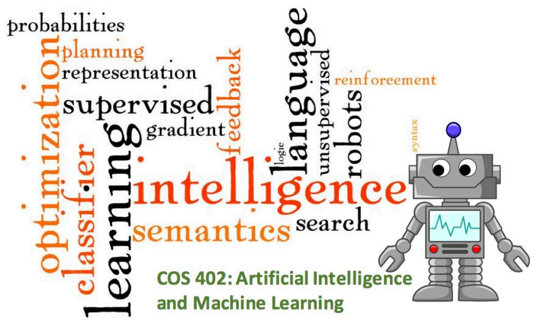


COS 402 – Machine Learning and Artificial Intelligence
Fall 2016

Lecture 10: Capturing semantics: Word Embeddings

Sanjeev Arora Elad Hazan

PRINCETON UNIVERSITY

(Borrows from slides of D. Jurafsky Stanford U.)

Last time

N-gram language models. Pr[w_2| w_1]

(Assign a probability to each sentence; trained using an unlabeled corpus)

Unsupervised learning

Today: Unsupervised learning for semantics (meaning)

Semantics

Study of meaning of linguistic expressions.

Mastering and fluently operating with it seems a precursor to AI (Turing test, etc.)

But every bit of partial progress can immediately be used to improve technology:

- Improving web search.
- Siri, etc.
- Machine translation
- Information retrieval,..

Let’s talk about meaning

What does the sentence “You like green cream” mean?
”Come up and see me some time.”
How did words arrive at their meanings?

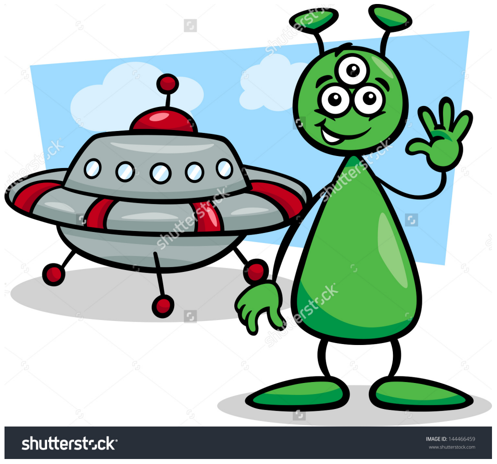
Your answers seem to involve a mix of

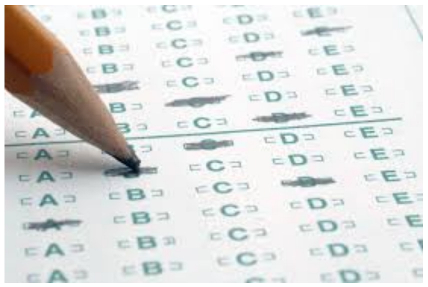
- Grammar/syntax
- How words map to physical experience
- Physical sensations “sweet”, “breathing”, “pain” etc. and mental states “happy,” “regretful” appear to be experienced roughly the same way by everybody, which help anchor some meanings)
-
-
- (at some level, becomes philosophy).

What are simple tests for understanding meaning?

Which of the following means same as pulchritude:
(a) Truth (b) Disgust (c) Anger (d) Beauty?

Analogy questions:
Man : Woman :: King : ??





"How can we teach a computer the notion of word similarity?"

Test: Think of a word that co-occurs with:
Cow, drink, babies, calcium…

Distributional hypothesis of meaning, [Harris'54], [Firth'57]

Meaning of a word is determined by words it co-occurs with.

"Tiger" and "Lion" are similar because they cooccur with similar words ("jungle", "hunt", "predator", "claws", etc.)

A computer could learn similarity by simply reading a text corpus; no need to wait for full AI!

How can we quantify “distribution of nearby words”?

```
A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.
```

Recall bigram frequency P(beer, drunk) $= \frac{count(beer, drunk)}{Corpus\ size}$

Redefine

count(beer, drunk) = # of times beer and drunk appear within 5 words of each other.

Matrix of all such bigram counts (# columns = V =10^6)

	study	alcohol	drink	bottle	lion
Beer	35	552	872	677	2
Milk	45	20	935	250	0
Jungle	12	35	110	28	931
Tesgüino	2	67	75	57	0

Vector representation of "beer."

Semantic Embedding [Deerwester et al 1990]

"Ignore bigrams with frequent words like "and", "for"

Which words look most similar to each other?

"Distributional hypothesis" → Word similarity boils down to vector similarity.

Cosine similarity

"Rescale vectors to unit length, compute dot product."

$$|\vec{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

- High when two vectors have large values in same dimensions.
- Low (in fact 0) for **orthogonal vectors** with zeros in complementary distribution

$$\text{Cosine Similarity} = \frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|}$$

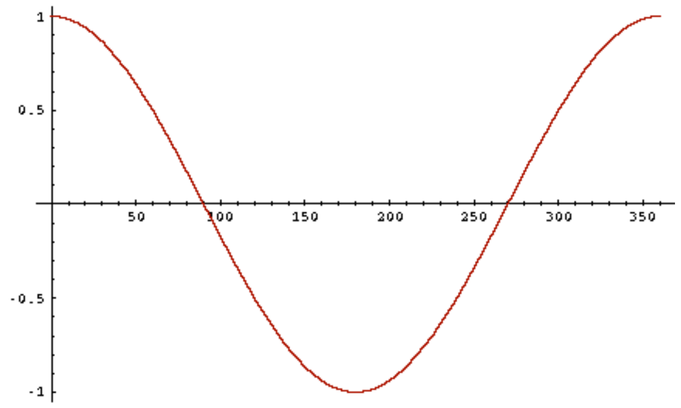
$$\vec{a}\cdot\vec{b} = |\vec{a}||\vec{b}|\cos\theta$$

$$\frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|} = \cos\theta$$

Other measures of similarity have been defined….

Reminder about properties of cosine

- -1: vectors point in opposite directions
- +1: vectors point in same directions
- 0: vectors are orthogonal



Nearest neighbors (cosine similarity)

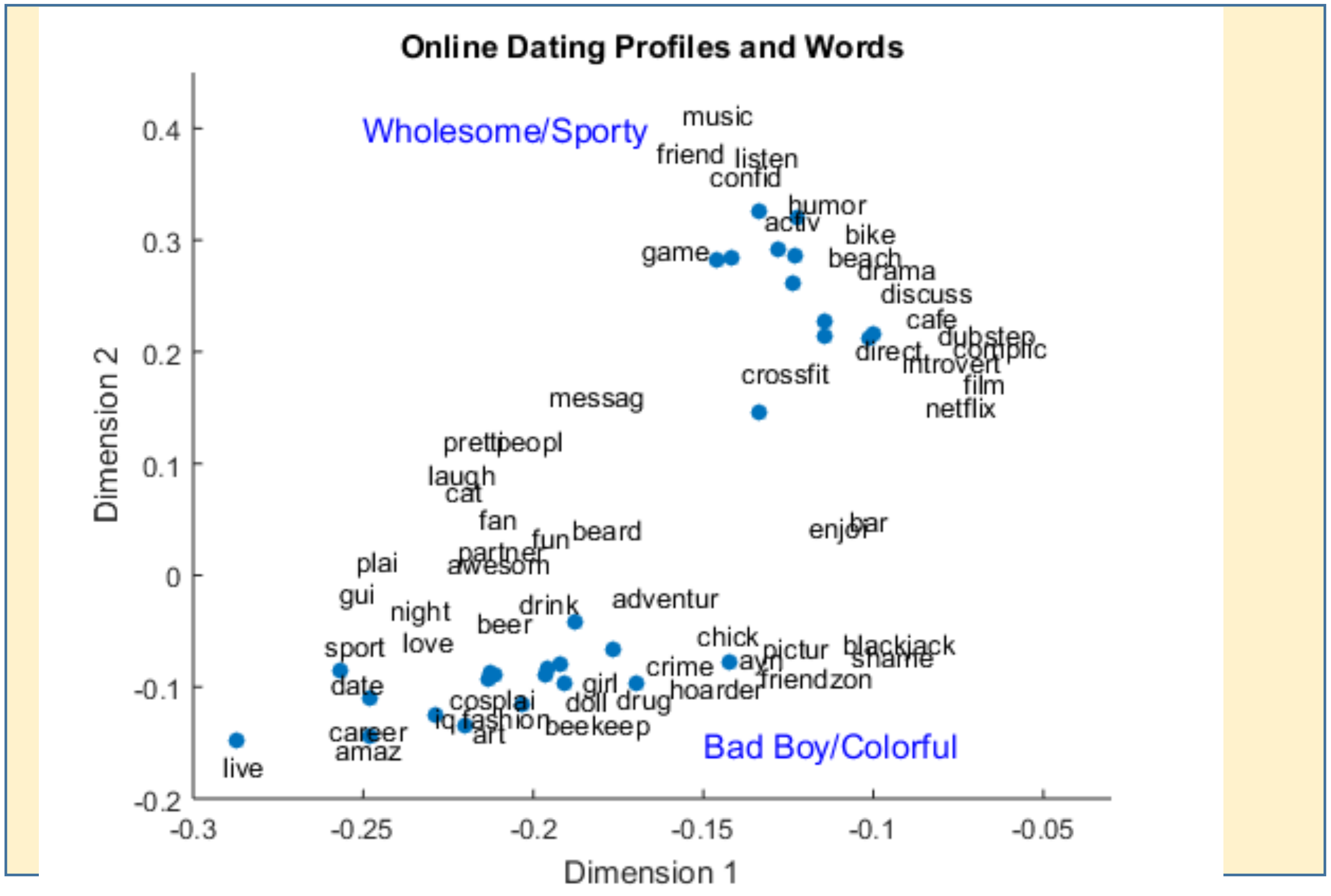
Apple:	Ball:
'macintosh'	'balls'
'microsoft'	'game'
'iphone'	'play'
'ibm'	'thrown'
'software'	'back'
'ios'	'hit'
'ipad'	'throwing'
'ipod'	'player'
'apples'	'throw'
'fruit'	'catch'

Important use of embeddings: allow language processing systems to make a guess when labeled data is insufficient. (borrow data/trained features of nearby words)

(Semisupervised learning: Learn embeddings from large unlabeled corpus; use as help for supervised learning)

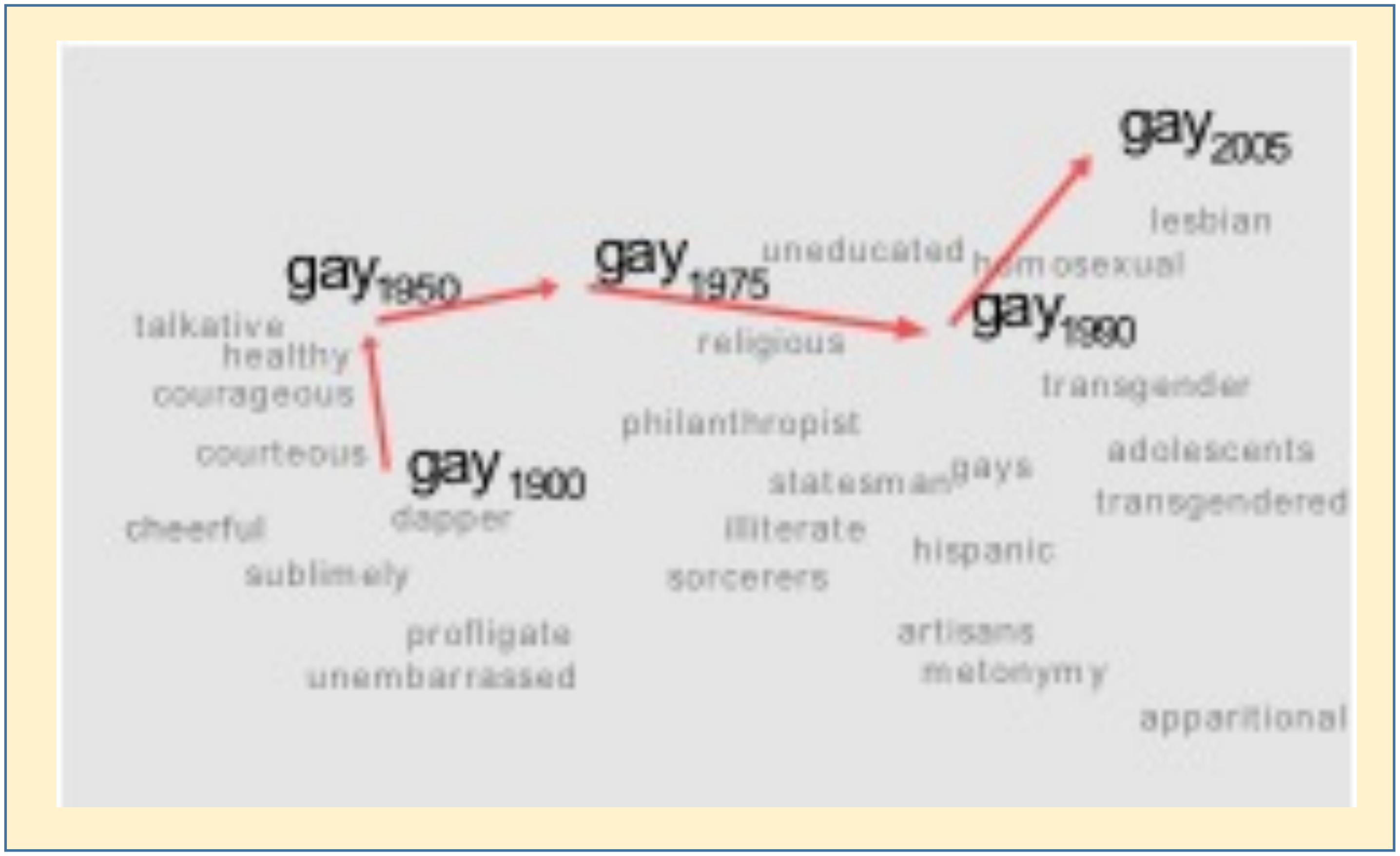
Anthropology field trip: visualization of word vectors obtained from dating profiles

[Loren Shure, matworks.com]



“Evolution of new word meanings”

Kulkarni, Al-Rfou, Perozzi, Skiena 2015



Problems with above naive embedding method

- Raw word frequency is not a great measure of association between words
  - It’s very skewed
    - “the” and “of” are very frequent, but maybe not the most discriminative
- We’d rather have a measure that asks whether a context word is **particularly informative** about the target word.
  - Positive Pointwise Mutual Information (PPMI)

Pointwise Mutual Information (PMI)

Pointwise mutual information:

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

Positive PMI: max{0, PMI} (if PMI is -ve, make it 0).

PMI between two words: (Church & Hanks 1989)

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

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

High for (lion, hunt) but ≈ 0 for (lion, cake). (Why?)

Improved embedding

	study	alcohol	drink	bottle	lion
Beer	35	552	872	677	2
Milk	45	20	935	250	0
Jungle	12	35	110	28	931
Tesgüino	2	67	75	57	0

Replace each entry by corresponding PPMI(word1, word2)

Better: Before computing PPMI(,), do Add-1 smoothing from last time.
(reduces crazy effects due to rare words)

Implementation issues

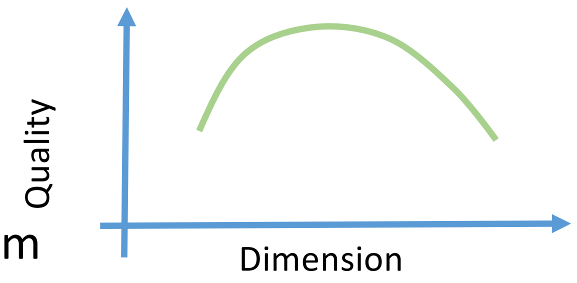
- V = vocabulary size (say 100,000)
- Then word embeddings defined above are V-dimensional. Very clunky to compute with! (Will improve soon.)
- We defined bigrams using "windows of size 5"
  - The shorter the windows , the more **syntactic** the representation
    $\pm$ 1-3 very syntacticy
  - The longer the windows, the more **semantic** the representation
    $\pm$ 4-10 more semanticy

Dense word embeddings.

- PPMI vectors are
  - **long** (dimension |V|= 20,000 to 50,000)
  - **sparse** (most elements are zero)
- Alternative: learn vectors which are
  - **short** (dimension 200-1000)
  - **dense** (most elements are non-zero)

Each dimension in dense version represents many old dimensions
(but it isn't some trivial consolidation)

Why dense embeddings



- Short vectors may be easier to use as features in downstream machine learning (fewer dimensions – fewer parameters to tune)
- Dense vectors may generalize better than storing explicit counts (perform better at many tasks)

- Sparse long vectors ignore synonymy:
    - *car* and *automobile* are synonyms; but are represented as distinct coordinates; this fails to capture similarity between a word with *car* as a neighbor and a word with *automobile* as a neighbor

How to compute dense embeddings.

For each word, seeking a vector v_w in R^{300} (nothing special about 300; could be 400)

For every pair of words w, w' desire

$$v_w \cdot v_{w'} \approx \text{PPMI}(w, w')$$

Word vector retains all info about the word's co-occurences

Training with l_2 loss: $\text{Minimize} \sum_{w,w'} (v_w \cdot v_{w'} - \text{PPMI}(w, w'))^2$

300V numbers replace V x V matrix of PPMI values. (Compression!)

Aside: this optimization is called "Rank-300 SVD" in linear algebra. (aka Principle Component Analysis)

Even better embeddings?

$$\text{Minimize} \sum_{w,w'} (v_w \cdot v_{w'} - \text{PPMI}(w,w'))^2$$

(replaces V x V matrix of PMI values by V vectors of dimension 300. Big win!)

Problem with this compression: gives equal importance to every matrix entry.

Recall: PPMI( ) estimated using cooccurence counts.
Are some estimates more reliable than others?

Better: $$\text{Minimize} \sum_{w,w'} \text{Count}(w,w')(v_w \cdot v_{w'} - \text{PPMI}(w,w'))^2$$

[Aside 1: Theoretical justification for this objective in [A, Li, Liang, Ma, Risteski TACL 2016].
Aside 2: Many other related methods for compressed embeddings: word2vec, Glove, neural nets,..)

How to solve training objective

$$\text{Minimize} \sum_{w,w'} \text{Count}(w,w')(v_w \cdot v_{w'} - \text{PPMI}(w,w'))^2$$

Suggestions?

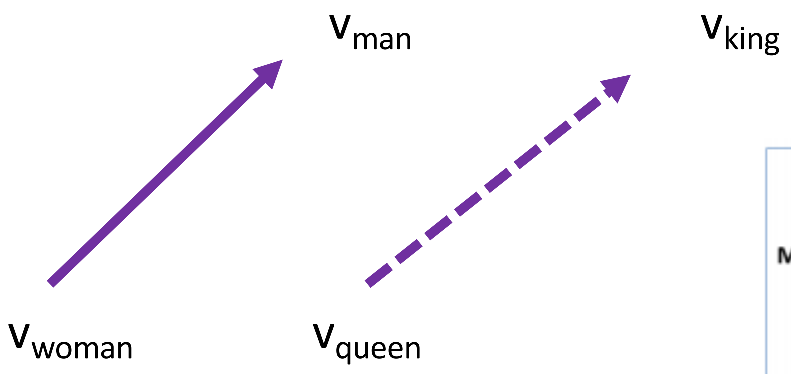
Gradient descent works. Number of variables is 300 V.
Objective is a quadratic function of these variables.

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v} \cdot \vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

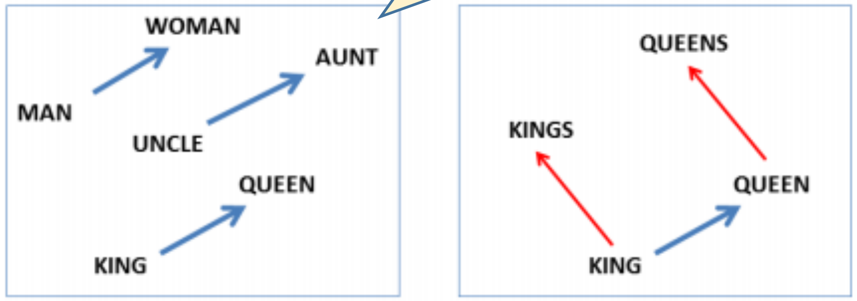
Cool property of PMI-based embeddings: Analogy solving

("word2vec" [Mikolov et al 2013])

Man : Woman :: King : ??



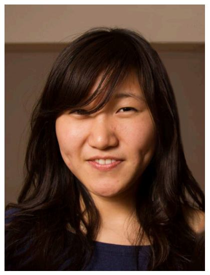
Warning: Such pictures (plentiful on internet) are v. misleading. Ask me about correct interpretation



Find w to minimize $\|v_{man} - v_{woman} + v_{king} - v_w\|_2$

Some other cool applications
(don’t expect you to fully understand; details won’t be on exam)

Extending knowledge bases



Lisa Lee'15

Given the list Kyoto, Shanghai, Chongqing, Chengdu, Busan, Incheon,.. generate more examples.

(use a large corpus like WSJ or wikipedia)

Solution: Estimate the "best line capturing their embeddings" (rank 1 SVD); find other words whose embeddings are close to this line.

(j) $c =$ asian_city

word	projection
taipei	0.837
taichung	0.819
kaohsiung	0.818
osaka	0.806
tianjin	0.765

(i) $c =$ animal

word	projection
horses	0.806
moose	0.784
elk	0.783
raccoon	0.763
goats	0.762

(d) $c =$ basketball_player

word	projection
dwyane_wade	0.788
aren	0.721
kobe_bryant	0.715
chris_bosh	0.712
tim_duncan	0.708

Resolving meanings of polysemous words

[A. Li, Liang, Ma, Risteski’16]

Polysemous: Has multiple meanings.

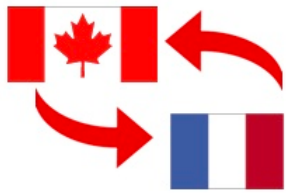
Example: How many meanings does “tie” have? “Spring?”

tie				
trousers	season	scoreline	wires	operatic
blouse	teams	goalless	cables	soprano
waistcoat	winning	equaliser	wiring	mezzo
skirt	league	clinching	electrical	contralto
sleeved	finished	scoreless	wire	baritone
pants	championship	replay	cable	coloratura

spring				
beginning	dampers	flower	creek	humid
until	brakes	flowers	brook	winters
months	suspension	flowering	river	summers
earlier	absorbers	fragrant	fork	ppen
year	wheels	lilies	piney	warm
last	damper	flowered	elk	temperatures

Meanings extracted using a linear-algebraic procedure called sparse coding.

Allow programs like machine translation to deal with unknown words



Translator trained using transcripts of Canadian parliament
(large corpus, but not huge)

Encounters a word that doesn’t occur in training corpus (eg, “dude”)

Can use word embeddings (trained on a large English-only corpus like wikipedia) to learn a lot about “dude,” and neighboring words → can guess approximate translation of ”dude”.

Sentence embeddings

a man with a jersey is dunking the ball at a basketball game

the ball is being dunked by a man with a jersey at a basketball game

Similar

people wearing costumes are gathering in a forest and are looking in the same direction

a little girl in costume looks like a woman

Dissimilar

Simplest sentence embedding = Average of word embeddings.
(Lots of research to find better embeddings, including using neural nets.)

Next lecture: Collaborative filtering

(eg movie or music recommender systems)

Homework is out today; due next Tues

Midterm: A week from Thurs.