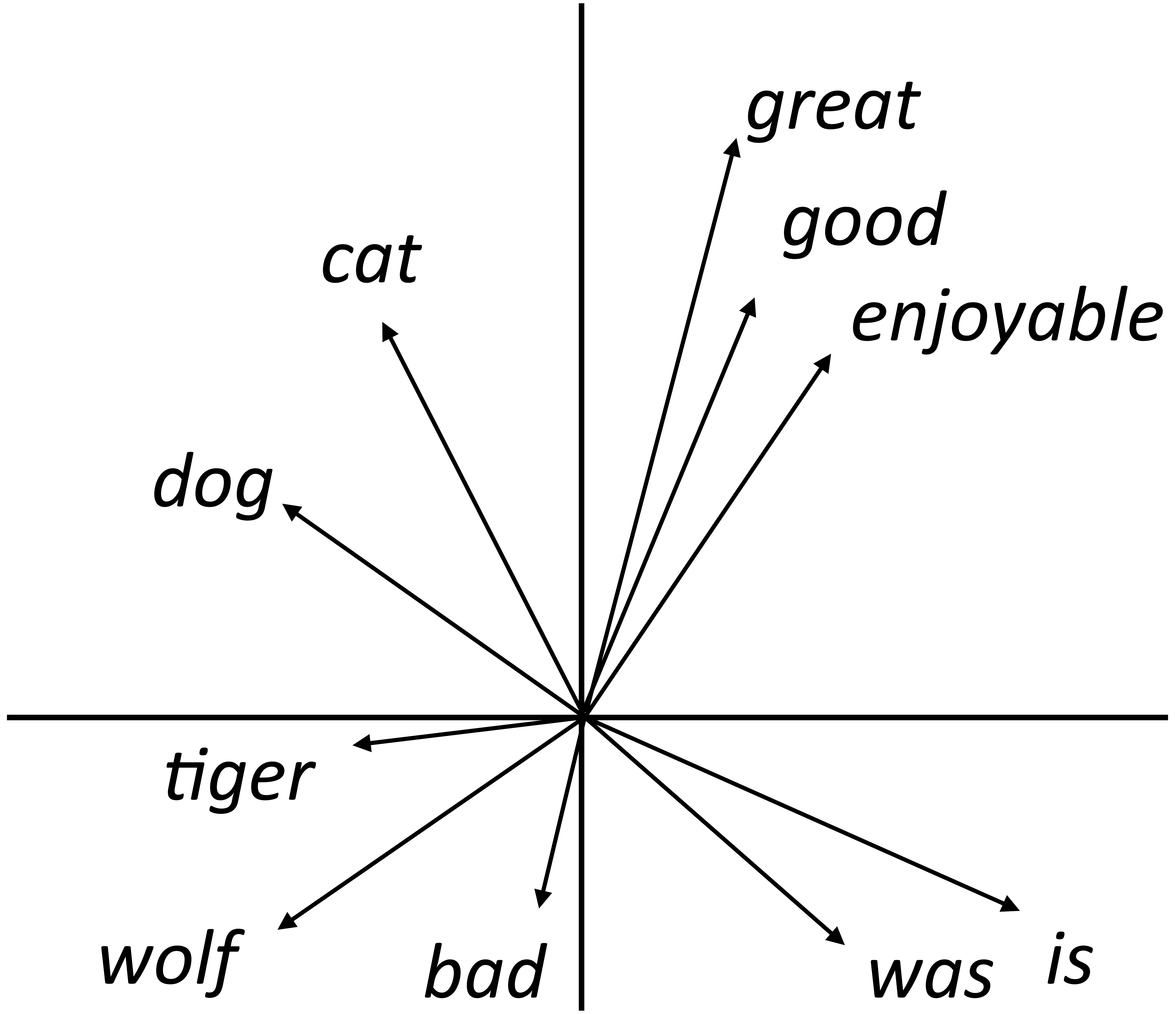
Word Embedding Evaluation

Evaluating Word Embeddings

- What properties of language should word embeddings capture?



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)

Hypernymy Detection

- Hypernyms: detective *is a* person, dog *is a* animal
- Do word vectors encode these relationships?

Dataset	TM14	Kotlerman 2010	HypeNet	WordNet	**Avg (10 datasets)**
Random	52.0	30.8	24.5	55.2	23.2
Word2Vec + C	52.1	**39.5**	20.7	**63.0**	25.3
GE + C	53.9	36.0	21.6	58.2	26.1
GE + KL	52.0	39.4	23.7	54.4	25.9
DIVE + C·ΔS	**57.2**	36.6	**32.0**	60.9	**32.7**

- word2vec (SGNS) works barely better than random guessing here

Chang et al. (2017)

Analogies

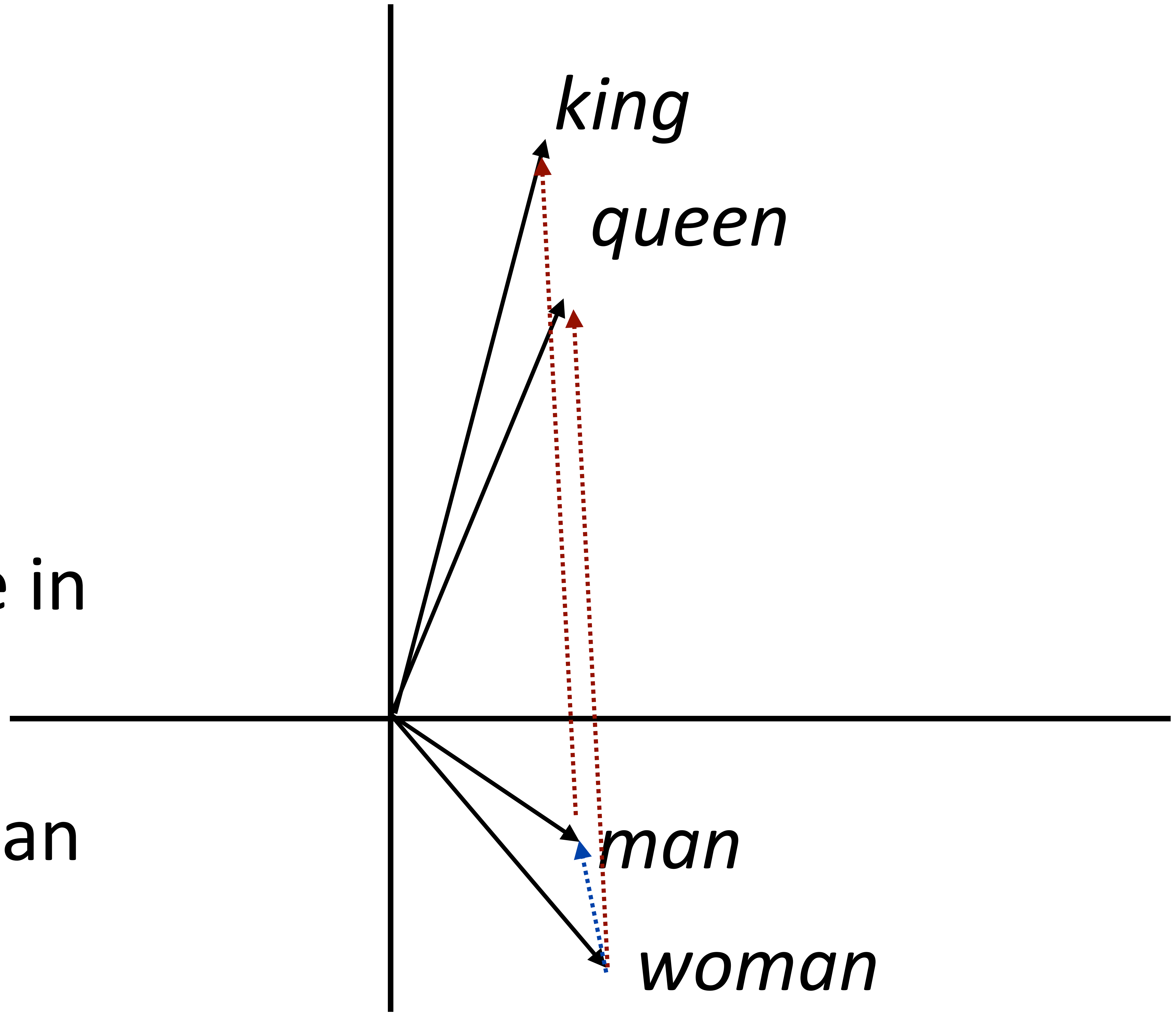
(king - man) + woman = queen

king + (woman - man) = queen

- Why would this be?

- woman - man captures the difference in the contexts that these occur in

- Dominant change: more “he” with man and “she” with woman — similar to difference between king and queen



Analogies

Method	Google Add / Mul	MSR Add / Mul
PPMI	.553 / .679	.306 / .535
SVD	.554 / .591	.408 / .468
SGNS	.676 / **.688**	.618 / **.645**
GloVe	.569 / .596	.533 / .580

- These methods can perform well on analogies on two different datasets using two different methods

Maximizing for *b*: Add = $\cos(b, a_2 - a_1 + b_1)$ Mul = $\dfrac{\cos(b_2, a_2)\cos(b_2, b_1)}{\cos(b_2, a_1) + \epsilon}$

Levy et al. (2015)

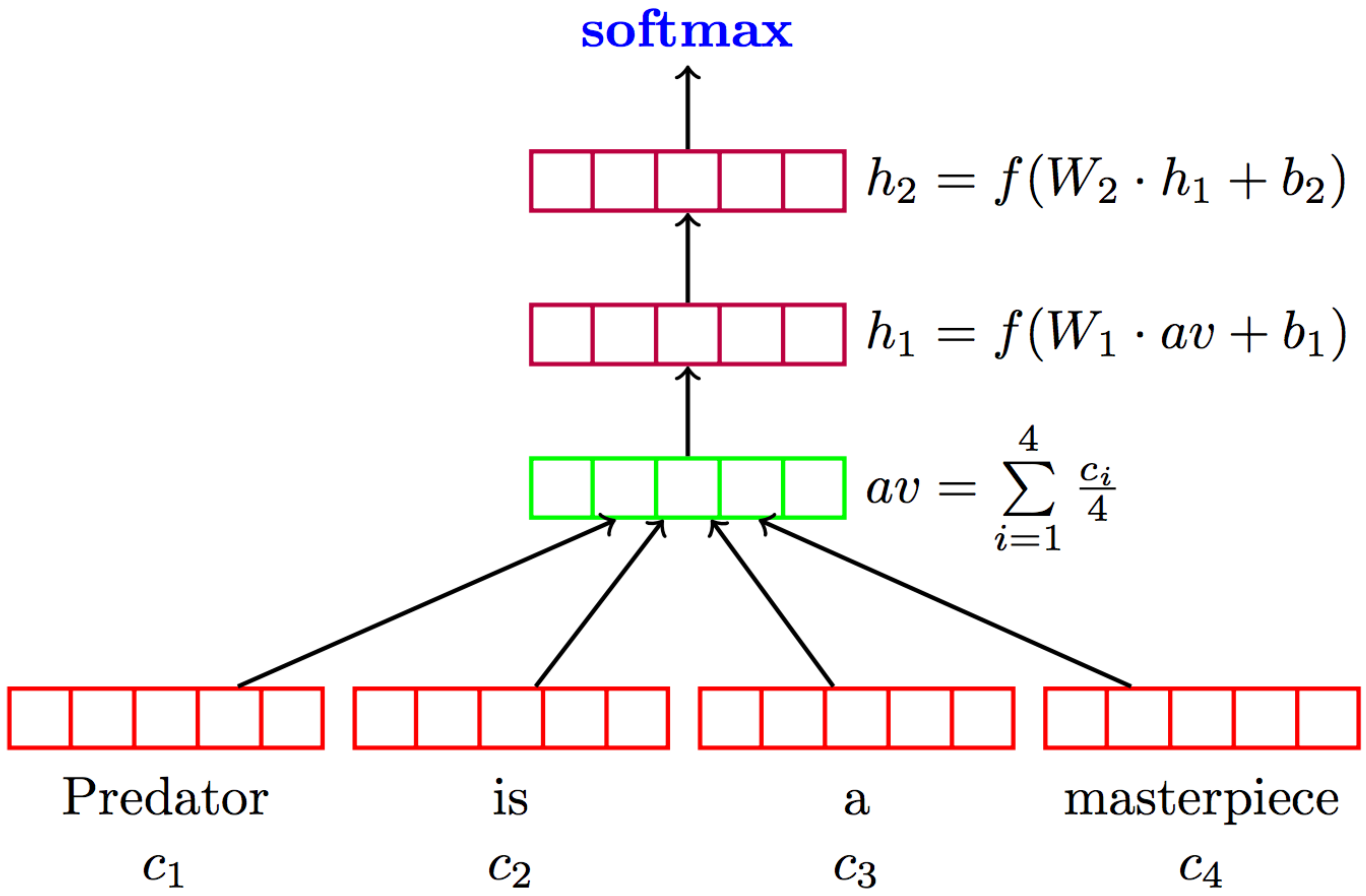
Using Word Embeddings

- Approach 1: learn embeddings as parameters from your data
  - Often works pretty well, especially if data is large
- Approach 2: initialize using GloVe, keep fixed
  - Faster because no need to update these parameters
- Approach 3: initialize using GloVe, fine-tune
  - Usually works the best

DANs

Deep Averaging Networks

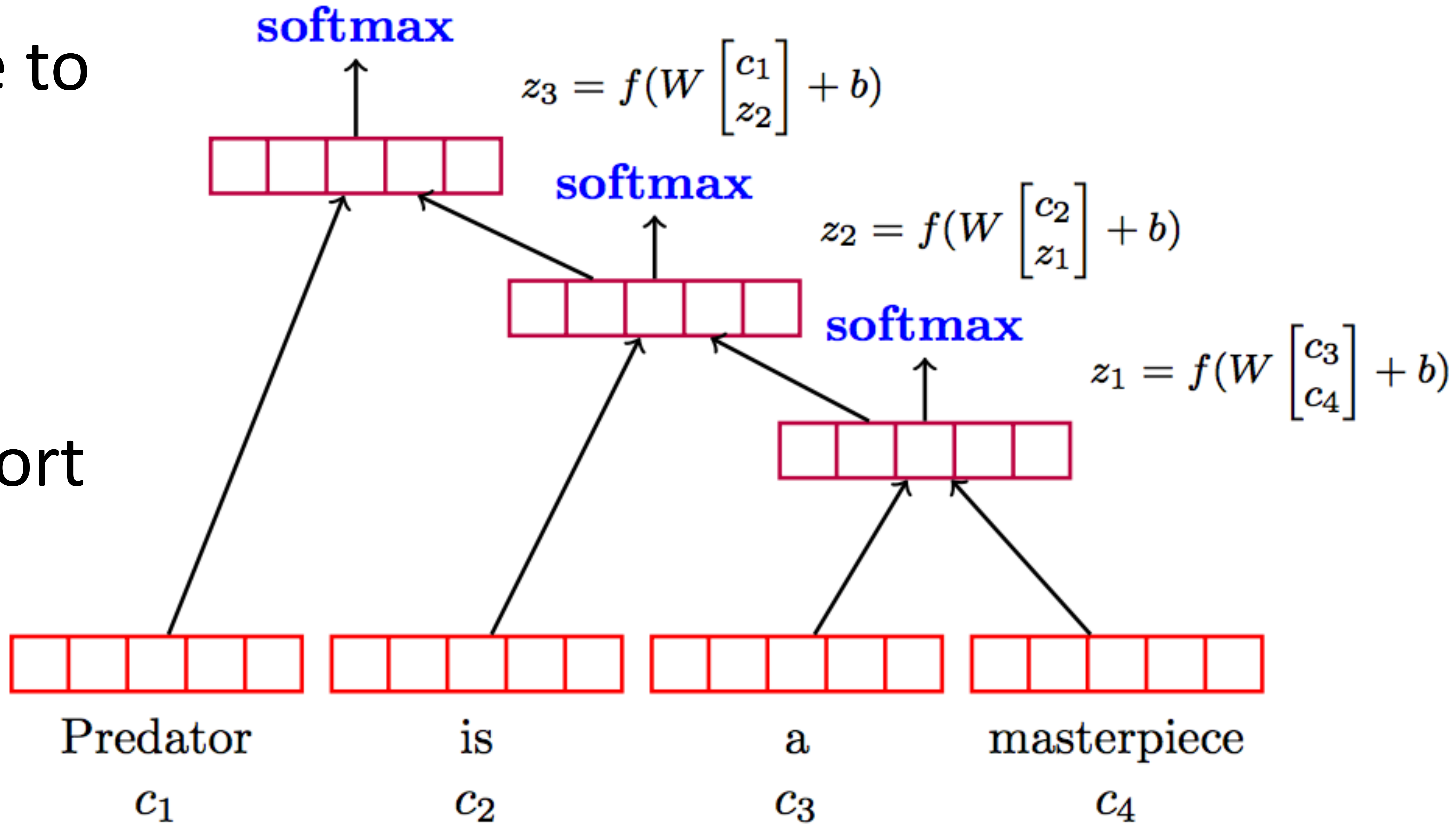
- Deep Averaging Networks: feedforward neural network on average of word embeddings from input



Iyyer et al. (2015)

Deep Averaging Networks

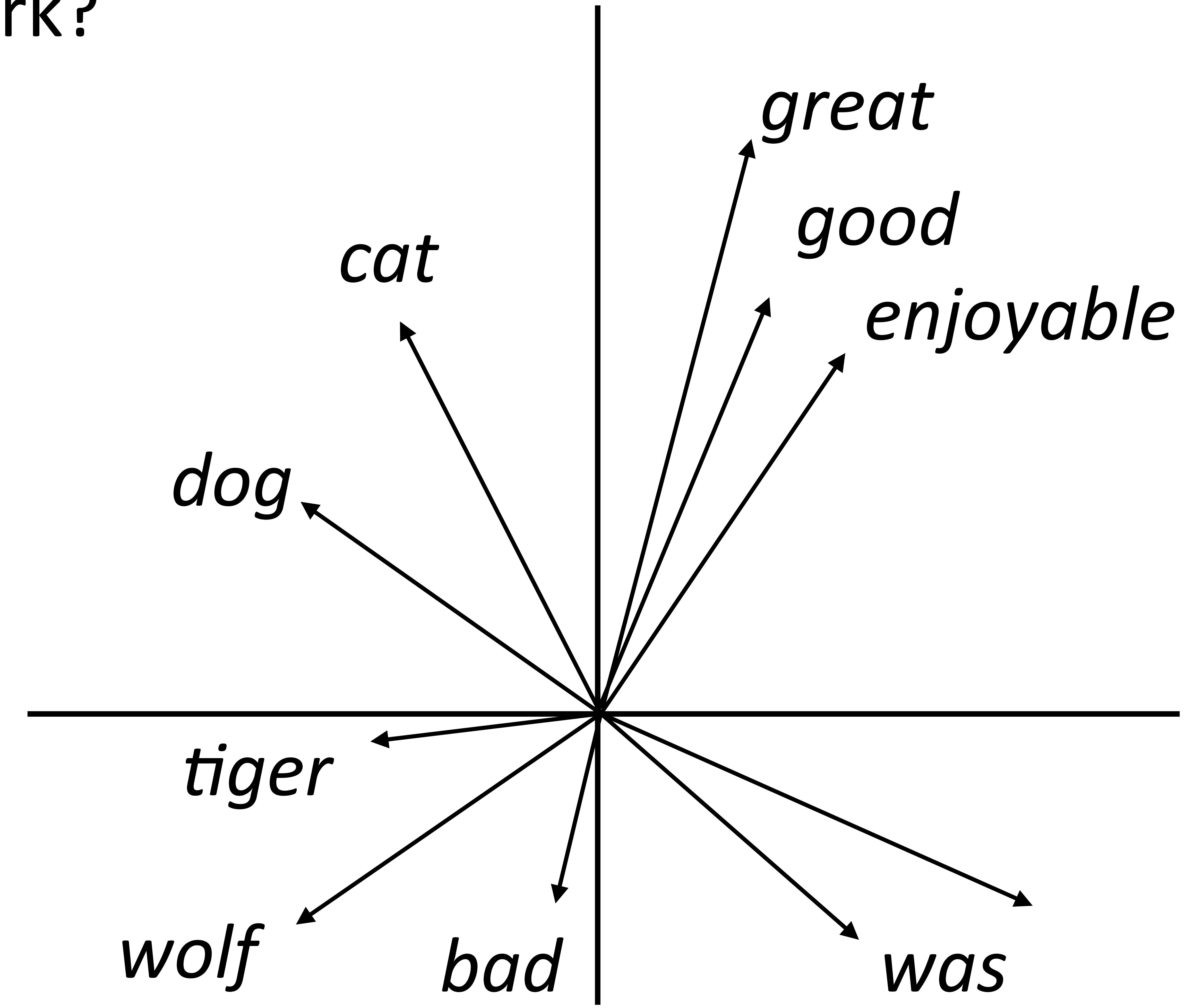
- Widely-held view: need to model syntactic structure to represent language
- Surprising that averaging can work as well as this sort of composition



Iyyer et al. (2015)

Deep Averaging Networks

- Why should these work?



Iyyer et al. (2015)

Sentiment Analysis

No pretrained embeddings

Model	RT	SST fine	SST bin	IMDB	Time (s)
DAN-ROOT	—	46.9	85.7	—	**31**
DAN-RAND	77.3	45.4	83.2	88.8	136
DAN	80.3	47.7	86.3	89.4	136
NBOW-RAND	76.2	42.3	81.4	88.9	91
NBOW	79.0	43.6	83.6	89.0	91
BiNB	—	41.9	83.1	—	—
NBSVM-bi	79.4	—	—	91.2	—
RecNN*	77.7	43.2	82.4	—	—
RecNTN*	—	45.7	85.4	—	—
DRecNN	—	49.8	86.6	—	431
TreeLSTM	—	**50.6**	86.9	—	—
DCNN*	—	48.5	86.9	89.4	—
PVEC*	—	48.7	87.8	**92.6**	—
CNN-MC	**81.1**	47.4	**88.1**	—	2,452
WRRBM*	—	—	—	89.2	—

Iyyer et al. (2015)

Bag-of-words

Wang and Manning (2012)

Tree RNNs / CNNS / LSTMS

Kim (2014)

Deep Averaging Networks

Sentence	DAN	DRecNN	Ground Truth
who knows what exactly godard is on about in this film, but his words and images do n't have to add up to mesmerize you.	positive	positive	positive
it's so good that its relentless, polished wit can withstand not only inept school productions, but even oliver parker's movie adaptation	negative	positive	positive
too bad, but thanks to some lovely comedic moments and several fine performances, it's not a total loss	negative	negative	positive
this movie was not good	negative	negative	negative
this movie was good	positive	positive	positive
this movie was bad	negative	negative	negative
the movie was not bad	negative	negative	positive

- Will return to compositionality with syntax and LSTMs

Iyyer et al. (2015)

Other Applications

Part-of-Speech Tagging

- Text classification: label applies to whole sentence

  The movie was great   Label = Positive

- Tagging: label each word individually

  *Fed raises **interest** rates in order to …*

  Label = Noun

- Next class: part-of-speech tagging

- Morphological analysis, named entity recognition, …

NLP with Feedforward Networks

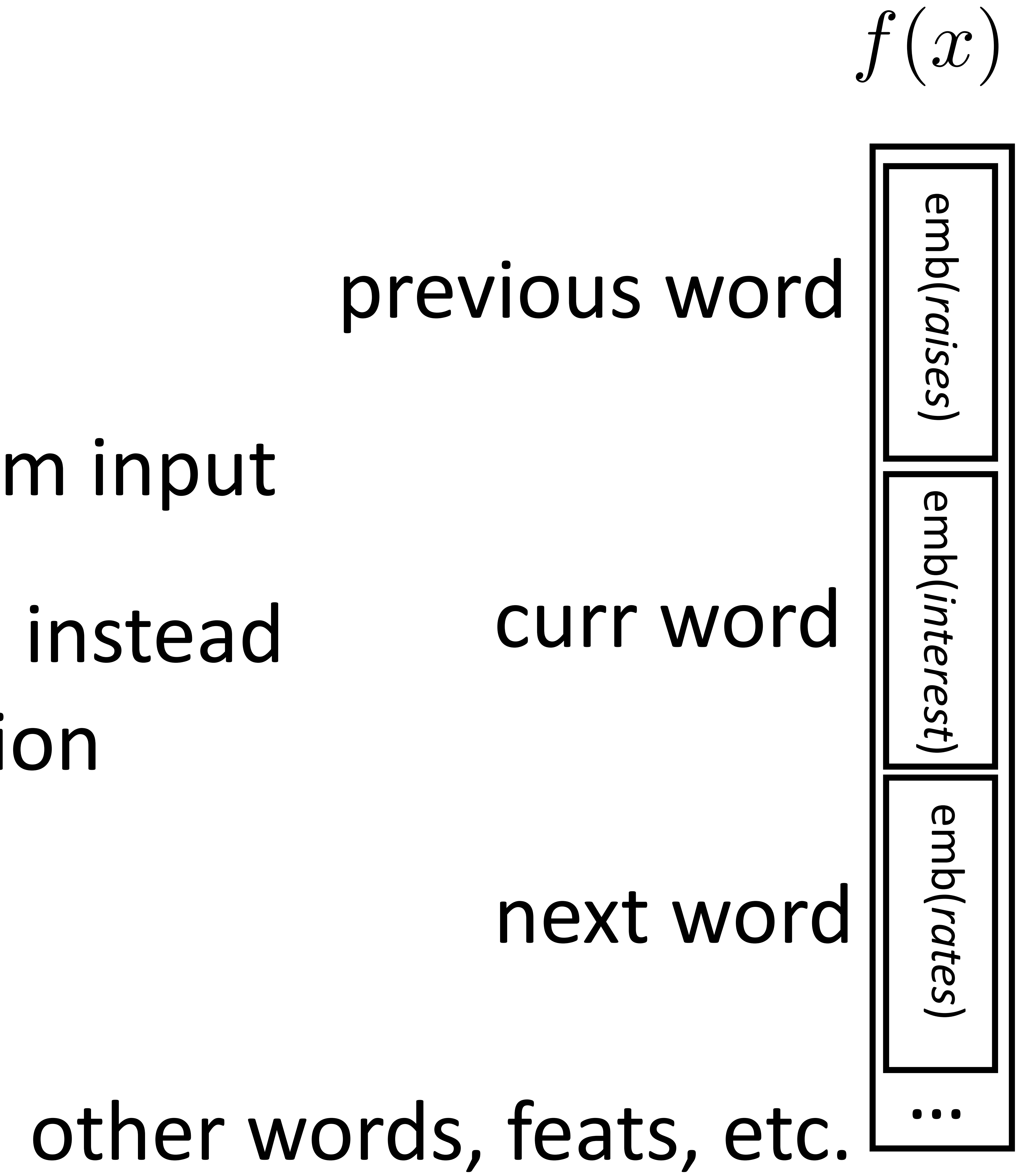
- Part-of-speech tagging with FFNNs

??

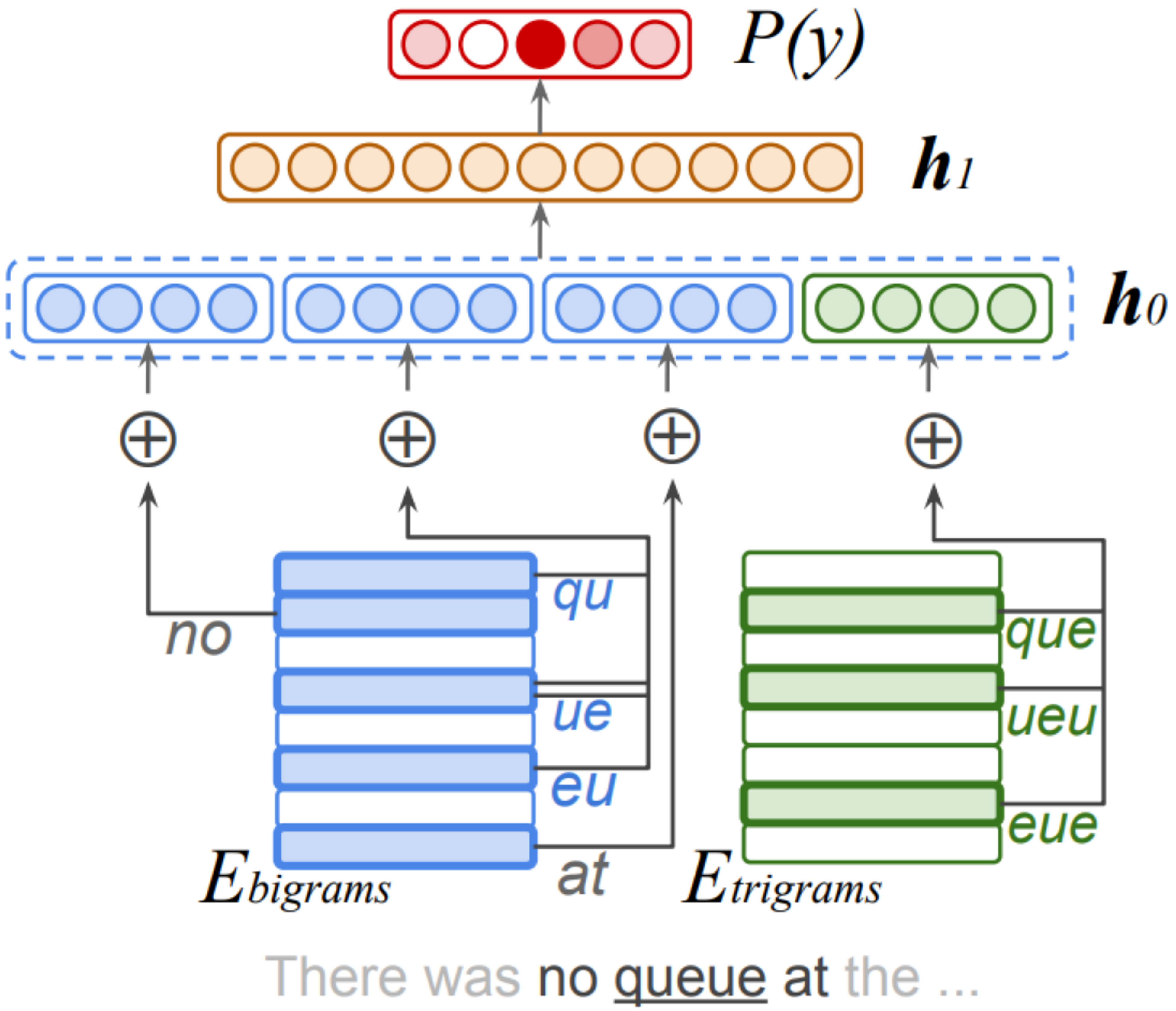
Fed *raises* ***interest*** *rates* *in order to …*

- Word embeddings for each word form input

- *f*(*x*) doesn't look like a bag-of-words, instead captures position-sensitive information



NLP with Feedforward Networks



- Botha et al. (2017): small FFNNs for NLP tasks
- Use character bigram + trigram embeddings
- Hidden layer mixes these different signals and learns feature conjunctions
- Works well on a range of languages

Botha et al. (2017)

Takeaways

- Continuous bag-of-words, Skip-gram, and Skip-gram with negative sampling are all similar ways to learn embeddings
- Matrix factorization approaches like GloVe are most standard
- Averaging inputs to feedforward networks can work well, will see other approaches later
- Later in the class: approaches to create “contextualized” word embeddings