

CS 2731 / ISSP 2230

# Introduction to Natural Language Processing

Session 8: Vector semantics, static word embeddings

---

Michael Miller Yoder

February 5, 2024



School of Computing and Information

# Course logistics

- [Project pre-proposal form](#) is **due today, Mon Feb 5**
  - Please plan meeting with your groups to discuss project ideas
  - If you don't have any specific ideas, that's fine! We will help you come up with some.
    - There is an opportunity to work with a Pitt Bioengineering prof on predicting how safe shoes are for restaurant workers (if they slip) based on Amazon reviews
  - Submit 1 form per group through Google Forms. No need to submit anything on Canvas

# Course logistics

- Homework 1 grades will be out by Feb 12 at the latest
- [Homework 2](#) is due next **Thu Feb 15**
  - Text classification
  - Written and programming components
  - Optional Kaggle competition for best LR and NN politeness classifiers

# Lecture overview: vector semantics, static word embeddings

- Vector semantics
- Distributional semantics
- Types of word vectors
- Word2vec
- Bias in word vectors

# A brief history of NLP



Sentences in Russian are punched into standard cards for feeding into the electronic data processing machine for translation into English

*The Georgetown-IBM Experiment.  
Credit: John Hutchins*

- 1950s: **foundations**
  - Turing Test ("Computing Machinery and Intelligence" paper)
  - Georgetown-IBM Experiment translating Russian to English
- 1960s-1980s: **symbolic reasoning**
  - ELIZA, rule-based parsing, hand-built conceptual ontologies
- 1990s-2010s: **statistical NLP**
  - Learn patterns from large corpora (feature-based machine learning)
- 2000s-today: **neural NLP**
  - SOTA on many tasks from "deep" layers of neural networks

# Vector semantics

---

# Semantics: the study of meaning

- Word representations in NLP draw on 2 areas of semantics
  - a. Vector semantics
  - b. Distributional semantics

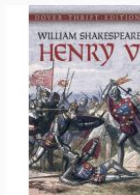
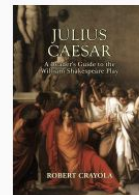
# Vector semantics

- Modeling semantics as points in vector space
  - Words or other text segments are represented by vectors
  - Multiple dimensions
  - Nearer = more similar words



# Term-document matrix: word vectors

Two words are similar if their vectors are similar.



	As You Like It	Twelfth Night	Julius Caesar	Henry V
<i>battle</i>	1	1	8	15
<i>soldier</i>	2	2	12	36
<i>fool</i>	37	58	1	5
<i>clown</i>	6	117	0	0

Pairs of similar words?

# Similarity and relatedness

- Synonyms: big/large, couch/sofa, automobile/car
- Similar: sharing some element of meaning
  - coffee/tea, car/bicycle, cow/horse
- Related: by a semantic field
  - coffee/cup, scalpel/surgeon



# Distributional semantics

---

# Distributional semantics: roots in anthropological linguistics

- In the early 20th century, many native languages of the Americas were dying due to the destruction of European colonization
- A group of American anthropologists (Boas, Sapir, Bloomfield, etc.) decided that they needed to describe all of these languages (produce grammars, dictionaries, and texts for them) before they were gone
- Earlier scholars who studied American languages tried to shoehorn them into the grammatical structure of European languages, but this group of researchers saw that they were very different from one another and from, e.g., Latin
- **They wanted to describe languages on their own terms**
- They developed techniques (in some cases, algorithms) for discovering meaning and grammatical structure without making reference to other languages
  - “To pass from one language to another is psychologically parallel from one geometrical system of reference to another.” (Sapir 1924)



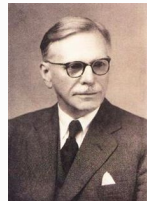
Edward Sapir

# Distributional semantics

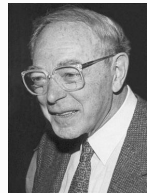
"The meaning of a word is its use in the language"  
[Wittgenstein 1953]



"You shall know a word by the company it keeps" [Firth 1957]



"If A and B have almost identical environments we say that they are synonyms" [Harris 1954]



# Distributional semantics

Define the meaning of a word by its **distribution in language use**: its neighboring words or grammatical environments.

# You Can Tell a Lot about *Beef* from Its Contexts

1 fertility. Organ meats such as beef and chicken liver, tongue and hear  
2 controlling scours. HOW TO FEED: BEEF AND DAIRY CALVES - 0.2 gram Dy  
3 ing process discolors the treated beef and liquid accumulates in prepackag  
4 say. He did say she could get her beef and vegetables in cans this summer  
5 and feed efficiency of fattening beef animals. HOW TO FEED: At the  
6 steaks, chops, chicken and prime beef as well as Tom's favorite dish, stu  
7 ross from him was surmounted by a beef barrel with ends knocked out. In t  
8 counter of boards laid across two beef barrels. There was, of course, no  
9 Because Holstein cattle weren't a beef breed, they were rarely seen on a  
10 2-5 grams of phenothiazine daily; beef calves- .5 to 1.5 grams daily depe  
11 ties of this drug. HOW TO FEED: BEEF CATTLE (FINISHING RATION) - To  
12 dairy cows and lesser amounts to beef cattle and poultry. About 90 percen  
13 raises enough poultry, pigs, and beef cattle for most of their needs. Lo  
14 on of liver abscesses in feed-lot beef cattle. Prevention of bacterial pne  
15 pal feed bunk types for dairy and beef cattle: (1) Fence-line bunks- catt  
16 es feed efficiency. HOW TO FEED: BEEF CATTLE - 10 milligrams of diet  
17 the rations you are feeding your beef, dairy cattle, and sheep are adequa  
18 itive business more profitable for beef, dairy, and sheep men. The tar  
19 o bear. She was ready to kill the beef, dress it out, and with vegetables  
20 . She had raised a calf, grown it beef-fat. She had, with her own work-wea  
21 with feeding low-moisture corn in beef-feeding programs. Several firms ar  
22 he shelf life (at 35 F) of fresh beef from 5 days to 5 or 6 weeks. Howeve  
23 canned pork products. Tests with beef have been largely unsuccessful beca  
24 for eggs, pigs to eat garbage, a beef herd and wastes of all kinds. Separ  
25 their money's worth. A good many beef-hungry settlers were accepting the



## Contexts for *Chicken* Are also Informative

1 y the irradiated and refrigerated [chicken](#). Acceptance of radiopasteurization  
2 torehouse". Glendora dropped a [chicken](#) and a flurry of feathers, and went  
3 will specialize in steaks, chops, [chicken](#) and prime beef as well as Tom's fa  
4 ard as the one concerned with the [chicken](#) and the egg. Which came first? Is  
5 he millions of buffalo and prairie [chicken](#) and the endless seas of grass that  
6 "! "Come on, there's some cold [chicken](#) and we'll see what else". They wen  
7 ves to extend the storage life of [chicken](#) at a low cost of about 0.5 cent per  
8 CHICKEN CADILLAC# Use one 6-ounce [chicken](#) breast for each guest. Salt and pe  
9 ion juice, to about half cover the [chicken](#) breasts. Bake slowly at least one-  
10 d, in butter. Sprinkle over top of [chicken](#) breasts. Serve each breast on a th  
11 around, they had a hard time". #CHICKEN CADILLAC# Use one 6-ounce chicken  
12 successful, and the shelf life of [chicken](#) can be extended to a month or more  
13 ay from making a cake, building a [chicken](#) coop, or producing a book, to found  
14 , they decided, but a deck full of [chicken](#) coops and pigpens was hardly suita  
15 im. "Johnny insisted on cooking a [chicken](#) dinner in my honor- he's always bee  
16 nutes. Kid Ory, the trombonist [chicken](#) farmer, is also one of the solid a  
17 y Johnson reaching around the wire [chicken](#) fencing, which half covered the tr  
18 yes glittering behind dull silver [chicken](#) fencing. "That was Tee-wah I was t  
19 wine in the pot roast or that the [chicken](#) had been marinated in brandy, and  
20 yed this same game and called it "[Chicken](#)". He could not go through the f  
21 f the Mexicans hiding in a little [chicken](#) house had passed through his head,  
22 I'll never forget him cleaning the [chicken](#) in the tub". A story, no doubt  
23 . Organ meats such as beef and [chicken](#) liver, tongue and heart are planne  
24 p. "Miss Sarah, I can't cut up no [chicken](#). Miss Maude say she won't". Aga

# You Learn Words by Using Distributional Similarity



Consider

- A bottle of **pocarisweat** is on the table.
- Everybody likes **pocarisweat**.
- **Pocarisweat** makes you feel refreshed.
- They make **pocarisweat** out of ginger.

What does *pocarisweat* mean?

# You Know Pocarisweat by the Company It Keeps



From context words humans can guess *pocarisweat* means a beverage like **coke**.

How do you know?

- Other words can occur in the same context
- Those other words are often for beverages (that you drink cold)
- You assume that *pocarisweat* is probably similar

So the intuition is that **two words are similar if they have similar word contexts**.

# Sample Contexts of $\pm 7$ Words

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

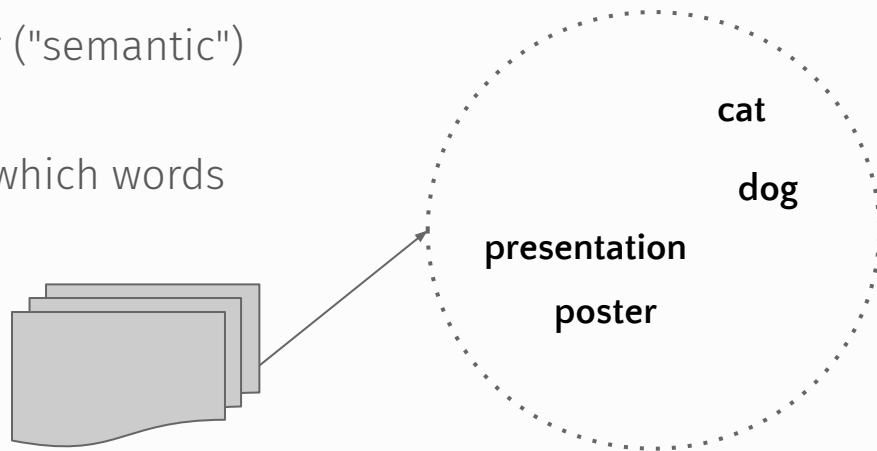
	aardvark	computer	data	pinch	result	sugar ...
⋮						
<i>apricot</i>	0	0	0	1	0	1
<i>pineapple</i>	0	0	0	1	0	1
<i>digital</i>	0	2	1	0	1	0
<i>information</i>	0	1	6	0	4	0
⋮						

# Types of word vectors

---

# Shared Intuition: Words are Vectors of Numbers Representing Meaning

- Model the meaning of a word by “**embedding**” it in a vector space.
- The meaning of a word is a vector of numbers:
  - Vector models are also called **embeddings**
  - Often, the word *embedding* is reserved for *dense* vector representations
- In contrast, word meaning is represented in many (early) NLP applications by a vocabulary index (“word number 545”; compare to **one-hot representations**)
- Similar words are nearby in vector ("semantic") space
- Build "semantic space" by seeing which words are nearby in text



# Why word embeddings?

- Can generalize to similar but unseen words

**cat** [0.31, 0.24, 0.07, 0.65 ... ]      **dog** [0.37, 0.29, 0.06, 0.63 ... ]

- Compute with meaning representations instead of string representations for words

荃者所以在鱼，得鱼而忘荃    Nets are for fish;  
Once you get the fish, you can forget the net.  
言者所以在意，得意而忘言    Words are for meaning;  
Once you get the meaning, you can forget the words  
庄子(Zhuangzi), Chapter 26

All modern NLP systems  
have embeddings as  
representations of word  
meaning



# There are Two Kinds of Vector Models

- **Sparse embeddings** (vectors from term-document matrix)
  - long (length of 20,000 to 50,000)
  - sparse: most elements are 0
- **Dense embeddings** (Word2vec)
  - short (length of 50-1000)
  - dense (most elements are non-zero)

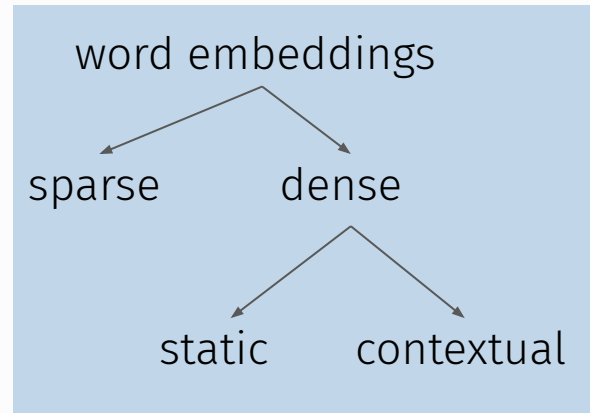


# Dense Vectors Have Three Advantages over Sparse Vectors

1. Short vectors may be **easier to use as features** in machine learning (less weights to tune).
2. Dense vectors may **generalize better** than storing explicit counts.
3. They may do **better at capturing synonymy**:
  - *car* and *automobile* are synonyms
  - But, in sparse vectors, they are represented as distinct dimensions
  - This fails to capture similarity between a word with *car* as a neighbor and a word with *automobile* as a neighbor

# Methods for learning short, dense word embeddings

- Static, neural embeddings
  - Fixed embeddings for word types
  - Word2Vec, GloVe
- Contextual embeddings
  - Embeddings for words vary by context
  - ELMo, BERT, LLMs



# Word2vec

---

# Word2vec [Mikolov et al. 2013]


- Instead of counting words, train a classifier on a binary prediction task
  - Is  $w_1$  likely to show up near  $w_2$ ?

# Word2vec [Mikolov et al. 2013]


- Instead of counting words, train a classifier on a binary prediction task
  - Is  $w_1$  likely to show up near *apricot*?



# Word2vec [Mikolov et al. 2013]

- Instead of counting words, train a classifier on a binary prediction task
  - Is  $w_1$  likely to show up near *apricot*? 
- Take the learned classifier weights as the word embeddings

# Word2vec [Mikolov et al. 2013]

- Instead of counting words, train a classifier on a binary prediction task
  - Is  $w_1$  likely to show up near *apricot*? 
- Take the learned classifier weights as the word embeddings
- Training techniques: skip-gram and CBOW



# Word2vec: training supervision

- **Self-supervision** [Bengio et al. 2003, Collobert et al. 2011]
- Use naturally occurring text as labels
- A word  $c$  that occurs near *apricot* in the corpus counts as the gold "correct answer" for supervised learning

# Word2vec training overview

1. Positive examples: the target word  $w$  and a neighboring context word  $c_{pos}$
2. Negative examples: Randomly sample other words  $c_{neg}$  in the lexicon to pair with  $w$
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the learned weights ( $W, C$ ) as the word embeddings

# Training for Embeddings

- We do not know what  $W$  and  $C$  are. So we learn them through an iterative process.
- We use a large corpus as a training data
- We also randomly sample the corpus to find words that are NOT in the context—negative sampling.



Positive Examples		Negative Examples			
t	c	t	c	t	c
ides	beware	ides	aardvark	ides	twelve
ides	of	ides	puddle	ides	hello
ides	March	ides	where	ides	dear
ides	the	ides	coaxial	ides	forever

# Word2vec: learning embeddings

- Start with randomly initialized context  $C$  and target word  $W$  matrices
- Go through the positive and negative training pairs, adjusting word vectors such that we:
  - Maximize the similarity of the target word, context word pairs  $(w, c_{pos})$  drawn from the positive data
  - Minimize the similarity of the  $(w, c_{neg})$  pairs drawn from the negative data.

# Skip-gram classifier

Classifier input pairs:

(target word  $w$ , context word  $c$ )

Classifier output: probabilities that  $w$  occurs with  $c$

$$P(+|w, c)$$

$$P(-|w, c) = 1 - P(+|w, c)$$

# Skip-gram classifier: calculating probabilities

- From input vectors, need to compare for similarity
- Start with dot product:  $\text{sim}(\mathbf{w}, \mathbf{c}) \approx \mathbf{w} \cdot \mathbf{c}$
- To turn this into a probability, use the sigmoid function from logistic regression:

$$P(+|\mathbf{w}, \mathbf{c}) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{c} \cdot \mathbf{w})}$$

# Skip-gram classifier: calculating probabilities

$$P(+|w, c) = \sigma(c \cdot w) = \frac{1}{1 + \exp(-c \cdot w)}$$

This is for one context word, but we have lots of context words. We'll assume independence and just multiply them:

$$P(+|w, c_{1:L}) = \prod_{i=1}^L \sigma(c_i \cdot w)$$
$$\log P(+|w, c_{1:L}) = \sum_{i=1}^L \log \sigma(c_i \cdot w)$$

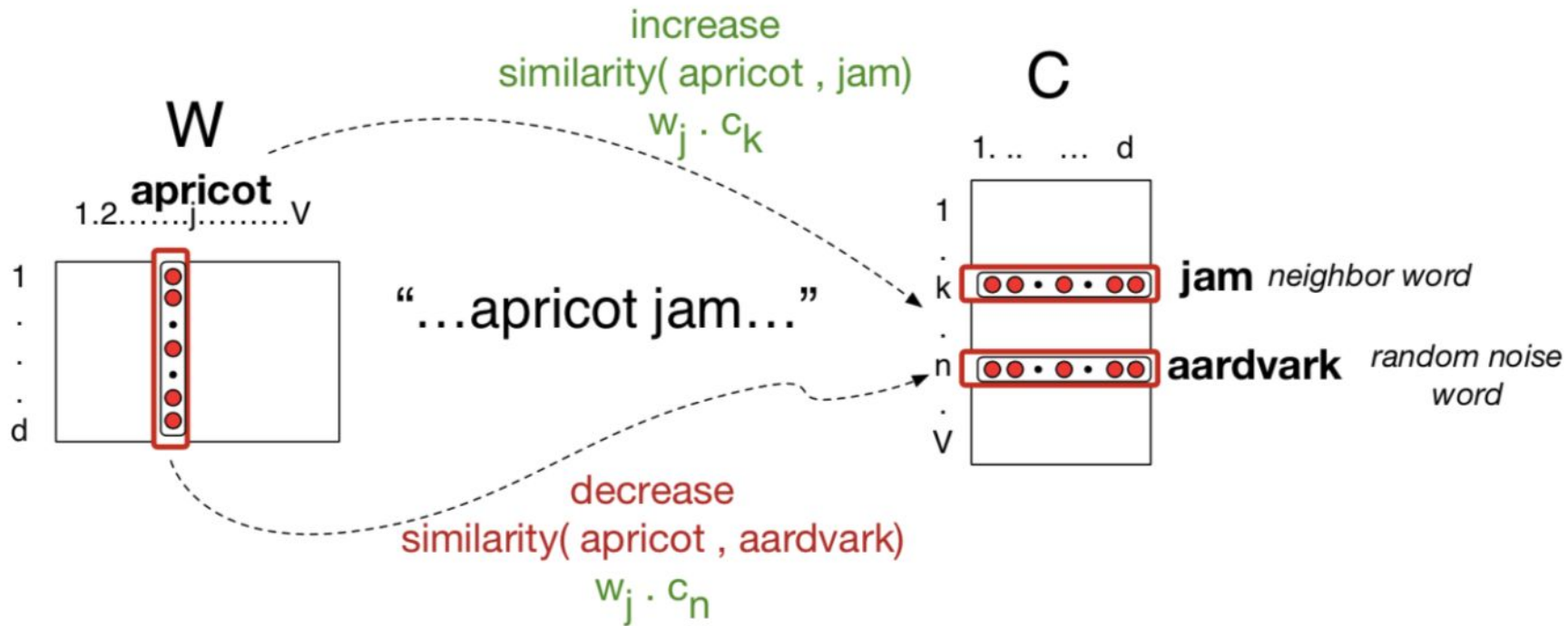
# Loss function for one $w$ with $c_{pos}$ , $c_{neg1}$ ... $c_{negk}$

Maximize the similarity of the target with the actual context words, and minimize the similarity of the target with the  $k$  negative sampled non-neighbor words.

$$\begin{aligned} L_{CE} &= -\log \left[ P(+|w, c_{pos}) \prod_{i=1}^k P(-|w, c_{neg_i}) \right] \\ &= - \left[ \log P(+|w, c_{pos}) + \sum_{i=1}^k \log P(-|w, c_{neg_i}) \right] \\ &= - \left[ \log P(+|w, c_{pos}) + \sum_{i=1}^k \log (1 - P(+|w, c_{neg_i})) \right] \\ &= - \left[ \log \sigma(c_{pos} \cdot w) + \sum_{i=1}^k \log \sigma(-c_{neg_i} \cdot w) \right] \end{aligned}$$



# Training for Embeddings



# Reminder: one step of gradient descent

- Direction: We move in the reverse direction from the gradient of the loss function
- Magnitude: we move the value of this gradient  $d/dw L(P(+|w,c) + P(-|w,c))$  weighted by a learning rate  $\eta$
- Higher learning rate means move  $w$  faster

# Word2vec training process

Updates on C and W

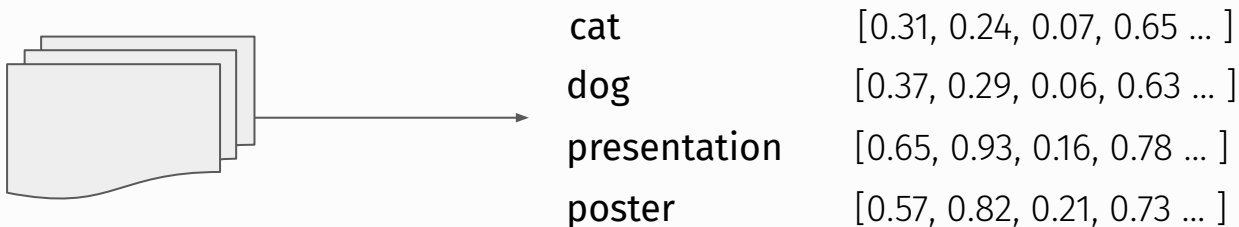
$$c_{pos}^{t+1} = c_{pos}^t - \eta [\sigma(c_{pos}^t \cdot w^t) - 1]$$

new context weights    old context weights    learning rate    derivative of loss wrt  $c_{pos}$

$$w^{t+1} = w^t - \eta \left[ [\sigma(c_{pos} \cdot w^t) - 1] c_{pos} + [\sigma(c_{neg_i} \cdot w^t)] c_{neg_i} \right]$$

new target word weights    old context weights    learning rate    derivative of loss wrt  $w$

# Summary: How to learn word2vec embeddings

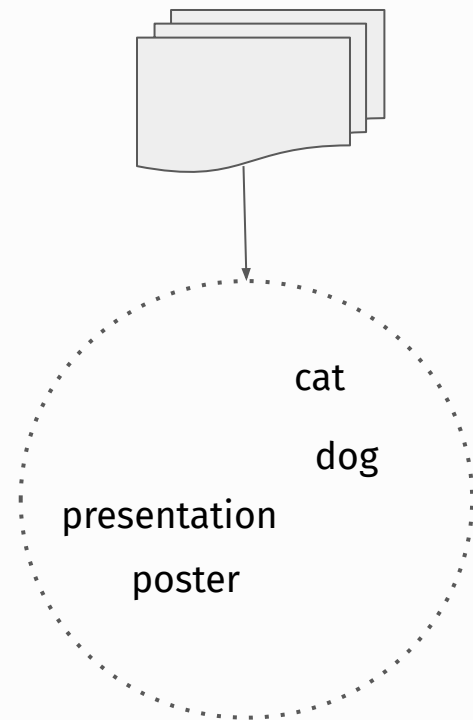


# Summary: How to learn word2vec embeddings

1. Start with randomly initialized word embeddings
2. From a corpus, extract pairs of words that co-occur (positive)
3. Extract pairs of words that don't co-occur (negative)
4. Train a classifier to distinguish between positive and negative examples by slowly adjusting all the embeddings to improve the classifier performance
5. Keep the weights as our word embeddings

# Final embeddings

- Can add representations for a word in  $W$  and in  $C$  together for final word vector for  $W_i$
- Can just keep  $W$  and throw away  $C$
- Can find "nearest neighbors" of certain words with cosine similarity in embedding space



# There are Tools and Resources Available for Training and Using Embeddings

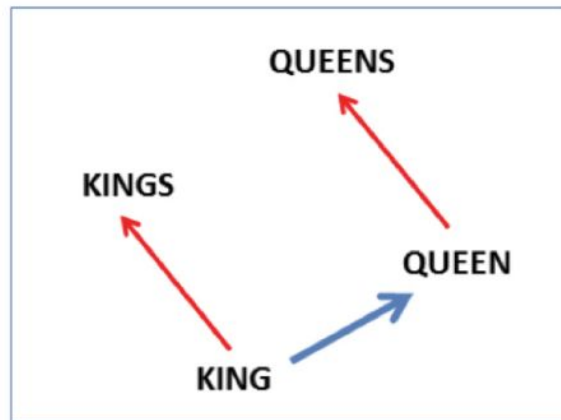
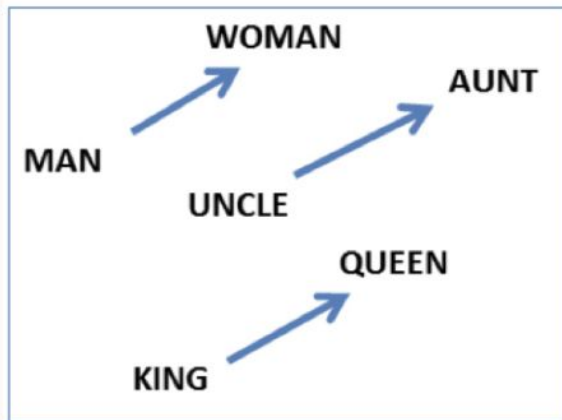
- **Pretrained embeddings**
  - Skip-gram
  - CBOW
  - fastText
  - GloVe
- **Training your own embeddings**
  - You can easily train skip-gram, CBOW, and fastText embeddings with `gensim`
  - Straightforward Python interface

# Observations on Embeddings

- Nearest words to some embeddings in the  $d$ -dimensional space.

<b>target:</b>	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

- Relation meanings
  - $\text{vector}(\text{king}) - \text{vector}(\text{man}) + \text{vector}(\text{woman}) \approx \text{vector}(\text{queen})$
  - $\text{vector}(\text{Paris}) - \text{vector}(\text{France}) + \text{vector}(\text{Italy}) \approx \text{vector}(\text{Rome})$

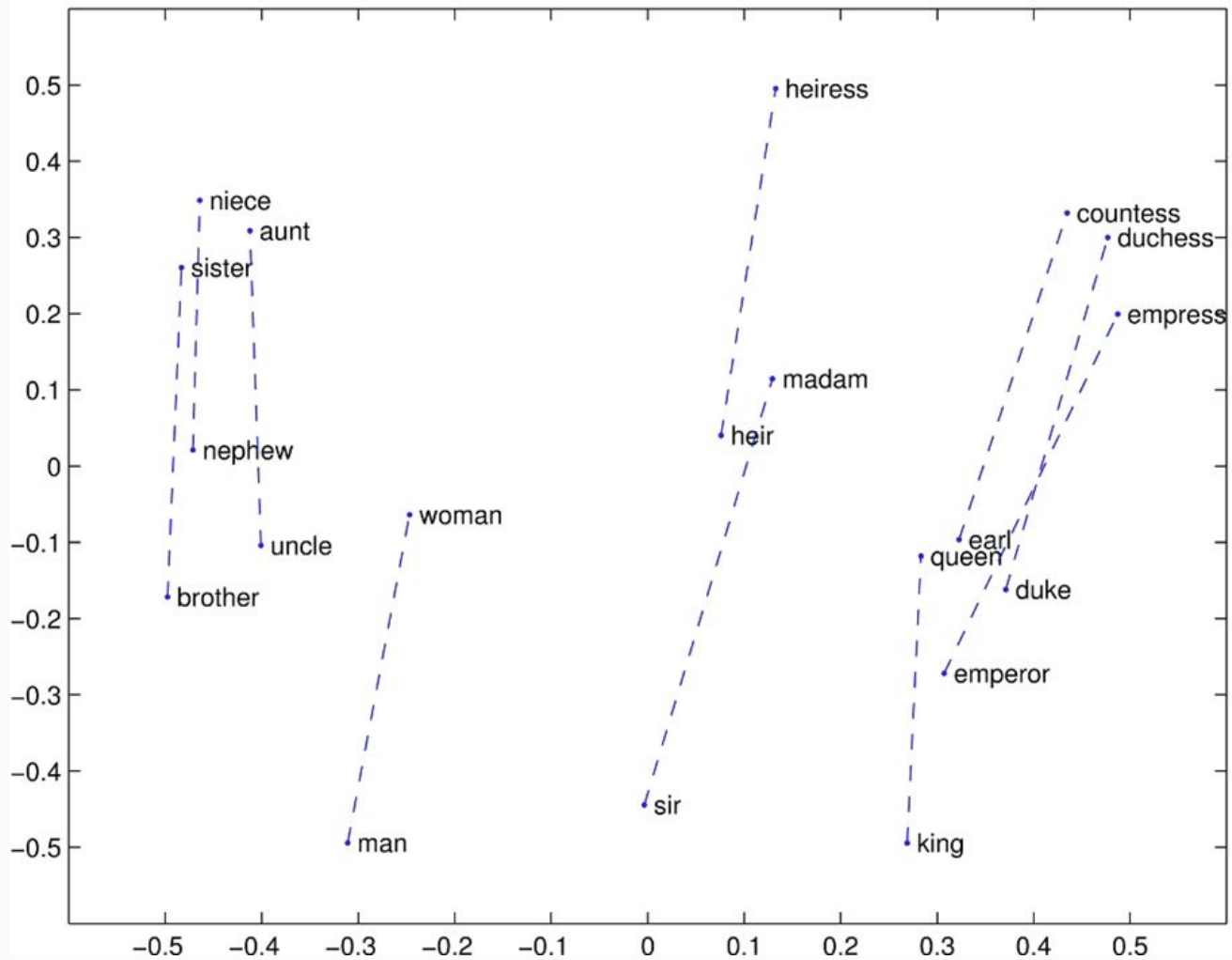




# Analogies

$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}}$  is close to  $\overrightarrow{\text{queen}}$   
 $\overrightarrow{\text{Paris}} - \overrightarrow{\text{France}} + \overrightarrow{\text{Italy}}$  is close to  $\overrightarrow{\text{Rome}}$

**Caveats:** only seems to work for frequent words, small distances and certain relations, like relating countries to capitals, or parts of speech [Linzen 2016, Gladkova et al. 2016, Ethayarajh et al. 2019a]



# Embeddings reflect cultural biases [Bolukbasi et al. 2016]

- Paris : France :: Tokyo : *Japan*
- Sexist occupational stereotypes
  - father : doctor :: mother : *nurse*
  - man : computer programmer :: woman : *homemaker*
- Would be problematic to use embeddings in hiring searches for programmers

# Discussion forum: Blodgett et al. 2020

- Recommendations from Blodgett et al. for better work on bias
  1. Ground work analyzing bias in relevant literature outside of NLP that explores relationships between language and social hierarchies. Treat representational harms as harmful in their own right
  2. Explicitly state why “bias” in systems is harmful, in what ways, and to whom. Be explicit about normative reasoning behind these judgements.
  3. Engage with the lived experiences of members of communities affected by NLP systems. Reimagine power relations between technologists and such communities.

# Discussion forum: Blodgett et al. 2020

- Challenges of Blodgett et al. 2020's recommendations
  - a. R1: differences in methodology and terminologies (Shayan)
  - b. R2:
    - Statements of values are controversial (Werner)
    - Stakeholders might be hesitant to acknowledge negative consequences (Aparna, Shiva)
    - Impossible to check for all types of biases (Arushi)
  - c. R3: challenging
    - What does it mean to be affected by an NLP system? (Werner)
    - Objectivity of research (Brian)
    - Need to rely on people/surveys/interviews, which is hard! (Sean)

# Discussion forum: Blodgett et al. 2020

- Allocational harms
  - a. Criminal justice prediction for sentencing (Purva, Brian)
  - b. Creditworthiness (Purva, Annanya, Deyasini)
  - c. Healthcare: diagnoses (Annanya, Brian, Deyasini)
    - Little data on minoritized groups (Hongtao)
  - d. Employment (Deyasini)
- “Using” bias in embeddings to investigate society, etc
  - a. Audit AI systems like chatbot interactions, customer service records, etc (Purva)
  - b. Assess textbooks, reading materials to identify representations of gender, race, etc (Purva)
  - c. Look at bias in non-US contexts like India (Kartik)
  - d. Looking at shifts in word connotation (Chonghao)
  - e. Biases among dialects of English, among geography within US (Jayden)

# Discussion forum: Blodgett et al. 2020

- Language ideologies
  - a. Chinese-style English seen as bad (Pengyu)
  - b. African American Vernacular English (AAVE) seen as worse (Yuelyu, Ayush, Deyasini, etc), which could affect e.g. dialogue systems (Yingda, Xiaoyan)
  - c. Southern American accent seen as “different” or “worse” (Kasvitha, Fatemeh) which could affect dialogue systems (Owen)
  - d. Castilian Spanish seen as better than Latin American Spanish (Hongtao)
  - e. Mandarin seen as “best” Chinese, though Xunfei company is developing translators for Chinese dialects (Yuning)

# Conclusion: vector semantics, static word embeddings

- NLP typically represents words as vectors in spaces where distance  $\approx$  semantic similarity
- Word2vec learns static embeddings (vectors) for words by predicting which words occur together in training data
- These embeddings are effective in downstream NLP tasks, but also reflect social biases of training data text



*Questions?*