CS378: Natural Language Processing

Lecture 11: Word Embeddings

Eunsol Choi

Slides adapted from Greg Durrett, Princeton NLP course

Representation of Words

- Traditional approaches build one-hot feature vectors.

$$\text{truck} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]$$
$$\text{austin} = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

- The dimension of the vector: the size of the vocabulary!
- More over, no generalization….

Distributional Semantics

“tejuino”

Context 1: A bottle of ___ is on the table.

Context 2: Everybody likes ___.

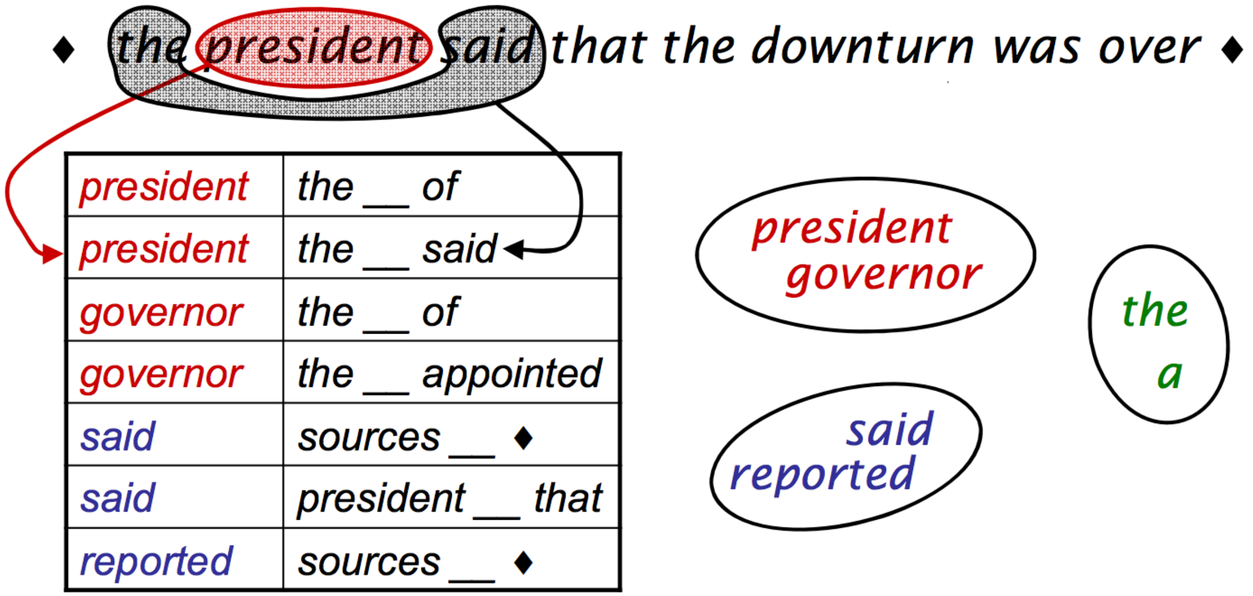
Context 3: Don’t have ___ before you drive.

Context 4: We make ___ out of corn.

the example is from Lin, 1998

Distributional Semantics

- "You shall know a word by the company it keeps" Firth (1957)



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

slide credit: Dan Klein

Goal of Word Embeddings

- Learn a ***continuous, dense*** vector for each word type.
- Always the same vector, regardless of in which context the word appears

$$\text{employees} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 10.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

Earlier Approach: Discrete Word Clusters

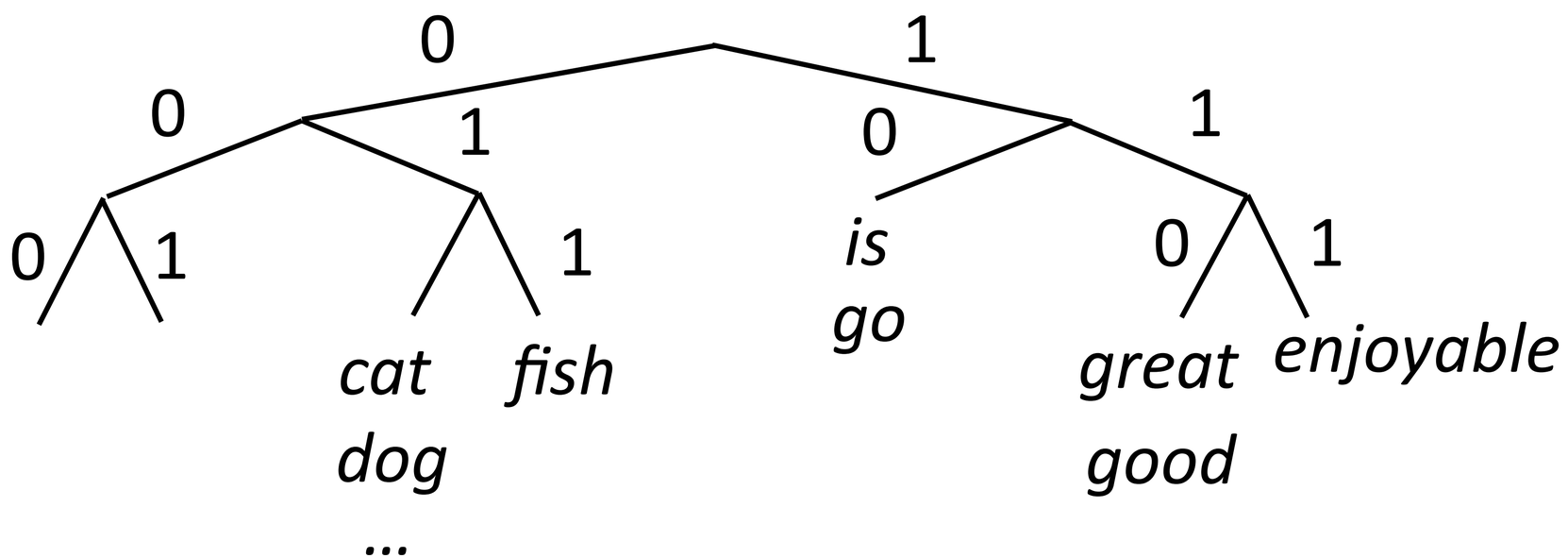
- Brown clusters: hierarchical agglomerative *hard* clustering (each word has one cluster)

The cats are under the table.

Word ids $w_1, w_2, w_3, w_4, w_5, w_6$

100111 100111 1001111 111011 110 10010

Cluster ids $c_1, c_2, c_3, c_4, c_5, c_6$



- Goal is to maximize

$$P(w_i|w_{i-1}) = P(c_i|c_{i-1})P(w_i|c_i)$$

lawyer	1000001101000
newspaperman	100000110100100
stewardess	100000110100101
toxicologist	10000011010011
slang	1000001101010
babysitter	100000110101100
conspirator	1000001101011010
womanizer	1000001101011011
mailman	10000011010111
salesman	100000110110000
bookkeeper	1000001101100010
troubleshooter	10000011011000110
bouncer	10000011011000111
technician	1000001101100100
janitor	1000001101100101
saleswoman	1000001101100110
...	
Nike	10110111001001010111100
Maytag	101101110010010101111010
Generali	10110111001001010111011
Gap	1011011100100101011110
Harley-Davidson	10110111001001010111110
Enfield	101101110010010101111110
genus	101101110010010101111111
Microsoft	10110111001001011000
Ventritex	1011011100100101110010
Tractebel	10110111001001011001110
Synopsys	10110111001001011001111
WordPerfect	10110111001001011010000

Brown et al. (1992)

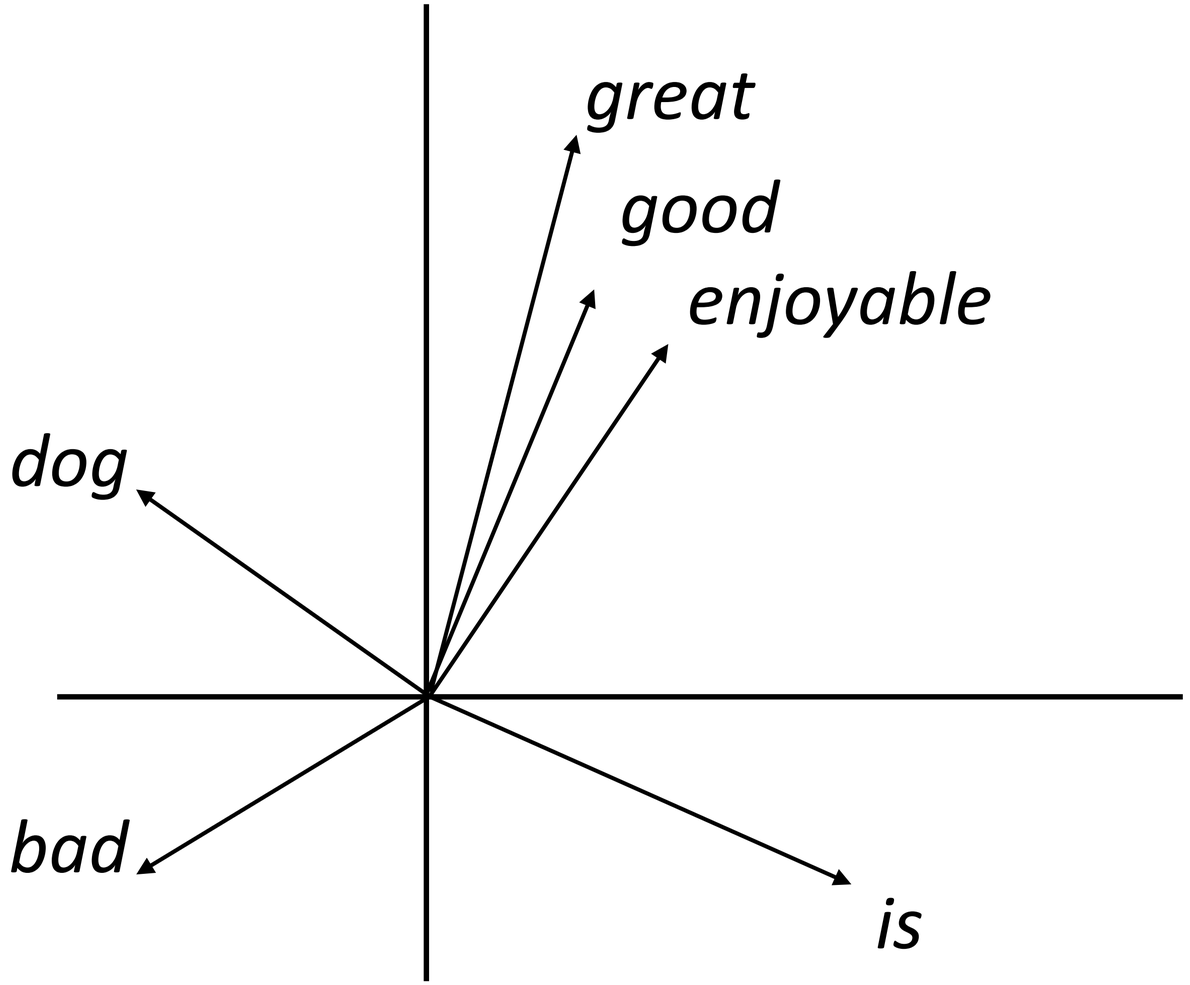
Word Embeddings

the movie was great

≈

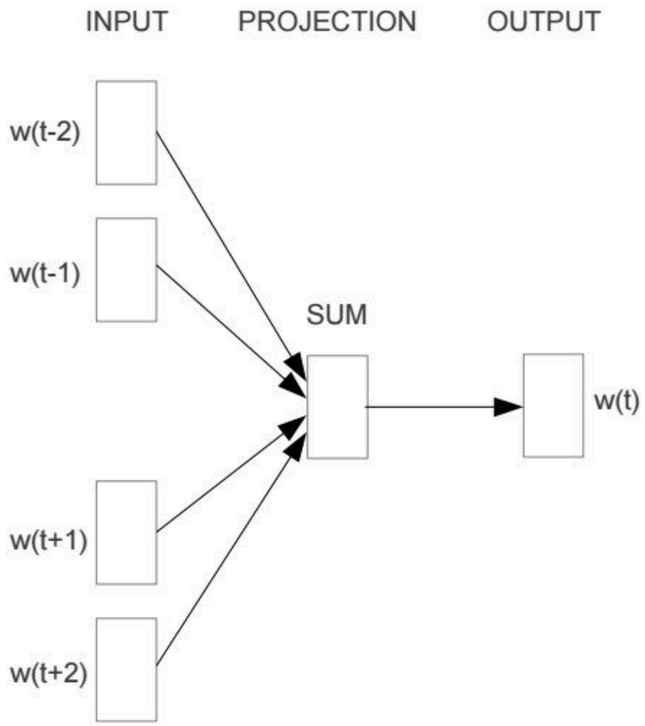
the movie was good

- Goal: come up with a way to produce these embeddings
- For each word, “medium” dimensional vector (50-300 dims) representing it

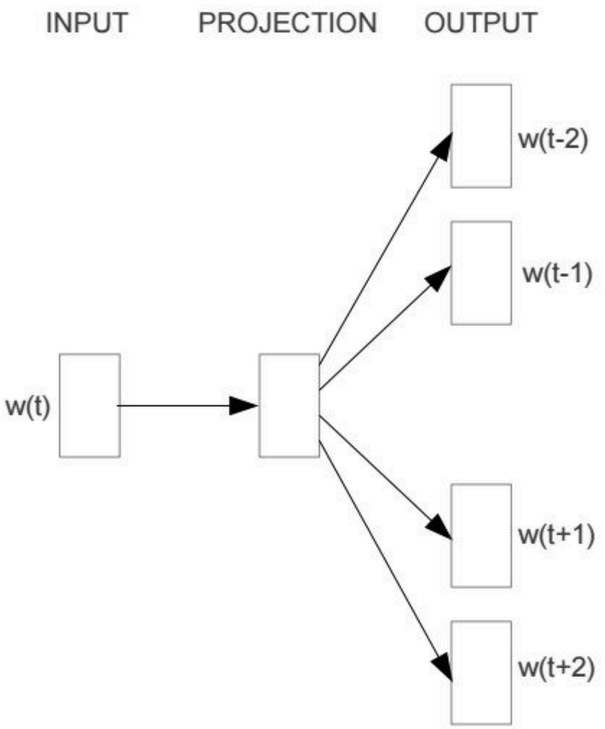


word2vec / GloVe (**Glo**bal **Ve**ctors)

word2vec



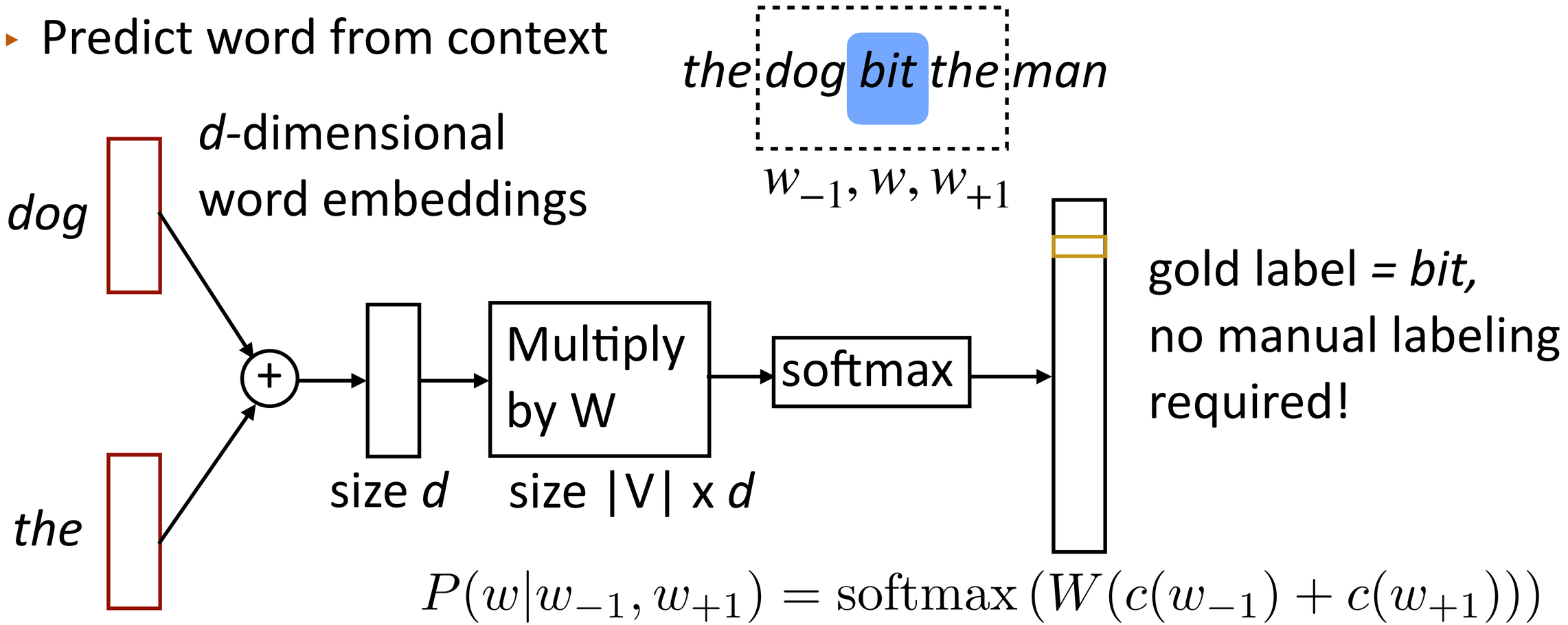
Continuous Bag of Words (CBOW)



Skip-grams

Continuous Bag-of-Words

- Predict word from context



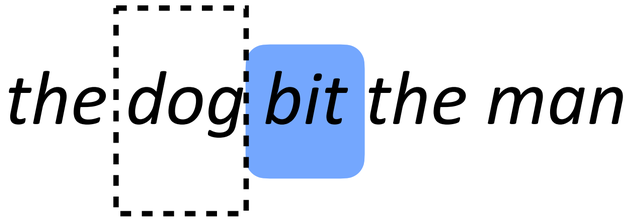
$$P(w|w_{-1}, w_{+1}) = \text{softmax}\left(W(c(w_{-1}) + c(w_{+1}))\right)$$

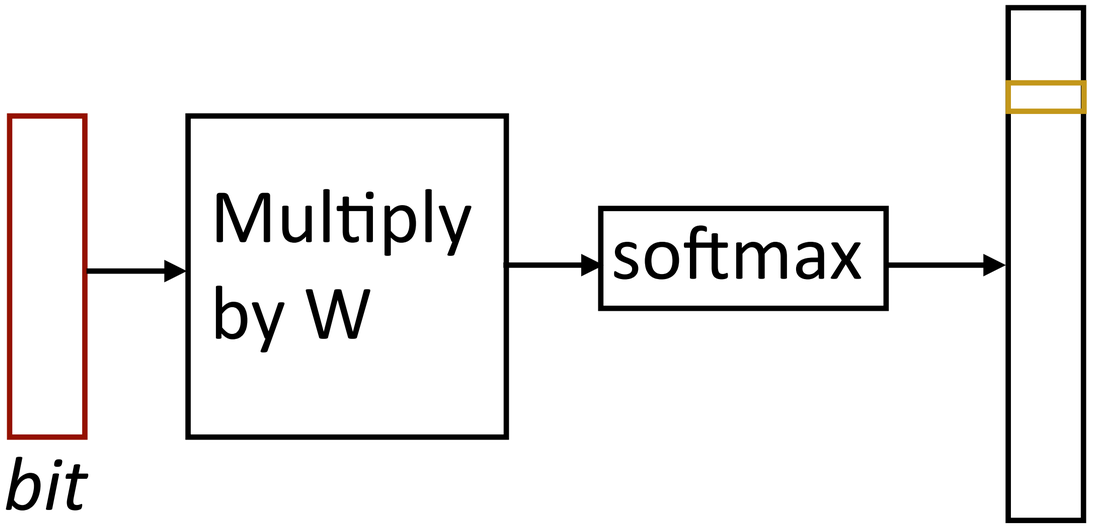
- Parameters: *d* x |V| (one *d*-length context vector per word),
  |V| x *d* output parameters (W)

Mikolov et al. (2013)

Skip-Gram

- Predict one word of context from word





gold = *dog*

$$P(w'|w) = \text{softmax}(We(w))$$

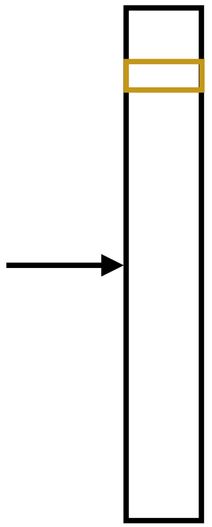
- Another training example: *bit* -> *the*
- Parameters: *d* x |V| word vector, |V| x *d* output parameters (W) (also usable as vectors!)

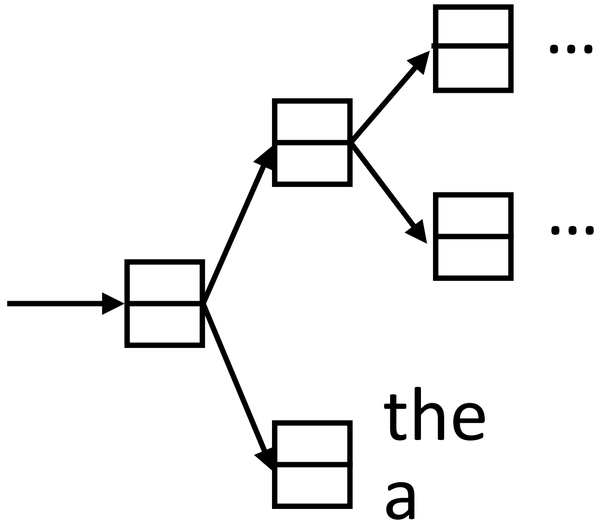
Mikolov et al. (2013)

Hierarchical Softmax

$$P(w|w_{-1}, w_{+1}) = \text{softmax}\left(W(c(w_{-1}) + c(w_{+1}))\right) \qquad P(w'|w) = \text{softmax}(We(w))$$

- Matmul + softmax over |V| is very slow to compute for CBOW and SG





- Binary classifiers to decide which branch to take
- log(|V|) binary decisions

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

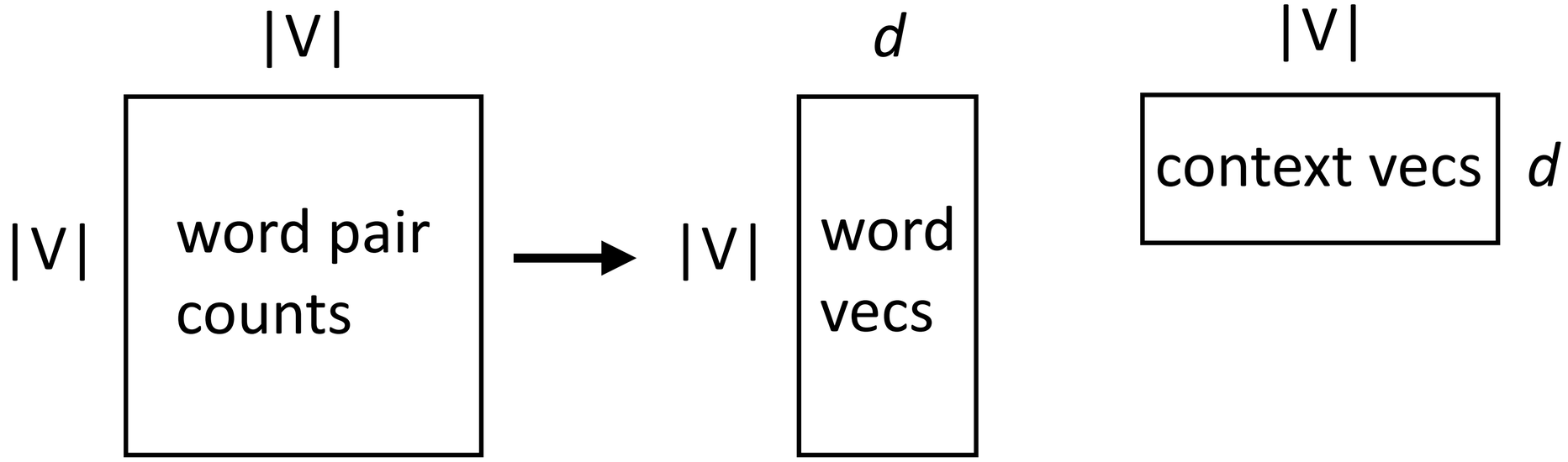
- *d* x |V| vectors, *d* x |V| context vectors (same # of params as before)
- Objective = $\log P(y=1\,|\,w,c) + \frac{1}{k}\sum_{i=1}^{k} \log P(y=0\,|\,w,c_i)$

sampled

Mikolov et al. (2013)

Connections with Matrix Factorization

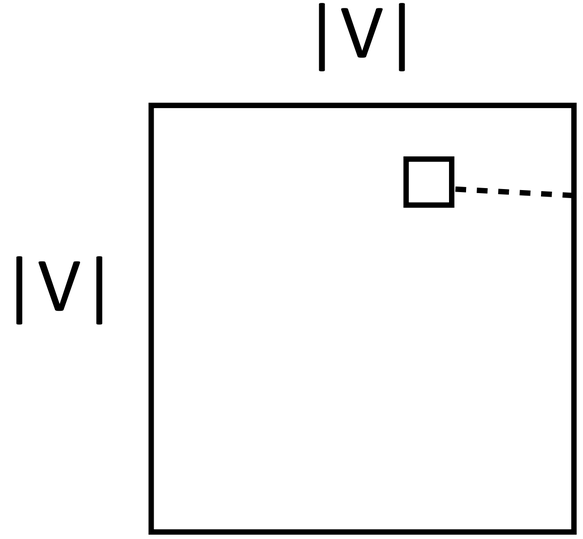
- Skip-gram model looks at word-word co-occurrences and produces two types of vectors



- Looks almost like a matrix factorization… can we interpret it this way?

Levy et al. (2014)

Skip-Gram as Matrix Factorization



Pointwise mutual linformation

num negative samples

$$M_{ij} = \text{PMI}(w_i, c_j) - \log k$$

$$\text{PMI}(w_i, c_j) = \frac{P(w_i, c_j)}{P(w_i)P(c_j)} = \frac{\frac{\text{count}(w_i, c_j)}{D}}{\frac{\text{count}(w_i)}{D}\frac{count(c_j)}{D}}$$

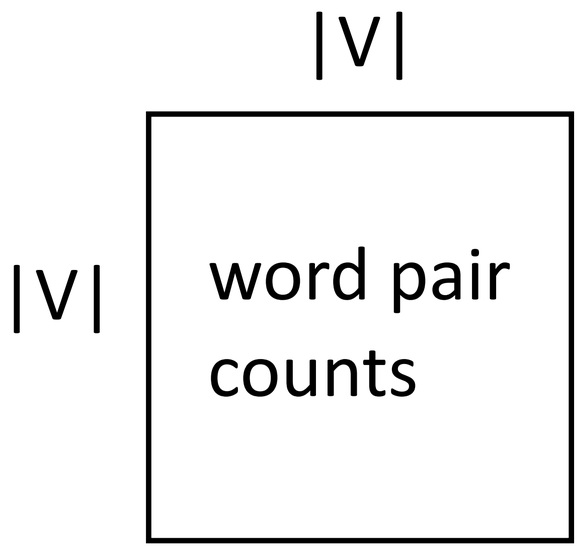
Skip-gram objective *exactly* corresponds to factoring this matrix:

- *If* we sample negative examples from the uniform distribution over words
- …and it's a *weighted* factorization problem (weighted by word freq)

Levy et al. (2014)

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

- Also operates on counts matrix, weighted regression on the log co-occurrence matrix



- Objective = $\sum_{i,j} f(\text{count}(w_i, c_j)) \left(w_i^\top c_j + a_i + b_j - \log \text{count}(w_i, c_j)\right)^2$
- Constant in the dataset size (just need counts), quadratic in voc size

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 $\left(\sum_{g \in \text{ngrams}} w_g \cdot c \right)$

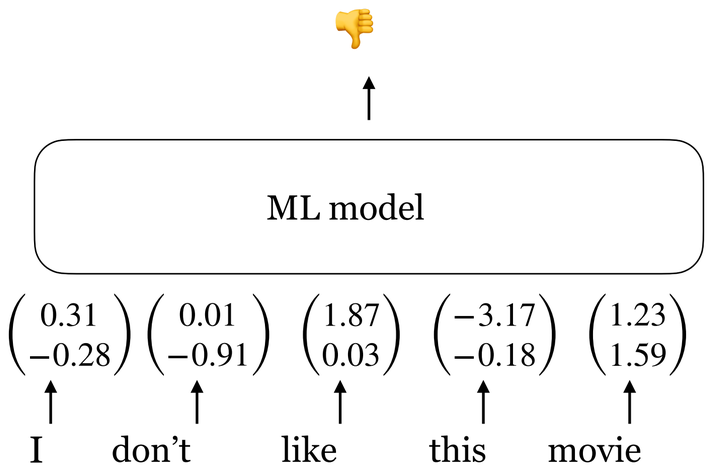
- Advantages?

Bojanowski et al. (2017)

Evaluating Word Embeddings

Extrinsinc vs. Intrinsic

- Plug the vectors into some end-task model, see which does well!



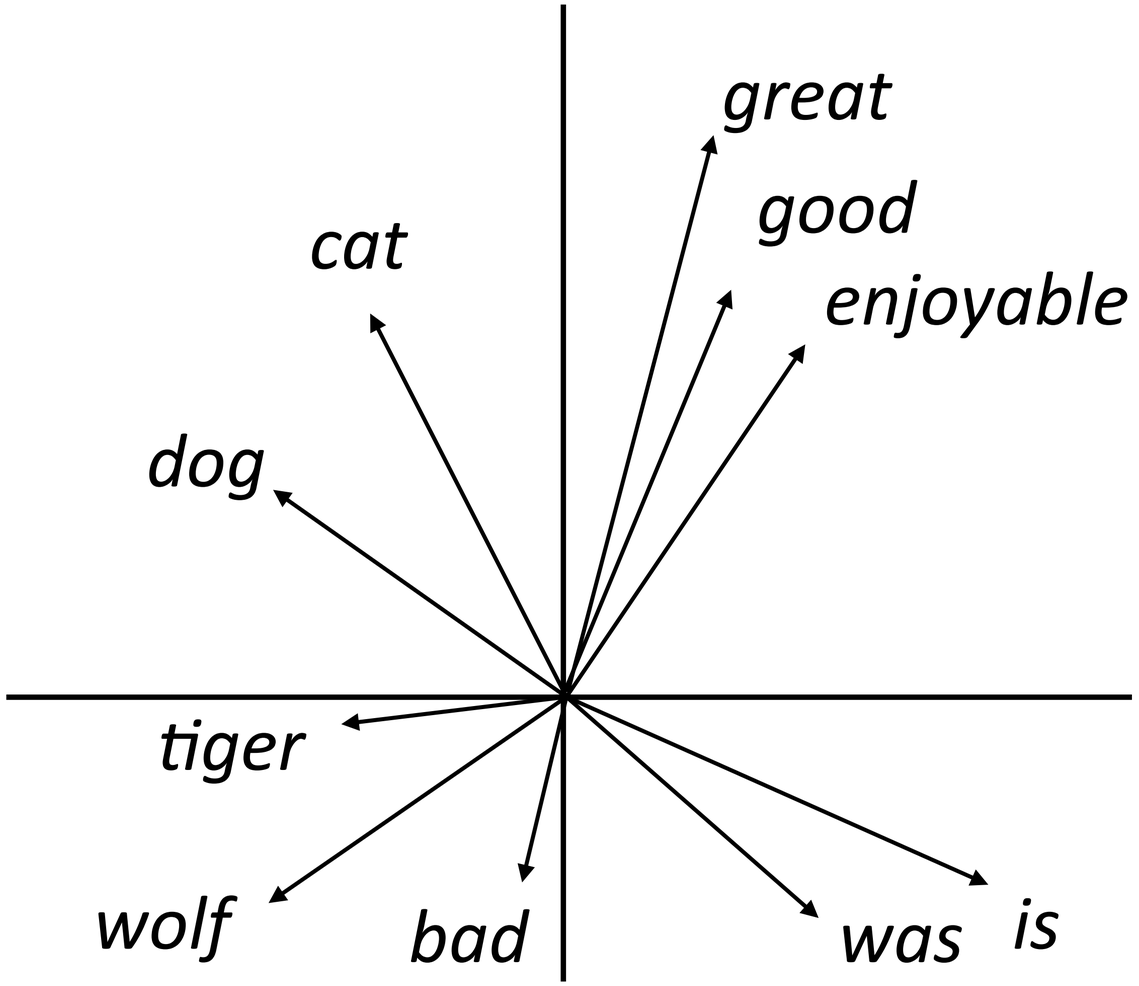
- Evaluate on some intermediate subtasks (in the coming slides..)

Evaluating Word Embeddings

- What properties of language should word embeddings capture?
- (1) Similarity: similar words are close to each other
- (2) Analogy:

  good is to best as smart is to ???

  Paris is to France as Tokyo is to ???



Similarity

Method	WordSim Similarity	WordSim Relatedness	Bruni et al. MEN	Radinsky et al. M. Turk	Luong et al. Rare Words	Hill et al. SimLex
PPMI	.755	**.697**	.745	.686	.462	.393
SVD	**.793**	.691	**.778**	.666	**.514**	.432
SGNS	**.793**	.685	.774	**.693**	.470	**.438**
GloVe	.725	.604	.729	.632	.403	.398

- SVD = singular value decomposition on PMI matrix
- GloVe does not appear to be the best when experiments are carefully controlled, but it depends on hyperparameters + these distinctions don't matter in practice

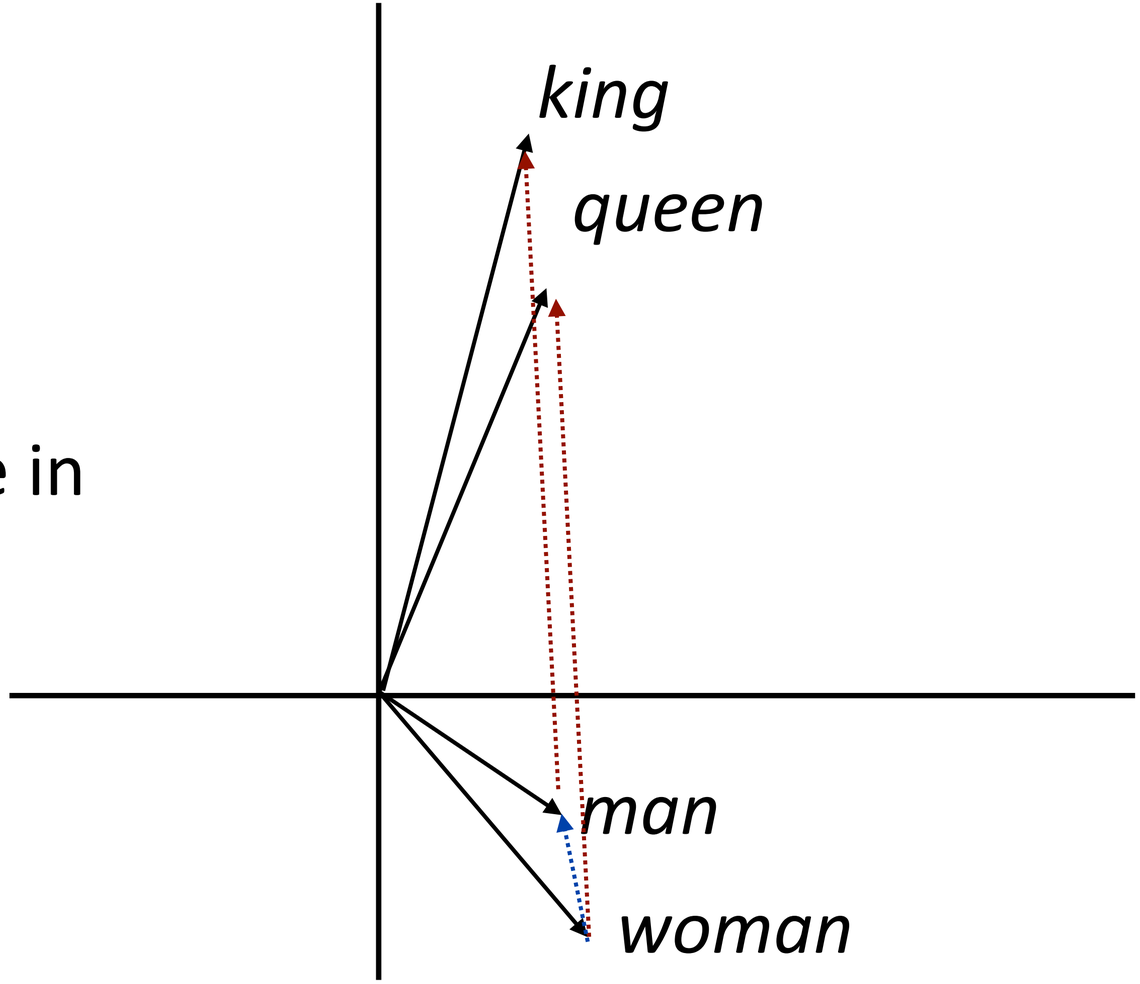
Levy et al. (2015)

Analogies

(king - man) + woman = queen

king + (woman - man) = queen

- Why would this be?
- woman - man captures the difference in the contexts that these occur in
- Often used to evaluate word embeddings



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?

What do we mean by bias?

- Identify *she* - *he* axis in word vector space, project words onto this axis

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

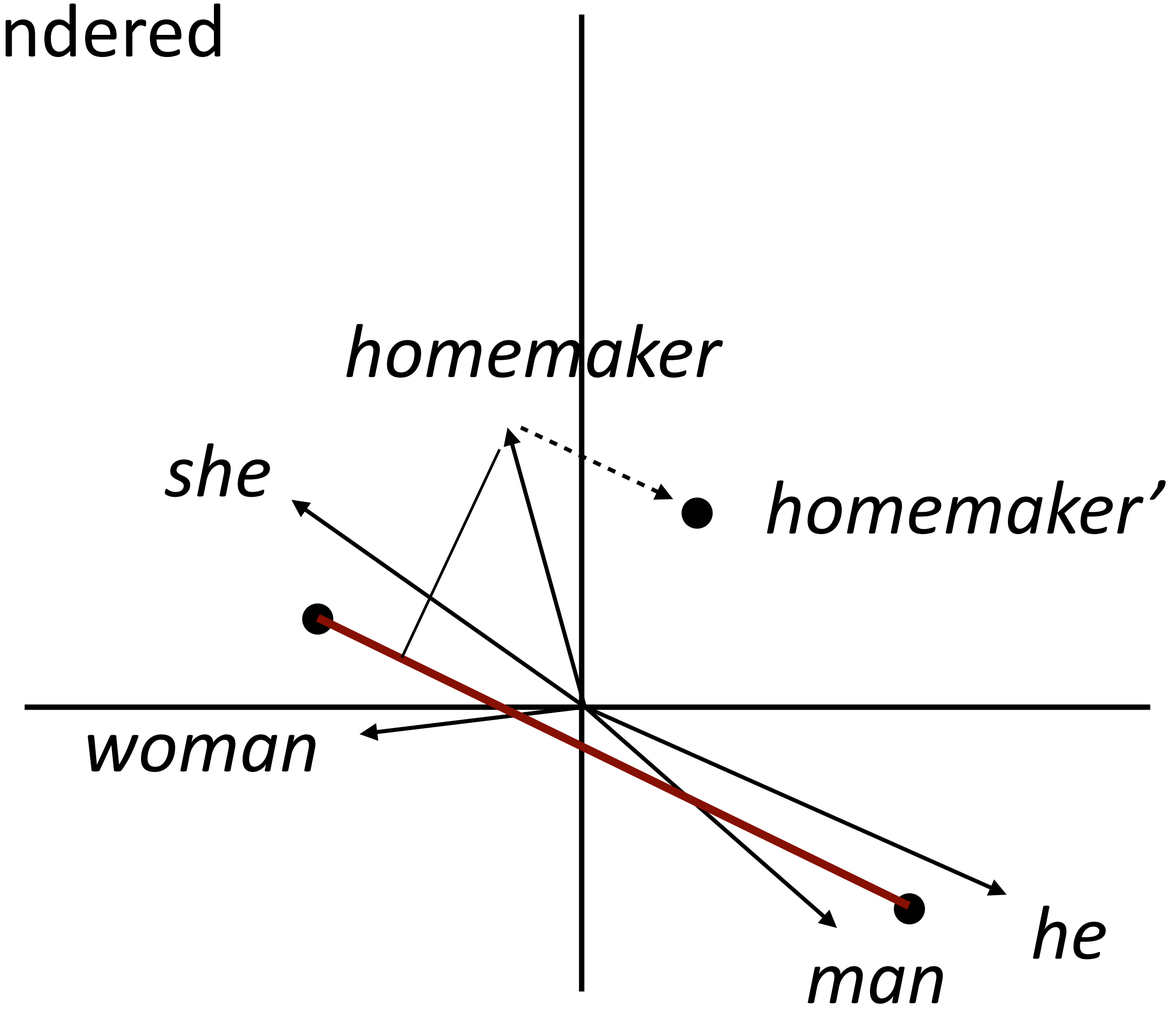
Extreme *he* occupations

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

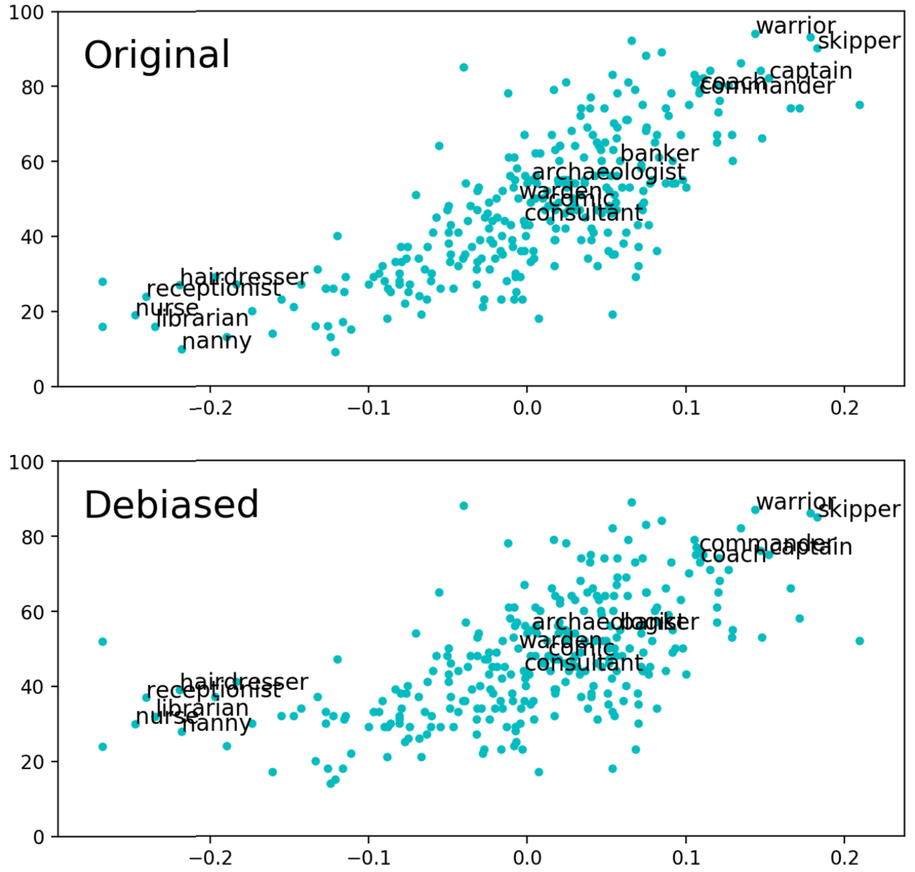


Bolukbasi et al. (2016)

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)

Trained Word Embeddings

- word2vec: https://code.google.com/archive/p/word2vec/
- GloVe: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Using Word Embeddings

- Approach 1: learn embeddings as parameters from your data
  - Often works pretty well
- Approach 2: initialize using pretained word vectors (e.g., GloVe), keep fixed
  - Faster because no need to update these parameters
- Approach 3: initialize using pretained word vectors (e.g., GloVe), fine-tune
  - Works best for some tasks

Preview: Context-dependent Embeddings

- How to handle different word senses? One vector for *balls*

they dance at balls they hit the balls

- Train a neural language model to predict the next word given previous words in the sentence, use its internal representations as word vectors

- *Context-sensitive* word embeddings: depend on rest of the sentence

- *Huge* improvements across nearly all NLP tasks over static word embeddings

Peters et al. (2018)

Compositional Semantics

- What if we want embedding representations for whole sentences?
- Skip-*thought* vectors (Kiros et al., 2015), similar to skip-gram generalized to a sentence level
- Is there a way we can compose vectors to make sentence representations?

Summary

- Lots of pretrained embeddings work well in practice, they capture some desirable properties
- Even better: context-sensitive word embeddings (ELMo)
- Next time: Language Models!