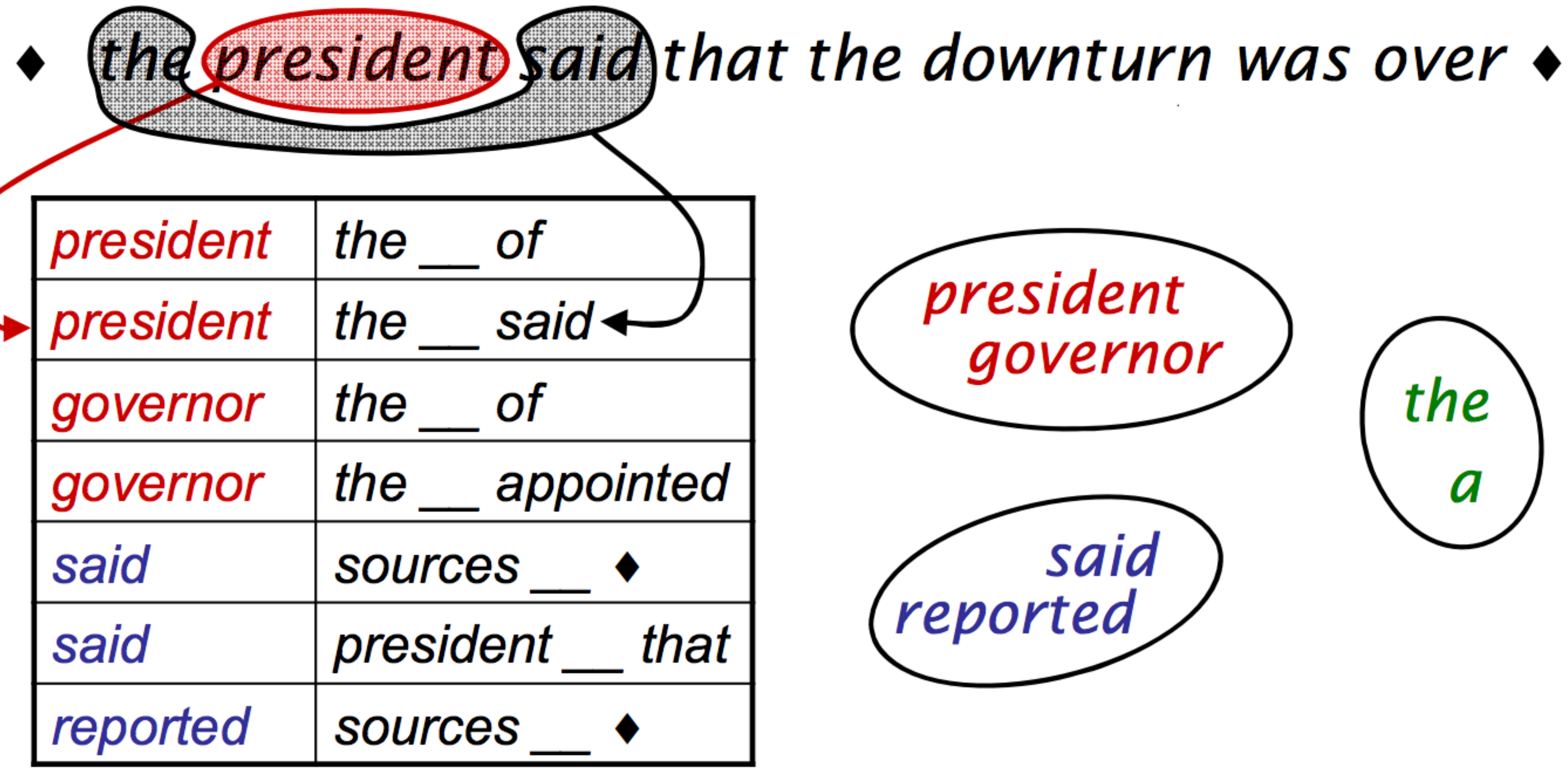
Word Representations

Word Representations

- Neural networks work very well at continuous data, but words are discrete
- Continuous model <-> expects continuous semantics from input
- “You shall know a word by the company it keeps” Firth (1957)



[Finch and Chater 92, Shuetze 93, many others]

slide credit: Dan Klein

Word Embeddings

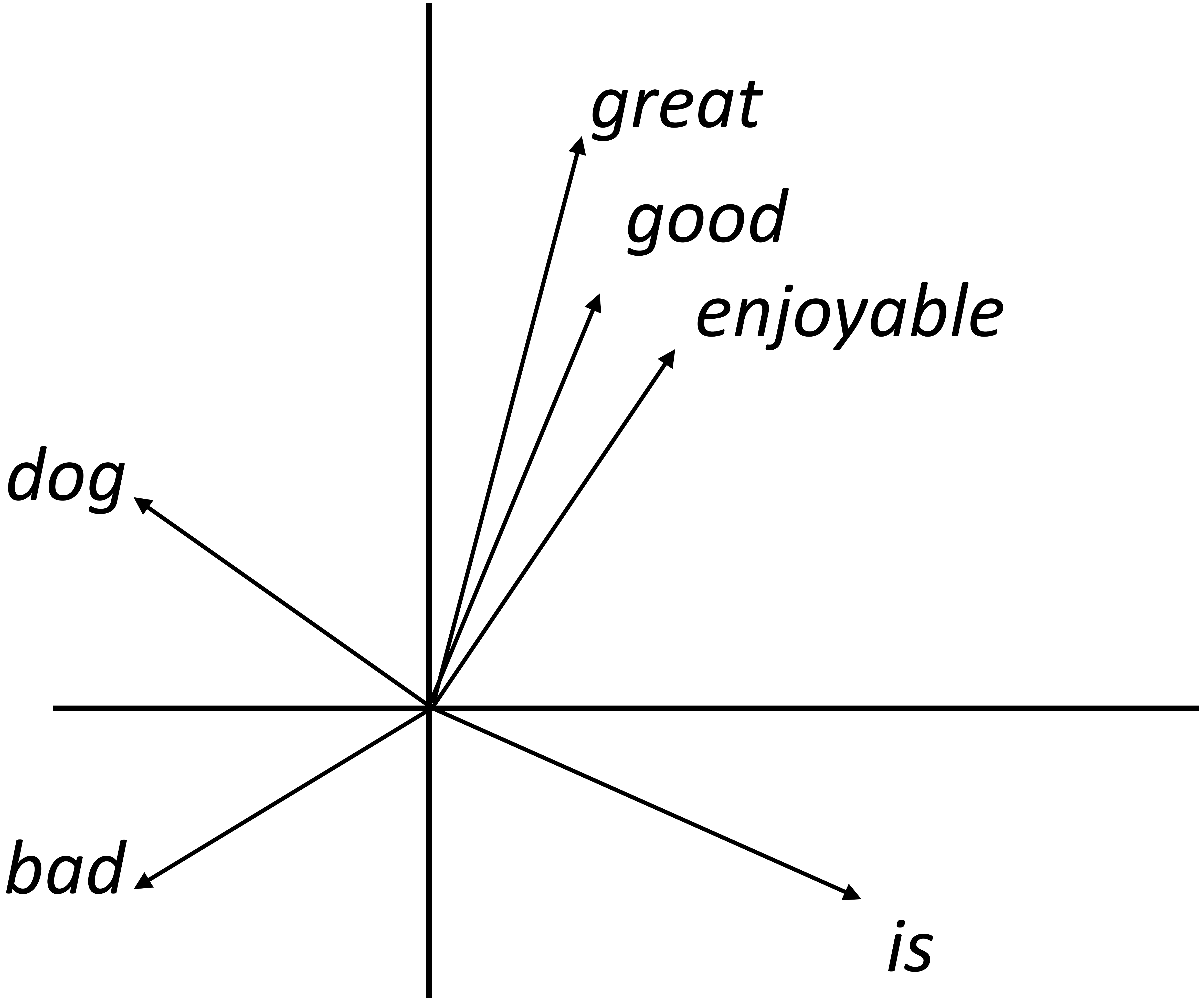
- Want a vector space where similar words have similar embeddings

the movie was great

≈

the movie was good

- Goal: come up with a way to produce these embeddings
- For each word, want "medium" dimensional vector (50-1000 dim) representing it

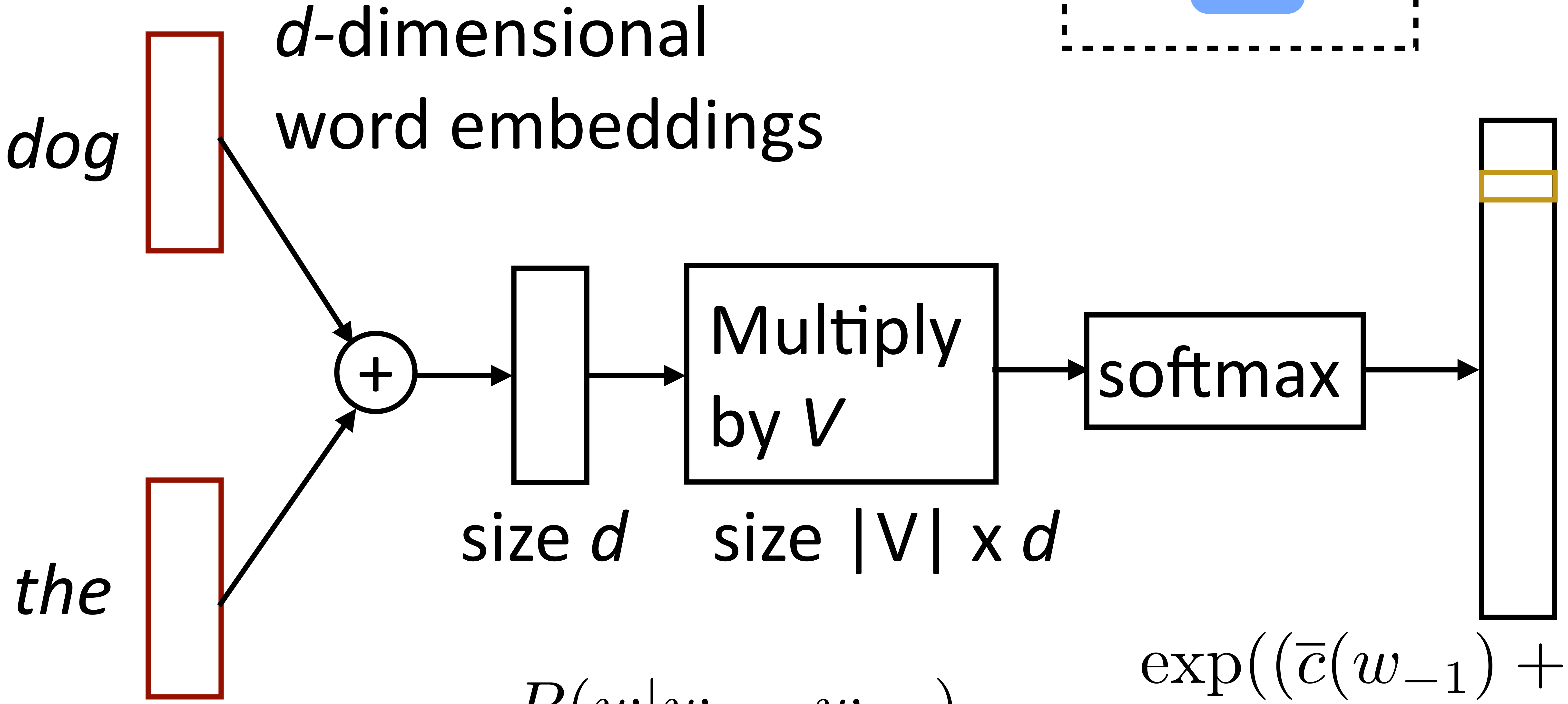


word2vec/GloVe

Continuous Bag-of-Words

- Predict word from context

*the dog **bit** the man*



gold label = *bit*, no manual labeling required!

$$P(w|w_{-1}, w_{+1}) = \frac{\exp((\overline{c}(w_{-1}) + \overline{c}(w_{+1})) \cdot \overline{v}(w))}{\sum_{w'} \exp((\overline{c}(w_{-1}) + \overline{c}(w_{+1})) \cdot \overline{v}(w'))}$$

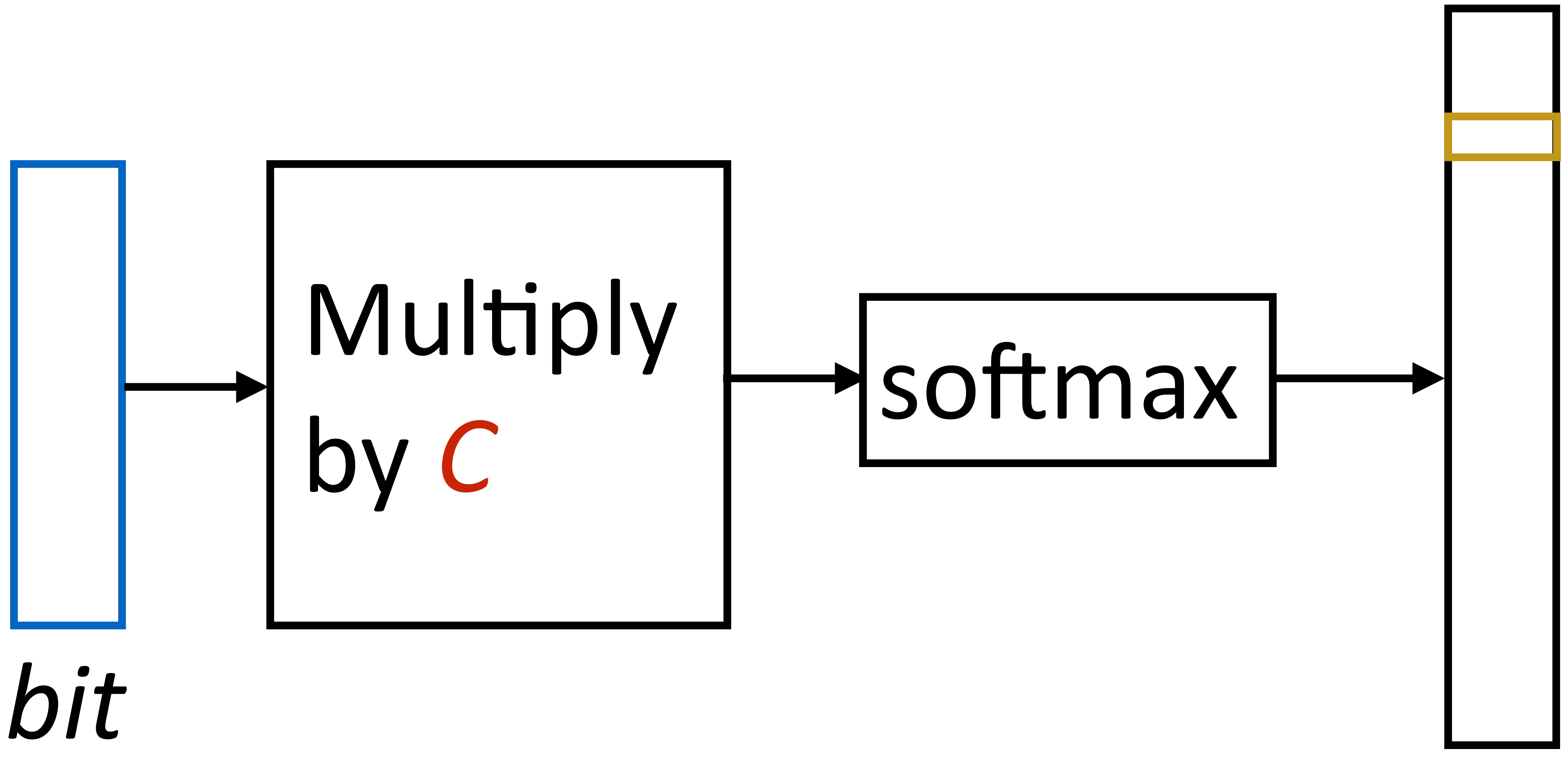
- Parameters: *d* x |V| (one *d*-length context vector *c* per voc word), |V| x *d* output parameters (*V*)

Mikolov et al. (2013)

Skip-Gram

- Predict one word of context from word

the dog bit the man



gold = *dog*

$$P(w_{-1}|w) = \frac{\exp((\overline{c}(w_{-1}) \cdot \overline{v}(w))}{\sum_{w'} \exp((\overline{c}(w_{-1}) \cdot \overline{v}(w'))}$$

- Another training example: *bit* -> *the*
- Parameters: *d* x |V| vectors, |V| x *d* context vectors (*C*) (also usable as vectors!)

Mikolov et al. (2013)

Problems with Skip-Gram

- CBOW: $P(w|w_{-1}, w_{+1}) = \frac{\exp((\overline{c}(w_{-1}) + \overline{c}(w_{+1})) \cdot \overline{v}(w))}{\sum_{w'} \exp((\overline{c}(w_{-1}) + \overline{c}(w_{+1})) \cdot \overline{v}(w'))}$

- Skip-gram: $P(w_{-1}|w) = \frac{\exp((\overline{c}(w_{-1}) \cdot \overline{v}(w))}{\sum_{w'} \exp((\overline{c}(w_{-1}) \cdot \overline{v}(w'))}$

- Computing the denominator is extremely slow. How many operations?

- Very slow if we want to train on a ton of data!

- How can we make this faster?

Mikolov et al. (2013)

Skip-Gram with Negative Sampling

- Take (word, context) pairs and classify them as "real" or not. Create random negative examples by sampling from unigram distribution

(*bit*, *the*) => +1

(*bit*, *cat*) => -1

(*bit*, *a*) => -1

(*bit*, *fish*) => -1

$$P(y = 1|w, c) = \frac{e^{w \cdot c}}{e^{w \cdot c} + 1}$$

words in similar contexts select for similar *c* vectors

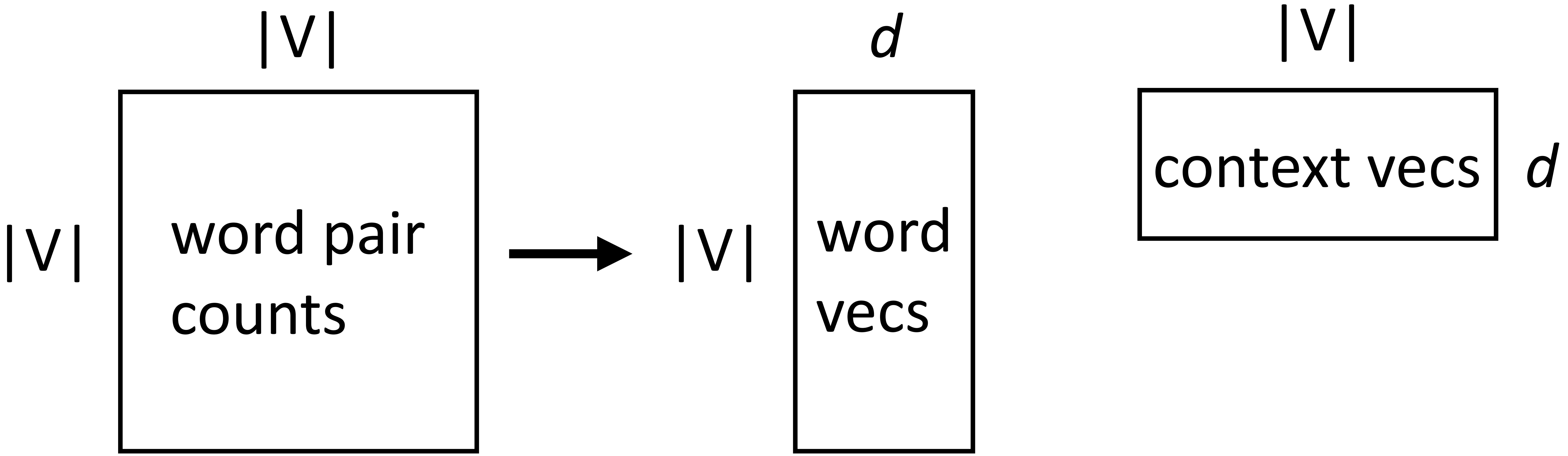
- *d* x |V| vectors, *d* x |V| context vectors (same # of params as before)

- Treat as a binary classification problem

Mikolov et al. (2013)

Connections with Matrix Factorization

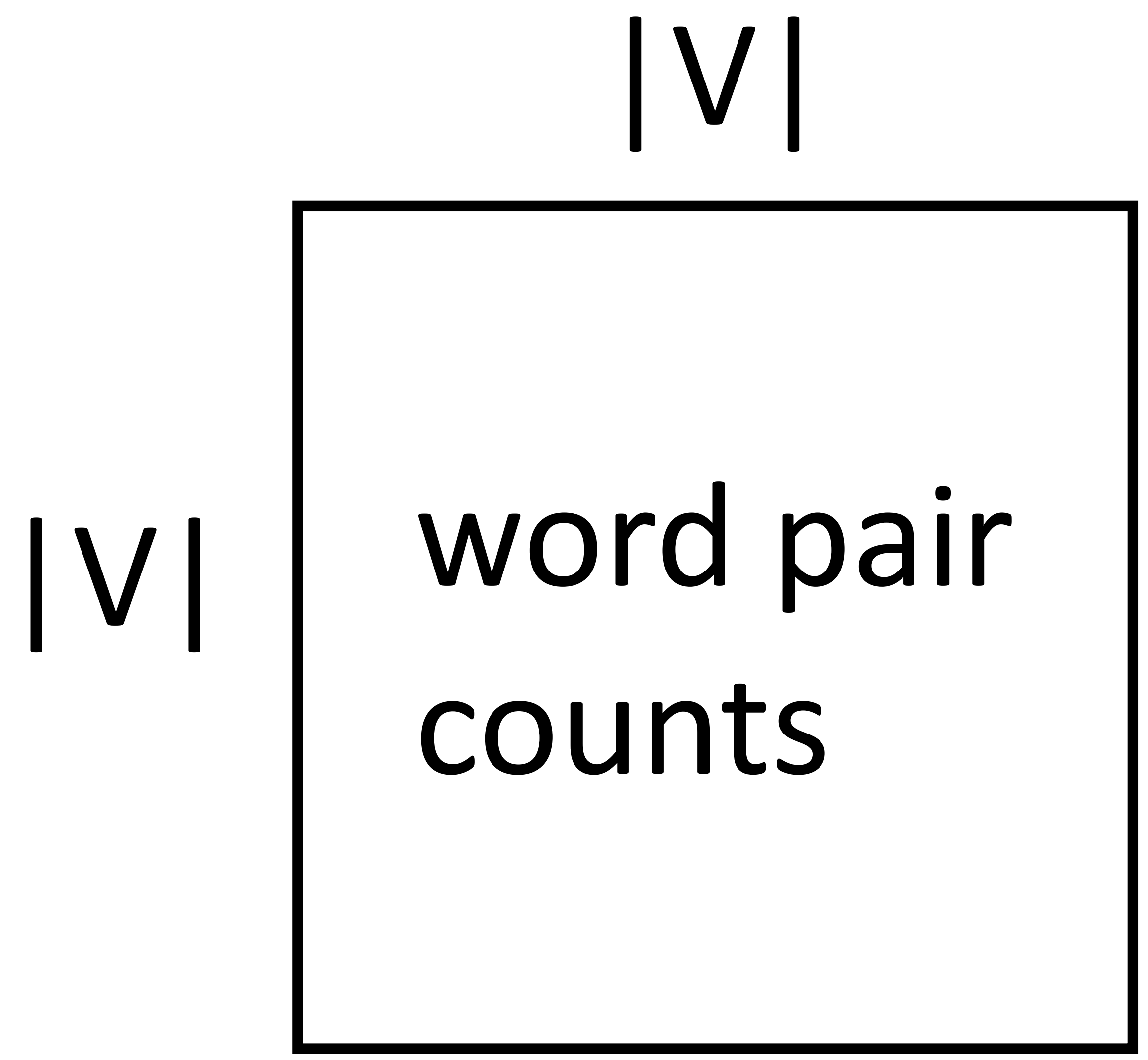
- Skip-gram model looks at word-word co-occurrences and produces two types of vectors



- There is an interpretation as matrix factorization, so this is one way to think about it

Levy et al. (2014)

GloVe (**Glo**bal **Ve**ctors)



- Also operates on counts matrix, weighted regression on the log co-occurrence matrix
- Learn word vectors so you can predict the expected count of a word pair based on the two word vectors
- *Constant* in the dataset size (just need counts), quadratic in voc size
- By far the most common word vectors used today (10,000+ citations)

Pennington et al. (2014)

fastText: Sub-word Embeddings

- Same as SGNS, but break words down 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 $w \cdot c$ in skip-gram computation with $\left(\sum_{g \in \mathrm{ngrams}} w_g \cdot c\right)$

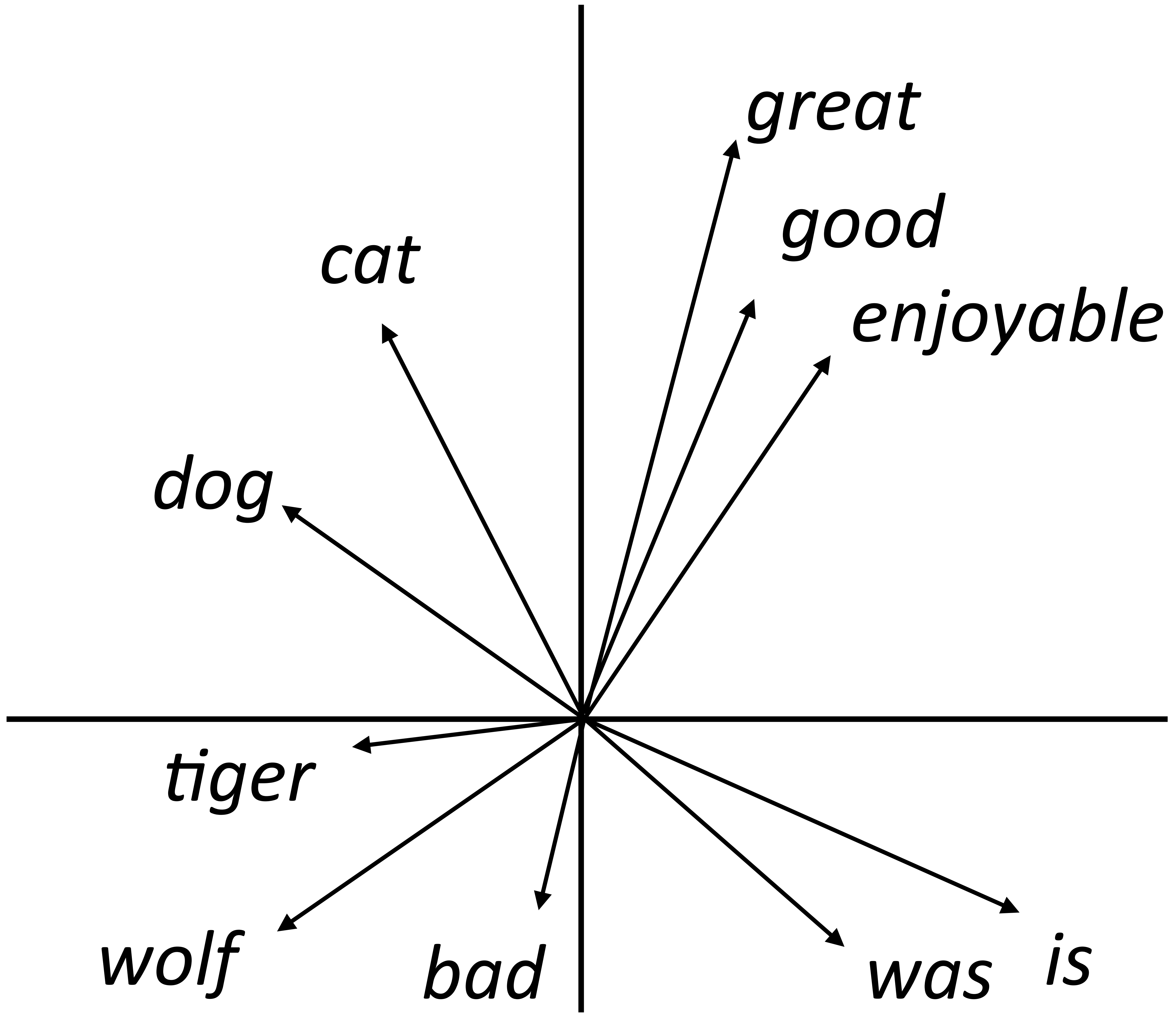
- Useful for generalizing to unknown words in practice

Bojanowski et al. (2017)

Evaluation

Evaluating Word Embeddings

- What properties of language should word embeddings capture?
- Similarity: similar words are close to each other
- Can measure correlation with human judgments
- Mostly word embeddings are evaluated on "downstream" tasks



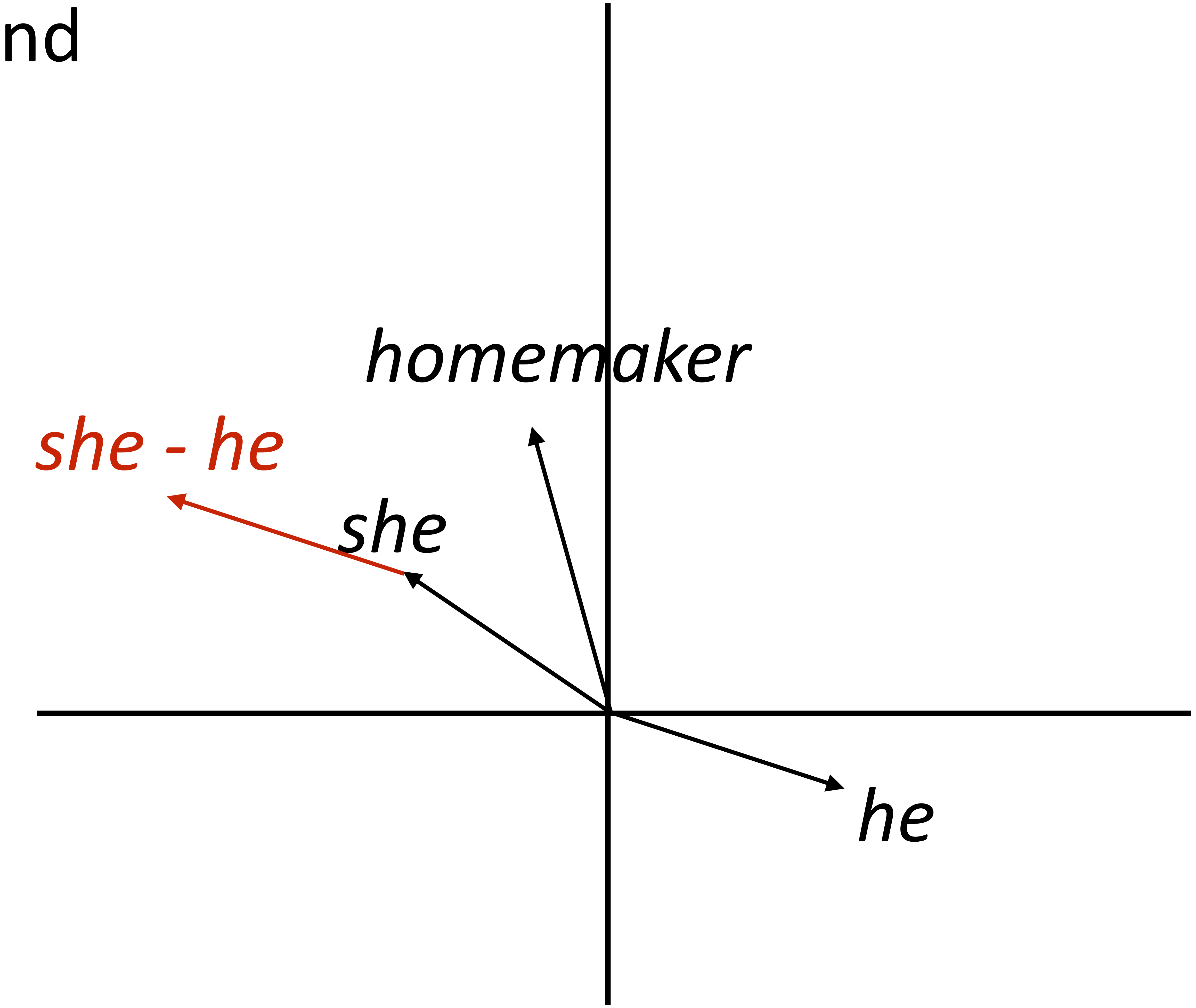
What can go wrong with word embeddings?

- What's wrong with learning a word's "meaning" from its usage?
  - What data are we learning from?
  - What are we going to learn from this data?
- Word embeddings can end up encoding real-world bias

Identifying Bias

- Which words are closest to *she* and farthest from *he*? Or vice versa?

- Compute the vector *she - he*, find words which have high dot product (either positive or negative) with this as gender-associated words



Bolukbasi et al. (2016)

Identifying Bias

- Identify *she* - *he* axis in word vector space, project words onto this axis
- What have we learned from our data?
- What implications will this have in a real system?

Extreme *she* occupations

1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

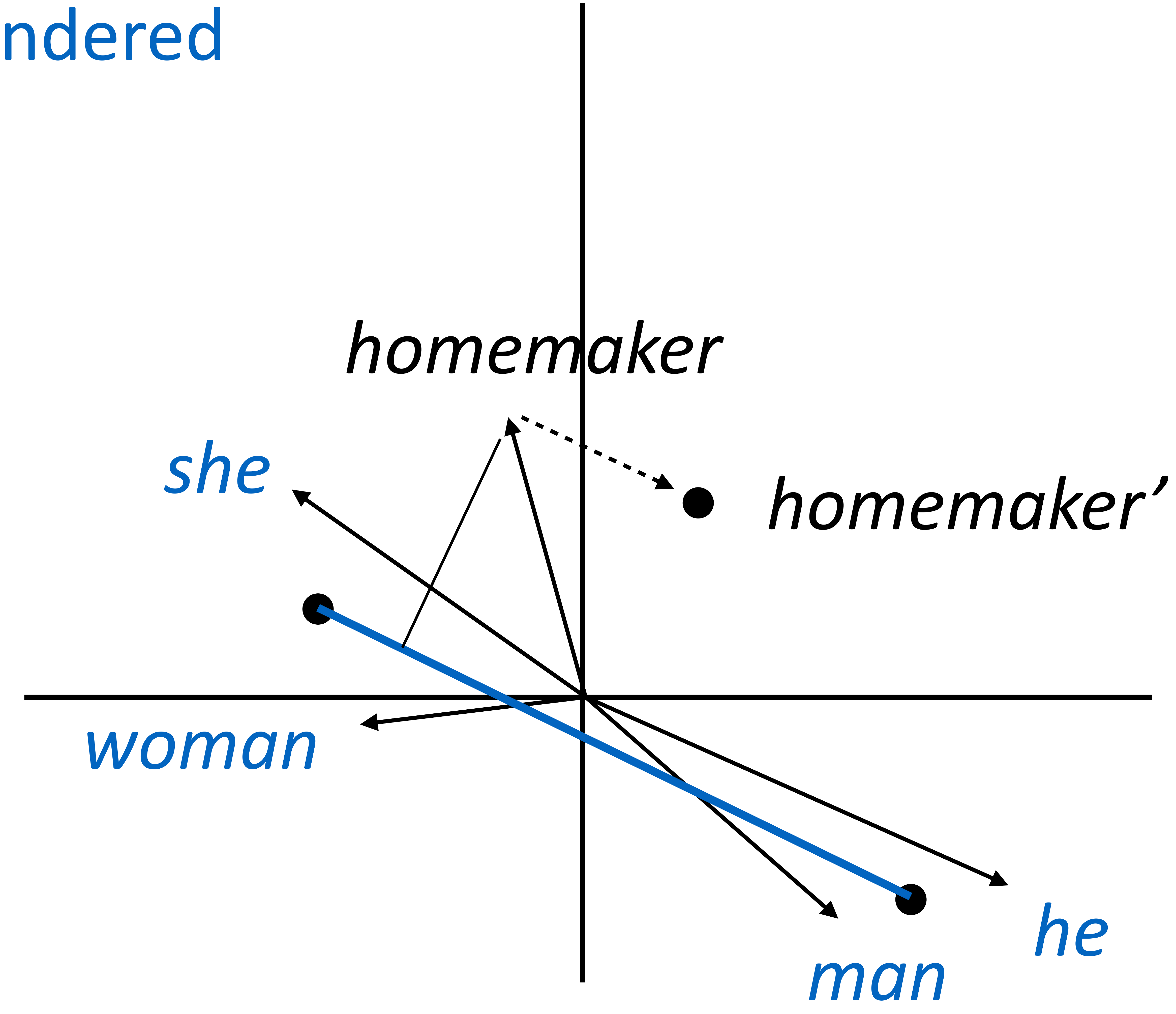
Extreme *he* occupations

1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss

Bolukbasi et al. (2016)

Debiasing

- Identify gender subspace with gendered words
- Project words onto this subspace
- Subtract those projections from the original word to form "gender-neutral" representations of those words

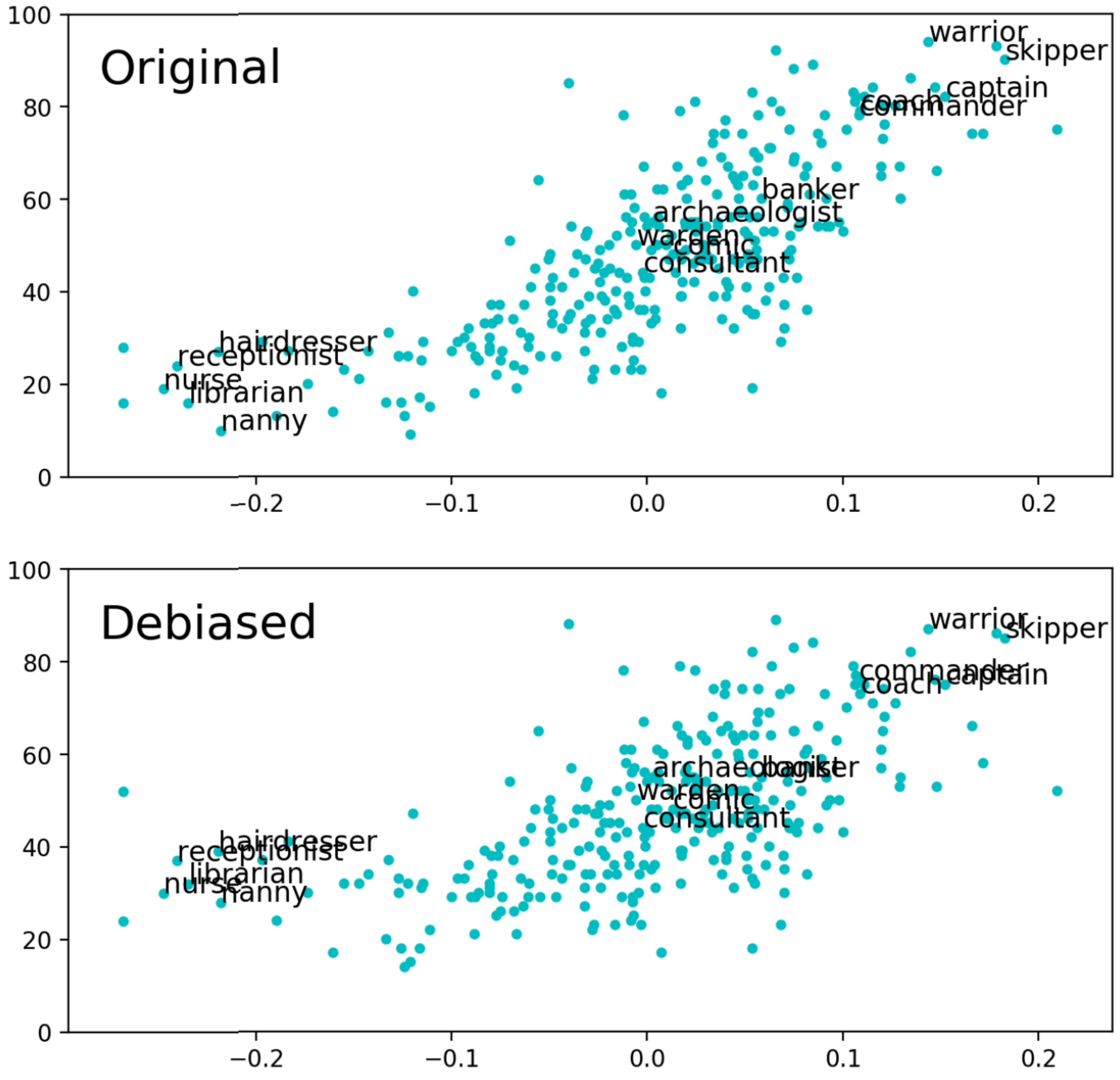


Bolukbasi et al. (2016)

Hardness of Debiasing

- Not that effective…and the male and female words are still clustered together

- Bias pervades the word embedding space and isn't just a local property of a few words



(a) The plots for HARD-DEBIASED embedding, before (top) and after (bottom) debiasing.

Gonen and Goldberg (2019)

Takeaways

- Word vectors: learning word -> context mappings has given way to matrix factorization approaches (constant in dataset size)
- Lots of pretrained embeddings work well in practice, they capture some desirable properties
- Bias in data can infiltrate word embeddings, hard to resolve