RICE SOCIAL SCIENCES
Department of Linguistics

word embeddings

LING 430 Computational Linguistics Fall 2026
lecture 11b
Nov 5 2026

Siyu Liang
siyu.liang@rice.edu

last time

- vector semantics
  - distributional semantics
  - cosine similarity
  - tf-idf

agenda

- word vectors
- word2vec
- properties of embeddings

today's notebook

class11b.ipynb

- course website → week 11 → activity
  - **File → Save a copy in Drive**

word vectors

word vectors

last time

- “you shall know a word by the company it keeps” (Firth 1957)
- we created **word vectors** by counting words within the **context window**
- we compared word vectors using **cosine similarity**
- we weighted the counts with **tf-idf** so that common words matter less

	pie	***data***	***computer***
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

word vectors

sparse vs dense vectors

- how long is the vector for *cherry* actually?
  - we only looked at three last time
  - in order to be compatible for all words, we need all of $|V|$ columns

	aardvark	***abacus***	**…**	***computer***	***data***	**…**	***pie***	***sugar***	**…**	***zebra***	***zydeco***
cherry	0	0	…	2	8	…	442	25	…	0	0
strawberry	0	0	…	0	0	…	60	19	…	0	0
digital	0	0	…	1670	1683	…	5	4	…	0	0
information	0	0	…	3325	3982	…	5	13	…	0	0

word vectors

sparse vs dense vectors

- count vectors (even if weighted by tf-idf)
  - **long** (length $|V|$ = 20,000 to 50,000)
  - **sparse** (most elements are zero)
- alternative: learn vectors which are
  - **short** (length 50-1000)
  - **dense** (most elements are non-zero)
- a short dense vector for a word is an **embedding**

word vectors

why dense vectors?

- short vectors may be easier to use as features in machine learning (fewer weights to tune)
- dense vectors may generalize better than explicit counts
- dense vectors 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**

word2vec

word2vec

the motivation

- is *tree* a neighbor of *apricot*? yes. is *queen*? no
- 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"?
- we don't care about this task
  - but we'll take the **learned classifier weights** as the **word embeddings**
- big idea: **self-supervision**
  - a word c that occurs near *apricot* acts as the gold "correct answer"
  - no need for human labels (Bengio et al. 2003; Collobert et al. 2011)

word2vec

the approach

- how to predict if candidate word c is a “neighbor”?
  - 1. treat the target word w and a neighboring context word c as positive examples
  - 2. randomly sample other words in the lexicon to get negative examples
  - 3. use logistic regression to train a classifier to distinguish those two cases
  - 4. use the learned weights as the embeddings

word2vec

skip-gram

- word2vec (Mikolov et al. 2013) has two main features:
  - **skip-gram**: from the target word, predict the words around it
    - a pair of words that need not be adjacent is a “skip-gram” (Guthrie et al. 2006)
    - e.g. (apricot, tablespoon) skips over “of” in “a tablespoon of apricot jam”
  - **CBOW** continuous bag of words: from the words around it, predict the target

skip-gram

positive examples

…lemon, a [tablespoon of apricot jam, a] pinch…
c1 c2 [target] c3 c4

- target w = *apricot*, window ±2

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

skip-gram

negative examples

- for each positive pair, k **noise words**: random words from the lexicon, never the target itself
- here k = 2, so eight negative pairs
- noise words are sampled by unigram frequency, flattened a little so rare words get a chance

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

skip-gram

the classifier

- then we need to train a classifier for the task
  - given a pair (w, c): how likely is it that c is a real neighbor of w?
- remember: two vectors are similar if they have a high dot product
  - and cosine is just a normalized dot product
- so: similarity(w, c) $\propto w \cdot c$
- we'll need to normalize to get a probability
  - cosine isn't a probability either

skip-gram

the classifier

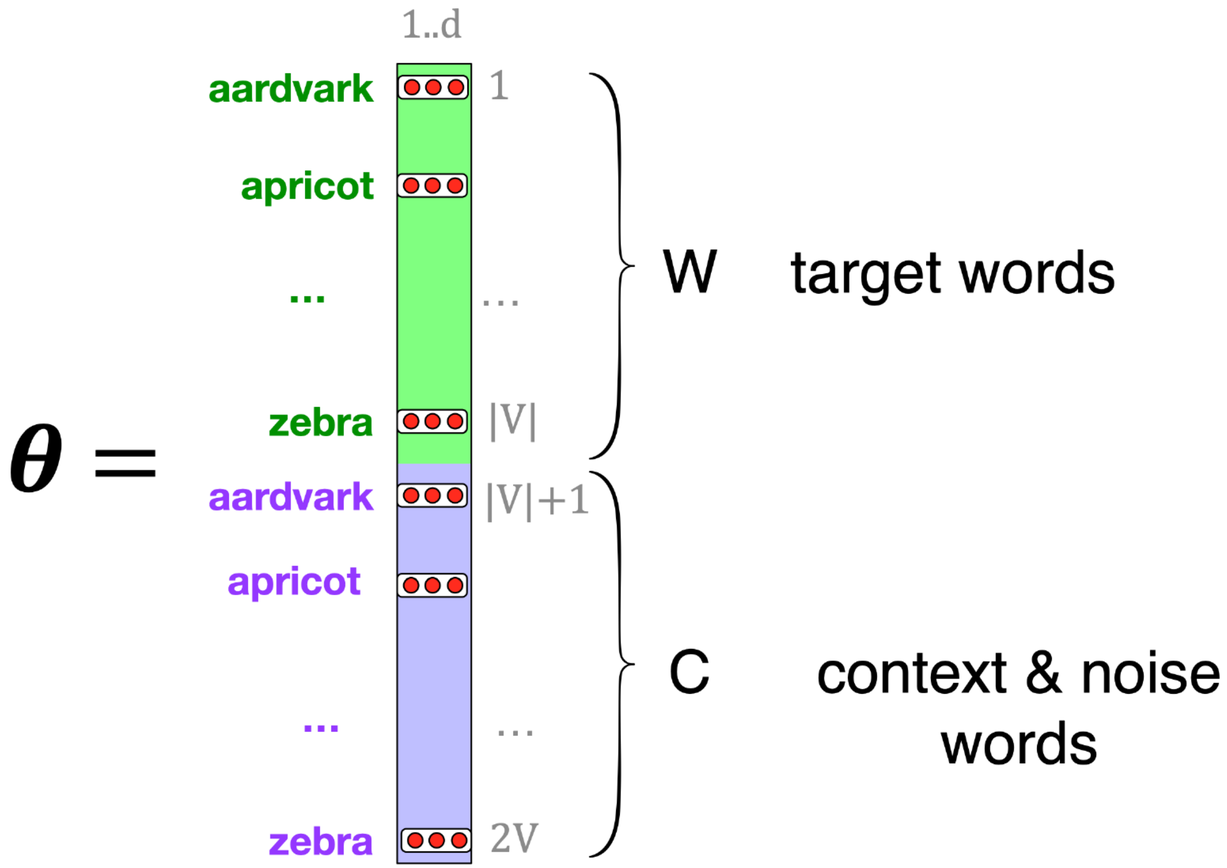
- sim(w, c) ≈ $w \cdot c$
- we then need to turn dot products into probabilities
- we'll use the sigmoid function from logistic regression

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

skip-gram

embedding structure

- every word gets two vectors:
  - a row in W for the word as a target
  - a row in C for the word as a context or noise word



skip-gram

example

- suppose we have a corpus of 200 two-word phrases made from 8 words
  - *apricot jam, zebra savanna, tangerine tart, aardvark zoo, apricot tart, zebra zoo, …*
  - *apricot* and *tangerine* never occur together; neither do *apricot* and *zoo*
- we will train an embedding of d = 3, window ±1, and k = 2

skip-gram

- is *jam* a neighbor of *apricot*?

  w_{apricot} = [.07, .52, .08]

  c_{jam} = [.07, −.06, −.52]

  $c_{\text{jam}} \cdot w_{\text{apricot}}$

  $= .07\times.07 + (-.06)\times.52 + (-.52)\times.08$

  $= .005 - .031 - .042 = -.07$

  $\sigma(-.07) = \frac{1}{1+e^{.07}} = .48$

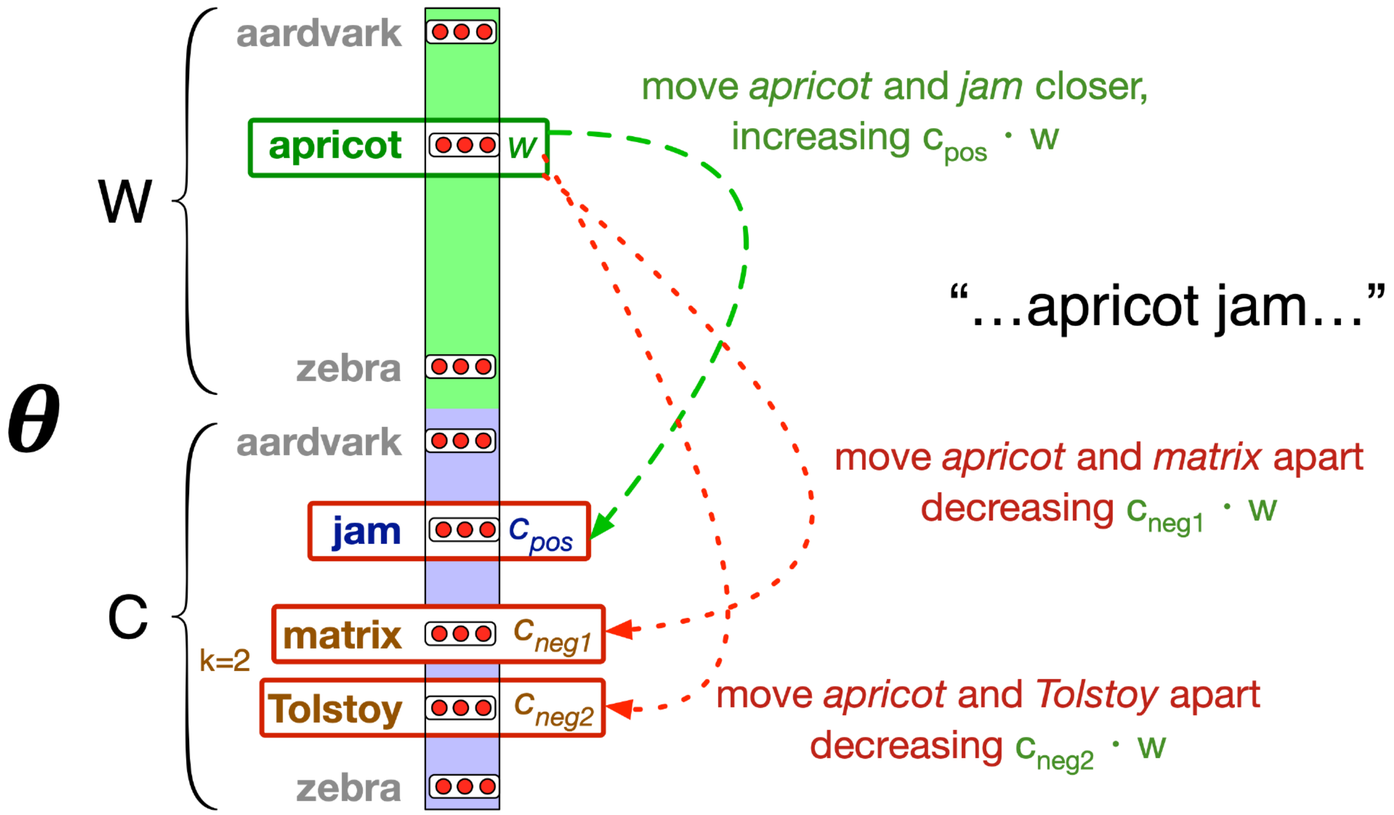
- the classifier knows nothing yet

	w_1	w_2	w_3	c_1	c_2	c_3
apricot	.07	.52	.08	−1.07	−.08	−.60
tangerine	.31	.82	.14	.56	.63	−.98
jam	.10	−.14	−.80	.07	−.06	−.52
aardvark	−.01	.56	−.03	−.13	−.07	.05
zebra	−.41	.18	−.28	−.62	.31	.37
zoo	−.09	.02	−.07	−.57	−.33	−.04

skip-gram

intuition of training

- the corpus says *apricot jam*, so:
  - "move" the target row of *apricot* toward the context row of *jam*
  - and away from the context rows of the two noise words



skip-gram

the loss

- we use loss to nudge the weights
  - lower loss is better
- one positive pair and its k noise words
  - make the real neighbor likely, and the noise words unlikely

$$L = -[\log\sigma(c_{\text{pos}} \cdot w) + \sum_{i=1}^{k} \log\sigma(-c_{\text{neg}i} \cdot w)]$$

e.g. is jam a real neighbor of apricot?

e.g. zoo is not a neighbor of apricot

skip-gram

the learning

- how do we nudge based on the loss?
  - stochastic gradient descent from week 8!
  - each step moves one target row and k + 1 context rows, by how wrong the classifier was
- “move” = add a small multiple of the other row, with learning rate η
  - toward the real neighbor: η × (1 − σ), the gap to “yes”
  - away from each noise word: η × σ, the gap to “no”
  - a row that is already right barely moves

skip-gram

the learning

- example of a tiny learning step

before		$\sigma(c \cdot w)$
w_{apricot}	[.07, .52, .08]	
c_{jam}	[.07, −.06, −.52]	.48
c_{zoo}	[−.57, −.33, −.04]	.45
c_{zebra}	[−.62, .31, .37]	.54



after		$\sigma(c \cdot w)$
w_{apricot}	[.10, .51, .06]	
c_{jam}	[.07, −.05, −.52]	.49
c_{zoo}	[−.58, −.34, −.04]	.44
c_{zebra}	[−.62, .29, .37]	.53

skip-gram

after learning

- after 100 steps: is *jam* a neighbor of *apricot*?

  $w_{\text{apricot}} = [2.48, 1.29, -.72]$

  $c_{\text{jam}} = [-1.06, 1.59, -2.85]$

  $c_{\text{jam}} \cdot w_{\text{apricot}}$
  $= (-1.06)\times 2.48 + 1.59\times 1.29$
  $+ (-2.85)\times(-.72)$
  $= -2.63 + 2.05 + 2.05 = 1.47$

  $$\sigma(1.47) = \frac{1}{1+e^{-1.47}} = .81$$

  - likely yes!
- and c_{aardvark} .00, c_{apricot} .04

	w_1	w_2	w_3	c_1	c_2	c_3
apricot	2.48	1.29	−.72	−1.57	−.90	−2.67
tangerine	2.58	1.06	−.48	−1.59	−1.04	−2.86
jam	1.90	−2.17	−.43	−1.06	1.59	−2.85
aardvark	.33	1.71	2.36	−3.26	−.65	−.48
zebra	.38	1.71	2.33	−3.13	−.50	−.81
zoo	−.17	−1.68	2.35	−2.76	1.98	−.71

skip-gram

why two matrices?

- why not one vector per word, with $c = w$?
  - then $w \cdot w = |w|^2$, the largest dot product in the row
  - the model would call every word its own best neighbor
- with two, w_{apricot} and c_{apricot} are free to point in different directions
- and the noise words are not stored anywhere negative
  - *zoo* is a noise word for *apricot* and a real neighbor of *zebra*, with the same row in both cases

skip-gram

what each matrix knows

- $W \cdot C$ answers “appears with”
  - $w_{\text{apricot}} \cdot c_{\text{jam}}$ is high
- $W \cdot W$ answers “appears in the same places as”
  - cos(w_{apricot}, $w_{\text{tangerine}}$) = .99, though the two never occur together
  - cos(w_{apricot}, w_{jam}) = .26: neighbors, not look-alikes

skip-gram

- when training stops, we discard the classifier
- the embedding of word i is its row w_i in W
  - or often $w_i + c_i$
- the window size matters, as it did for counting
  - we tune it on a dev set

word2vec

- some famous w2v models
  - **GloVe** (Pennington et al. 2014), Global Vectors
    - built from ratios of co-occurrence probabilities in the word-word matrix
  - **fasttext** (Bojanowski et al. 2017)
    - a word is itself plus its character n-grams: <*wh, whe, her, ere, re*> for *where*
    - so an unknown word gets a vector from its pieces

properties of embeddings

properties of embeddings

visualizing embeddings

- “I see well in many dimensions as long as the dimensions are around two” (Martin Shubik)
- the simplest way to look at a 100-dimensional vector: list the words with the highest cosine
  - the 7 closest words to *frog* in one GloVe space: *frogs, toad, litoria, leptodactylidae, rana, lizard, eleutherodactylus*
- but us humans can’t see 100dimensions at once
  - project the dimensions down to 2 (or 3), with PCA or t-SNE

properties of embeddings

pretrained embeddings in Colab

- GloVe trained on Wikipedia and newswire: 400,000 words, 50 dimensions, all lowercase

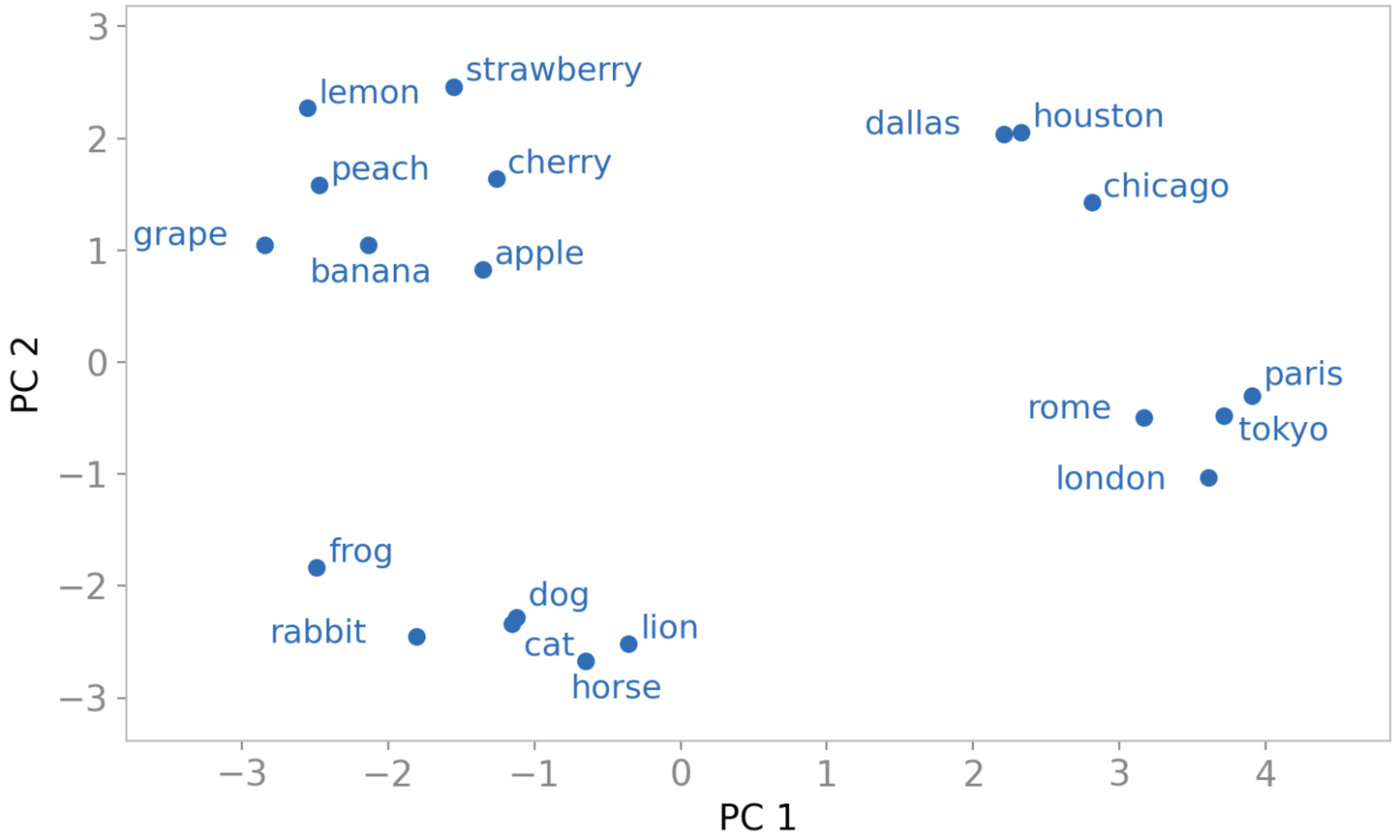
```
import gensim.downloader as api
glove = api.load("glove-wiki-gigaword-50")
glove.most_similar("linguistics", topn=5)
```

→ [('anthropology', 0.871), ('philology', 0.868),
→ ('sociology', 0.801), ('comparative', 0.794),
→ ('psychology', 0.792)]

properties of embeddings

20 words in two dimensions

- the same GloVe vectors, 50 dimensions projected onto their two principal components
- how would you describe the groupings?



properties of embeddings

window size

- short windows (±2): **syntactic** neighbors, same part of speech, substitutable
- long windows (±5): **topical** neighbors, related but not similar
- the neighbors of *Hogwarts* in skip-gram (Levy & Goldberg 2014)
  - ±2: *Sunnydale*, *Evernight*, other fictional schools
  - ±5: *Dumbledore*, *Malfoy*, *half-blood*, the Harry Potter world

properties of embeddings

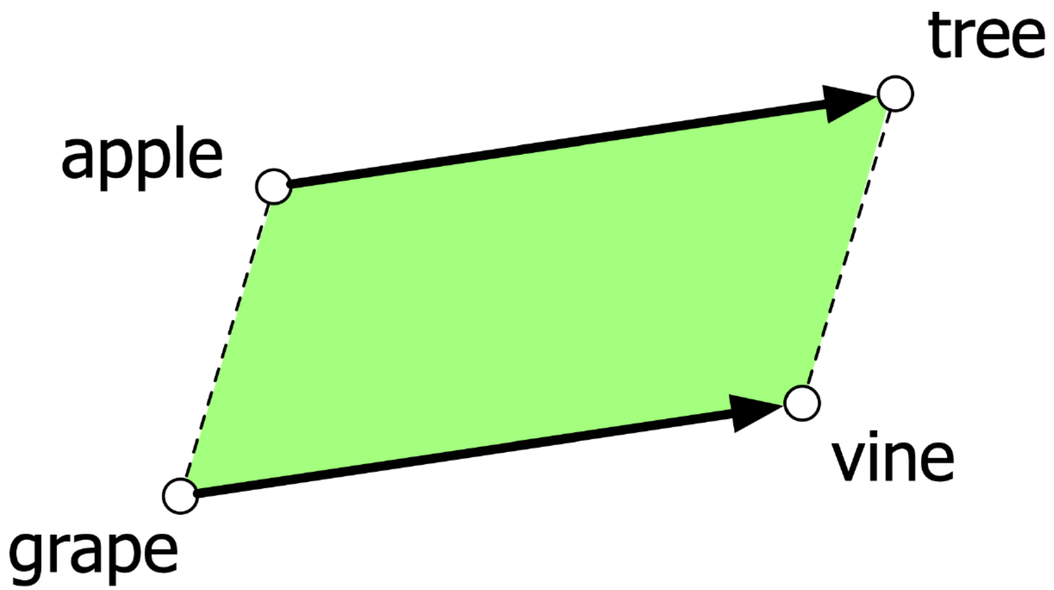
two kinds of association

- **first-order co-occurrence** (syntagmatic): the words are typically near each other
  - *wrote* and *book*
- **second-order co-occurrence** (paradigmatic): the words have similar neighbors
  - *wrote* and *said*
- nearest neighbors by cosine in W are second-order
  - in our GloVe, *cat* → *dog* .92, *tuesday* → *monday* .998

properties of embeddings

analogies

- *apple* is to *tree* as *grape* is to what?
- Rumelhart & Abrahamson (1973): take the vector from *apple* to *tree*, add it to *grape*, return the nearest word



properties of embeddings

analogies in GloVe

- *king − man + woman ≈ queen*
- *Paris − France + Italy ≈ Rome*

```
glove.most_similar(positive=["king", "woman"],
                   negative=["man"], topn=3)
```

→ [('queen', 0.852), ('throne', 0.766), ('prince', 0.759)]

```
glove.most_similar(positive=["paris", "italy"],
                   negative=["france"], topn=3)
```

→ [('rome', 0.847), ('milan', 0.777), ('turin', 0.767)]

properties of embeddings

caveats

- the nearest vector is usually an input word, which gensim excludes for you
- it works for frequent words and some relations (Linzen 2016)
  - *cherry* : *red* :: *potato* : ? answers *soup*
- what if we want to know which one is different, *lemon, orange, apple, cucumber*?
  - `doesnt_match` answers *apple*, not *cucumber*
  - this space probably knows what goes in a salad, not what a fruit is

properties of embeddings

WFS 7.3 (Xaa) n, sing, masc ⟷	(Xii) n, sing, fem
लड़का *laRkaa*, boy	लड़की *laRkii*, girl
बेटा *beTaa*, son	बेटी *beTii*, daughter

Other words that would follow the same pattern include बकरा *bakraa*, 'male goat', दादा *daadaa*, 'paternal grandfather', चेला *celaa*, 'pupil' etc., which have female counterparts in बकरी *bakrii*, दादी *daadii* and चेली *celii*. If a human noun ends in a consonant, we obtain the female counterpart by the addition of *-ii* as in कुमार *kumaar*, 'young boy'; कुमारी *kumaarii*, 'young girl' and दास *daas*, 'servant'; दासी *daasii*, 'female servant'. This, however, is a very small set of word pairs.

properties of embeddings

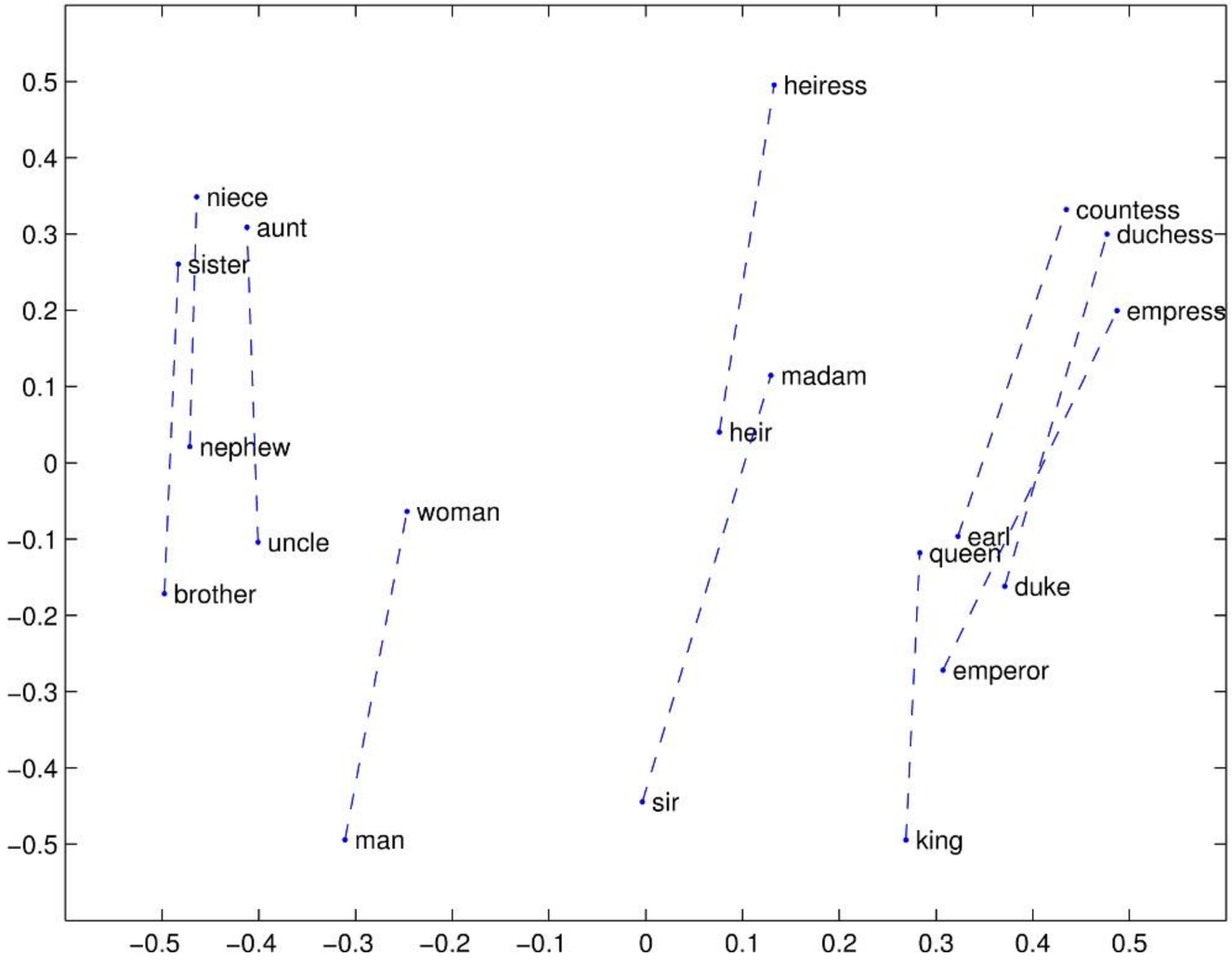
analogies in Hindi

- fastText has a Hindi version, built from the Hindi Wikipedia (Bojanowski et al. 2017)
- लड़का *laRkaa* “boy” : लड़की *laRkii* “girl” :: X : ?

X	nearest	cosine
बेटा *beTaa*, son	बेटी *beTii*, daughter	.62
बकरा *bakraa*, male goat	बकरी *bakrii*, goat	.63
दादा *daadaa*, grandfather	दादी *daadii*, grandmother	.63
कुमार *kumaar*, young boy	कुमारी *kumaarii*, young girl	.59
भाई *bhaaii*, brother	बहन *bahan*, sister	.64
धोबी *dhobii*, washerman	धोती *dhotii*, a garment: धोबिन is too rare for the vocabulary	.54

properties of embeddings

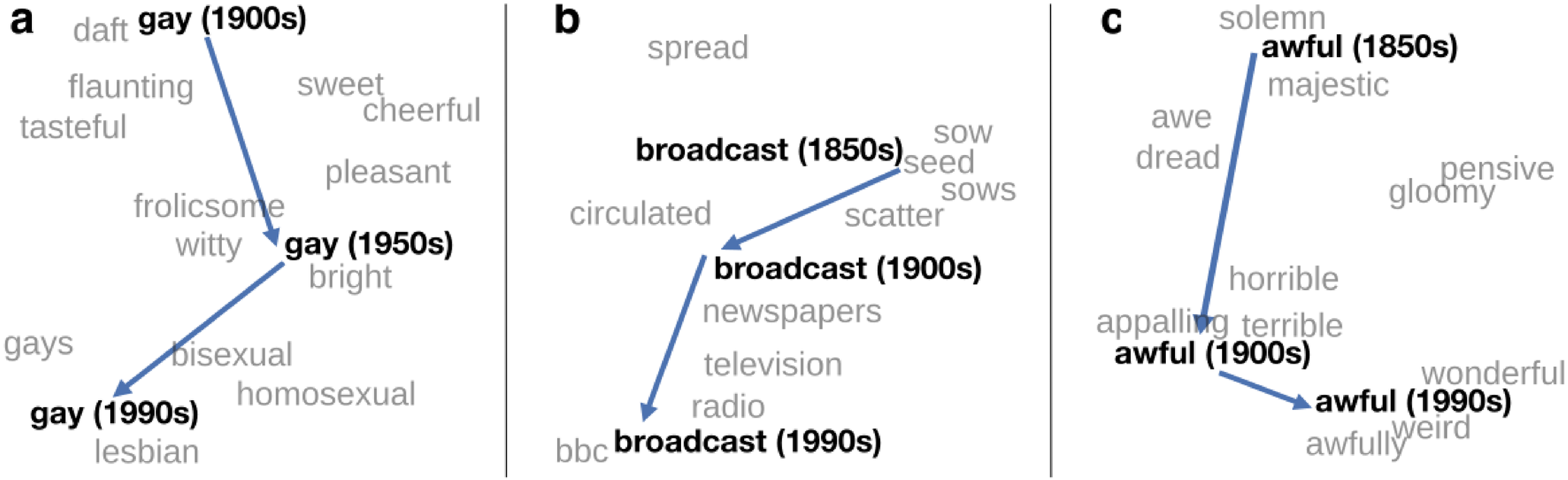
structure in GloVe space



properties of embeddings

change over time

- train one embedding space per decade (Hamilton et al. 2016)
  - Google n-grams and the Corpus of Historical American English



activity

- try fastText vectors on a language of your choice in the notebook today
- 1. pick a language you know: English, Spanish, Chinese, Vietnamese, Japanese, Korean, Hindi, Arabic, French, …
- 2. try combinations of *a* : *b* :: *c* : ?
  - e.g. gender pair, plural, past tense, capital city, etc.
- 3. see if you could find one that worked and one that failed, and why

properties of embeddings

bias

- embeddings learn the implicit biases and stereotypes in their text
- Bolukbasi et al. (2016), word2vec trained on news
  - *computer programmer* − *man* + *woman* = *homemaker*
  - *father* : *doctor* :: *mother* : *nurse*
- our GloVe vector: *doctor* − *father* + *mother* → *nurse*
- an **allocational harm** (Crawford 2017, Blodgett et al. 2020)
  - a system allocates resources, jobs or credit, unfairly
  - e.g. a résumé search for programmers that downweights women's names

properties of embeddings

implicit associations

- the Implicit Association Test (Greenwald et al. 1998)
  - people pair *flowers* with pleasant words faster than with unpleasant ones
- Caliskan et al. (2017) replaced reaction times with GloVe cosines and replicated every finding
  - African American names like *Leroy*, *Shaniqua*: higher cosine with unpleasant words
  - European American names like *Brad*, *Courtney*: with pleasant words
- GloVe trained on tweets gives *gay* the neighbors *troll*, *fake*, *hater,* reflecting how the word is thrown around on Twitter
- a **representational harm**: a system demeaning or ignoring a social group

properties of embeddings

can we remove it?

- **debiasing**: transform the space to remove gender stereotypes but keep definitional gender (Bolukbasi et al. 2016), or change the training procedure (Zhao et al. 2018)
- it reduces the bias but does not eliminate it (Gonen & Goldberg 2019)
- Garg et al. (2018) track the changes through the 20th century
  - the cosine of women's names to an occupation tracks the real share of women in it
  - competence adjectives (*smart*, *wise*, *thoughtful*) sit closer to male words, less so every decade since 1960
  - dehumanizing adjectives (*barbaric*, *monstrous*, *bizarre*) sat closest to Asian last names in 1930s text; that bias fell over the century

HW 6

- released today, due **next Thursday**
- last hw of the course!
- written/typed homework
- given a word vector constructed from a song, guess the artist

next week

- Tuesday, Nov 10: neural networks
- Thursday, Nov 12: large language models