate the probability of a word sequence

$$P(w_1, \cdots, w_m)$$

- N-gram language model
  - Probability is conditioned on a window of $(n\text{-}1)$ previous words

$$P(w_1, \cdots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \cdots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \cdots, w_{i-1})$$

  - Estimate the probability based on the training data

$$P(\text{beach}|\text{nice}) = \frac{C(nice\ beach)}{C(nice)}$$

Count of "nice beach" in the training data

Count of "nice" in the training data

Issue: some sequences may not appear in the training data

N-gram Language Modeling (cont.)

- Training data:
  - The dog ran ……
  - The cat jumped ……

P(jumped | dog) = ~~0~~ 0.0001

P(ran | cat) = ~~0~~ 0.0001

give some small probability
→ smoothing

- ➢ The probability is not accurate.
- ➢ The phenomenon happens because we cannot collect all the possible text in the world as training data.

Language Modeling – Laplace (Add-1) Smoothing

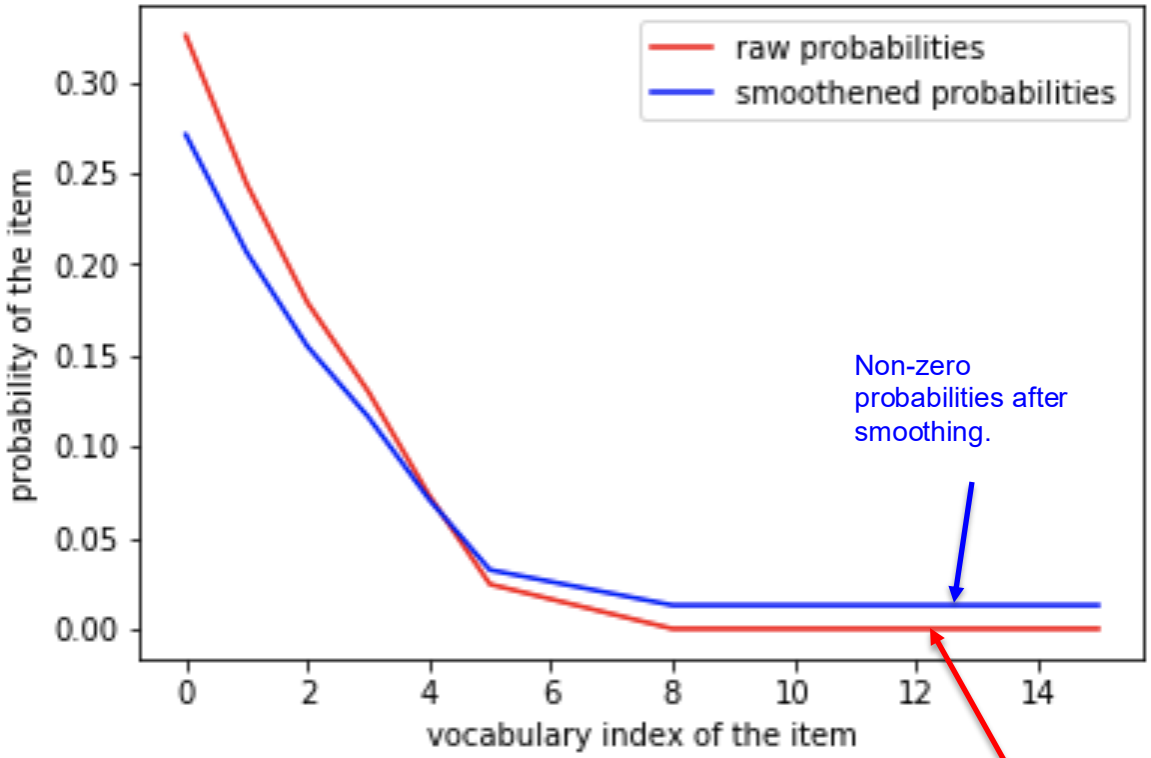
$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

Laplace (add-1) Smoothing Example

- Corpus of 1K tokens, |V| = 20, C(the) = 100 C(orange) = 14, C(apple) = 0
- Before smoothing:
  - P(the) = 100 / 1000 = 0.1
  - P(orange) = 14/1000 = 0.014
  - P(apple) = 0/1000 = 0
- After smoothing:
  - P(the) = (100 + 1) / (1000 + 20) = 0.099
  - P(orange) = 14 + 1 / (1000 + 20) = 0.0147
  - P(apple) = (0 + 1) / (1000 + 20) = 0.00098

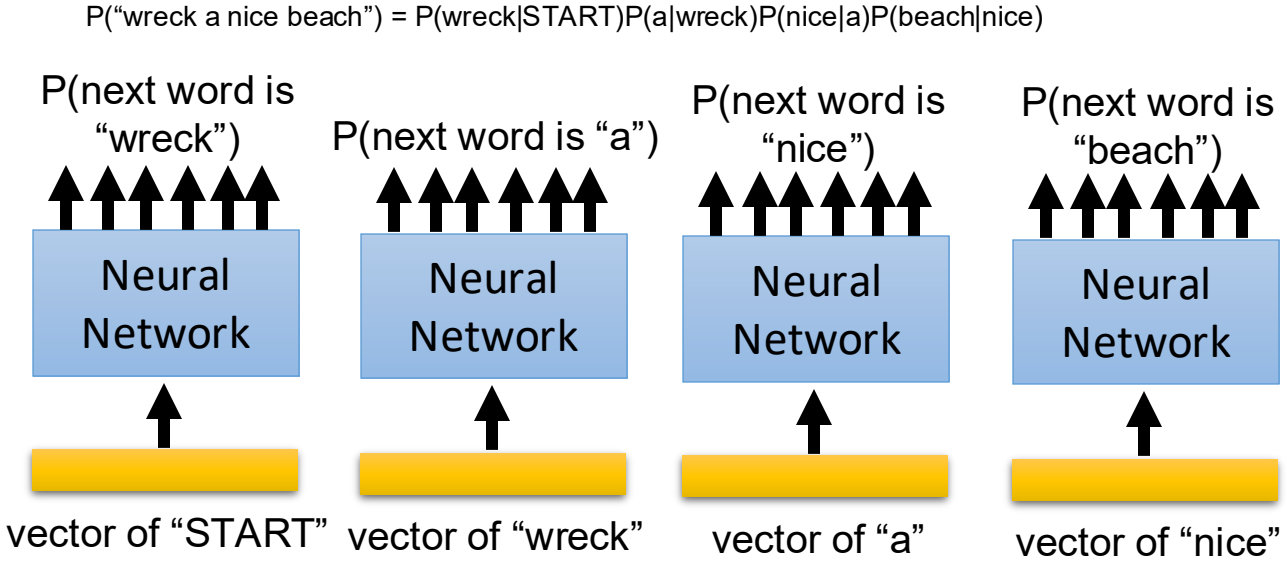
Laplace (add-1) Smoothing Depiction



* Items (samples) are sorted in order of decreasing probability

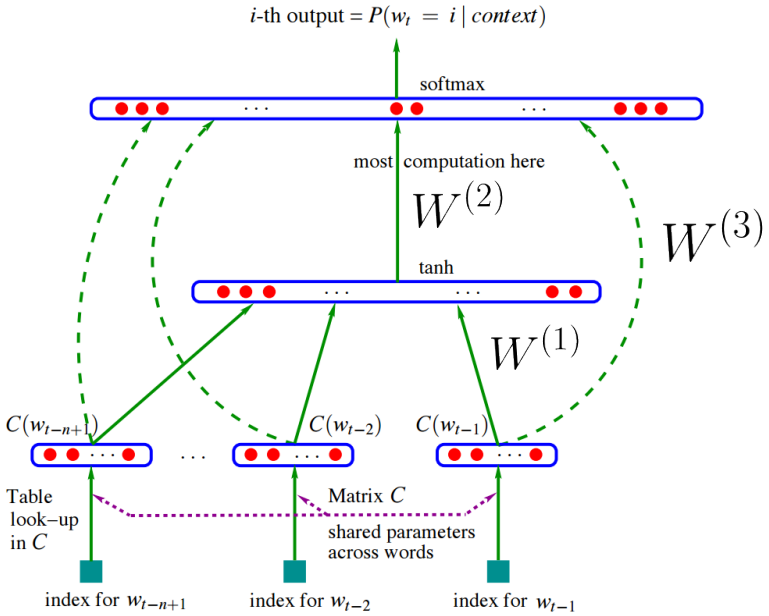
Neural Language Modeling

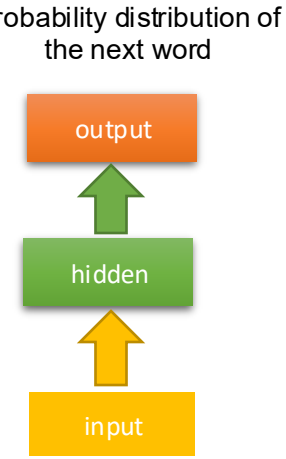
- Idea: estimate $P(w_i \mid w_{i-(n-1)}, \cdots, w_{i-1})$ not from counts, but based on the NN prediction



Neural Language Modeling (cont.)

$$\hat{y} = \text{softmax}(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + W^{(3)}x + b^{(3)})$$

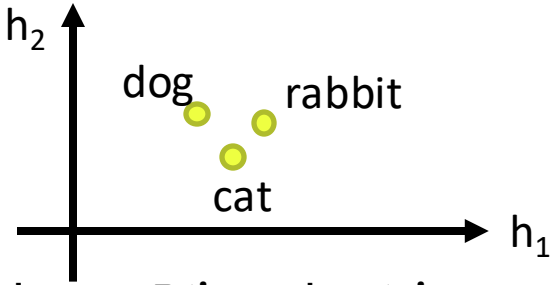




Bengio et al., "A Neural Probabilistic Language Model," in *JMLR*, 2003.

Neural Language Modeling (cont.)

- The input layer (or hidden layer) of the related words are close



  - If P(jump|dog) is large, P(jump|cat) increases accordingly (even there is no "… cat jump …" in the data)

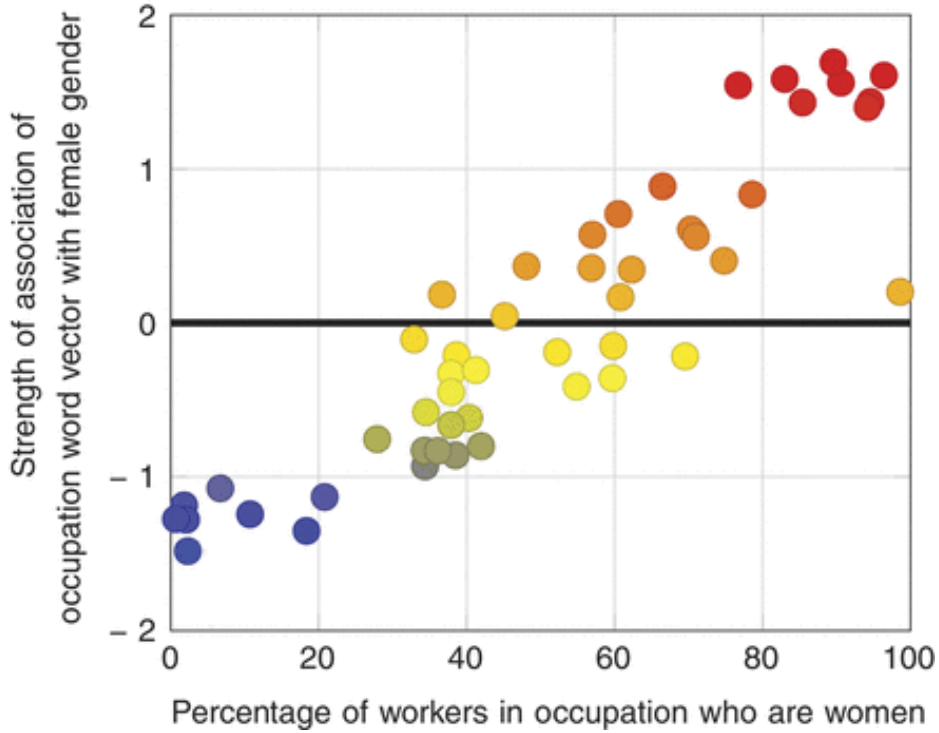
Smoothing is automatically done

Issue: fixed context window for conditioning

Topics for Today

- Distributional Similarity
- Sparse Word Representations
- Word Embeddings and Word2Vec
- Language Modeling
- Unexpected things we learn with word embeddings

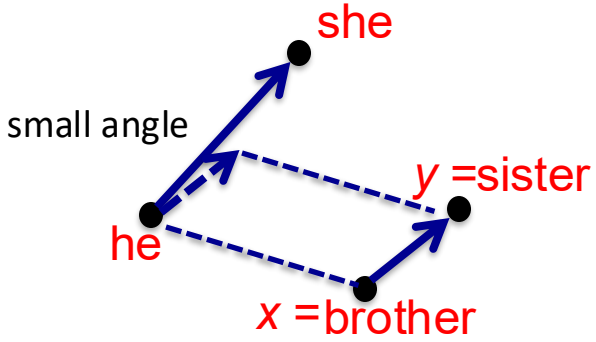
Occupation word vectors: job titles & gender



Semantics derived automatically from language corpora contain human-like biases, Aylin Caliskan, Joanna J. Bryson, Arvind Narayanan, Science 2017.

Gender vectors in word embeddings: analogies

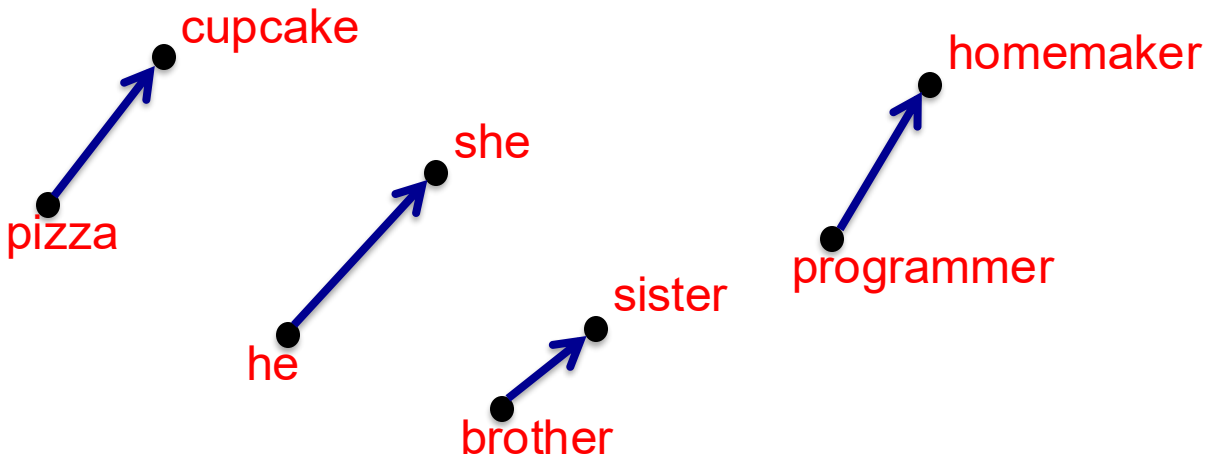
Automatically generate he : *x* :: she : *y* analogies.



$$\min \cos(\text{he} - \text{she}, x - y) \text{ such that } ||x - y||_2 < \delta$$

Gender **stereotypes** in word embeddings: analogies

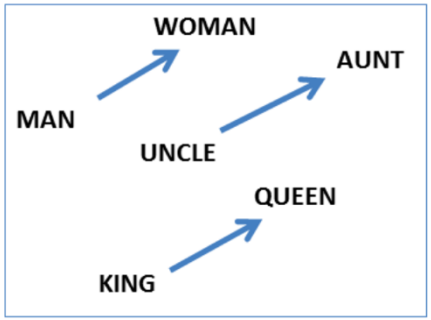
Automatically generate he : *x* :: she : *y* analogies.



$$\min \cos(\text{he} - \text{she}, x - y) \text{ such that } ||x - y||_2 < \delta$$

Word Embeddings can be Dreadfully Sexist

- $v_{man} - v_{woman} + v_{uncle} \sim v_{aunt}$



he: _____	she:_____
brother	sister
barbershop	salon
carpentry	sewing
bartender	hostess
physician	registered_nurse
professor	associate professor

Google w2v embedding trained from the news

They can be also Racist

Typical Caucasian names	Typical African American names
Molly	Aisha
Amy	Keisha
Jake	Leroy
Luke	Jermaine
singer	rapper
lobster	shrimp
geeks	dudes
hockey	basketball
urban	slums

Kai-Wei Chang (kwchang.net/talks/sp.html)

Gender stereotypes in word embeddings: Analogies

19% of the top 150 analogies were rated as gender stereotypical by a majority of crowdworkers

Before executing debiasing		
Analogy	Appropriate	Biased
midwife:doctor	1	10
sewing:carpentry	2	9
pediatrician:orthopedic_surgeon	0	9
registered_nurse:physician	1	9
housewife:shopkeeper	1	9

Topics for Next Week

Tuesday:

- Convolutional Neural Networks
- Recurrent Neural Networks

Thursday:

- Long-Short Term Memory (LSTM)
- Gated Recurrent Units (GRU)
- Example Sequence Classification Tasks
- Elmo and Contextual Embeddings

Newcomers

- Previous lectures are not recorded, but previous slides are under Modules in Canvas

- The homework timeline is on the main Canvas page for the course

- Immediate thing to act on:
  - Form a final project team!