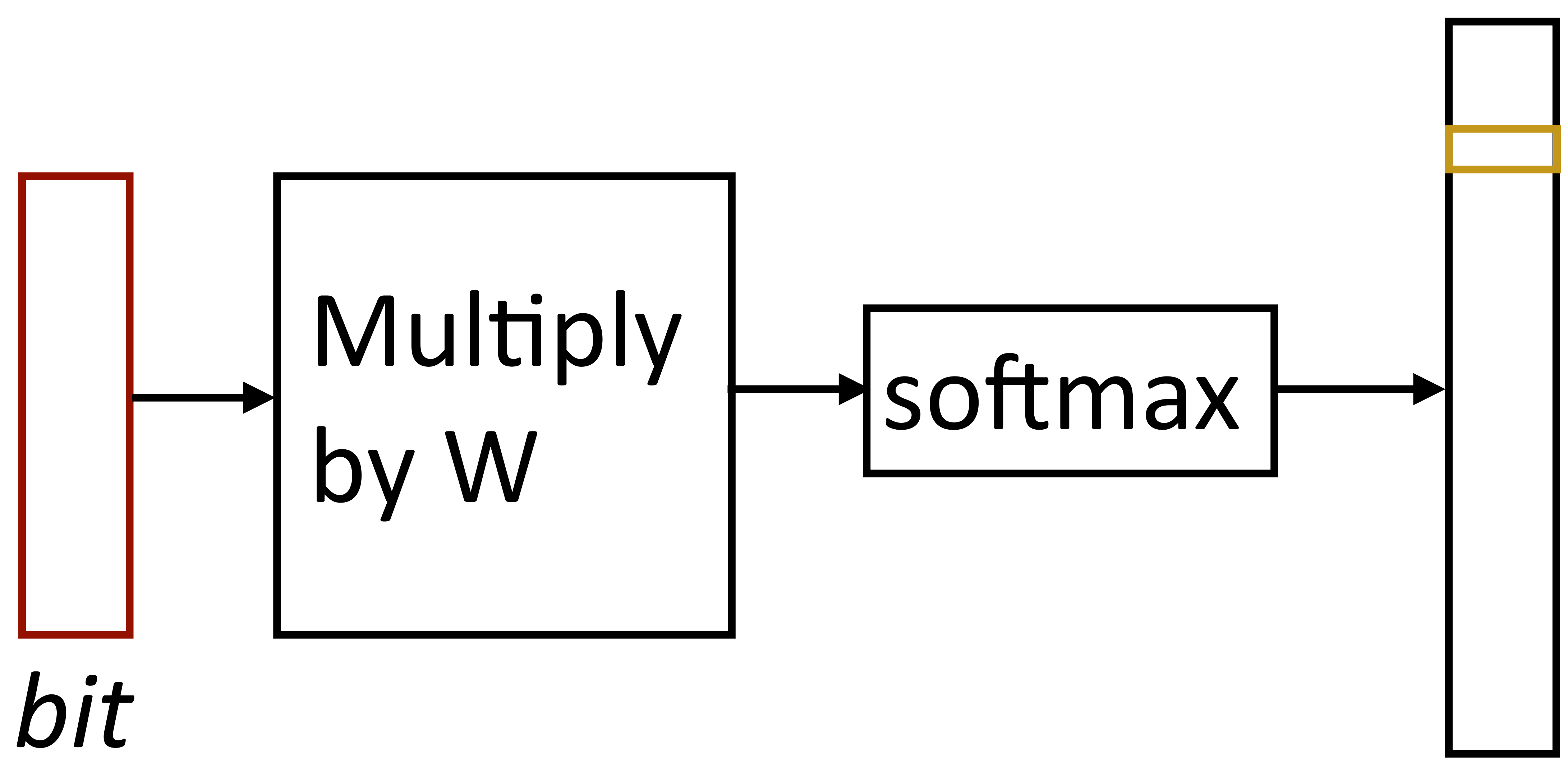
Skip-Gram

- Predict each word of context from word in turn, up to distance *k* — *the dog bit the man*



gold = *dog*

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

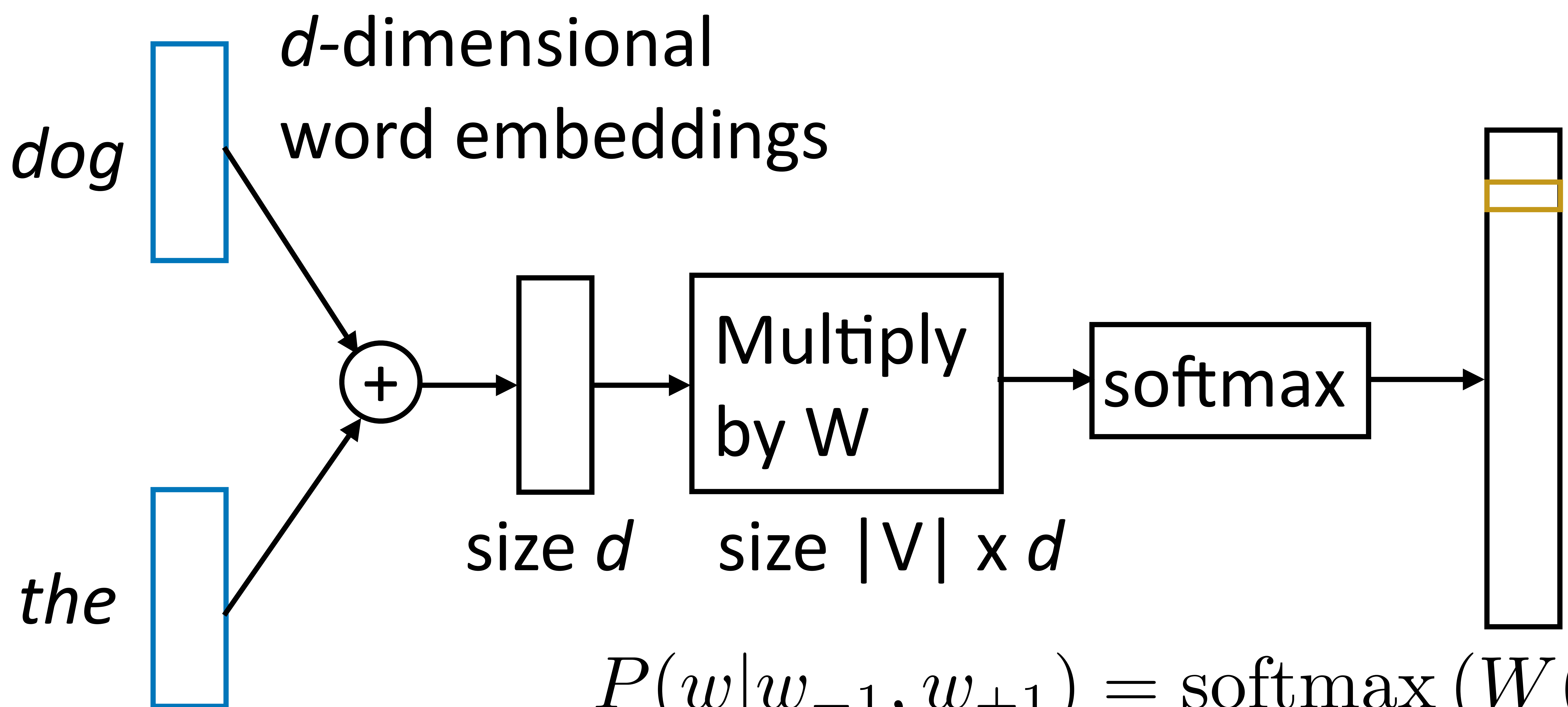
- Another training example: *bit -> the*
- Parameters: *d* x |V| word vectors, |V| x *d* context vectors (stacked into a matrix W)
- Why skip-gram? With window size >1, we predict a context word skipping over intermediate words

Mikolov et al. (2013)

Continuous Bag-of-words

- Predict *word* from multiple words of *context*

the dog ***bit*** *the man*



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

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

- Parameters: *d* x |V| (one *d*-length context vector per voc word),
  |V| x *d* word vectors (in matrix W)

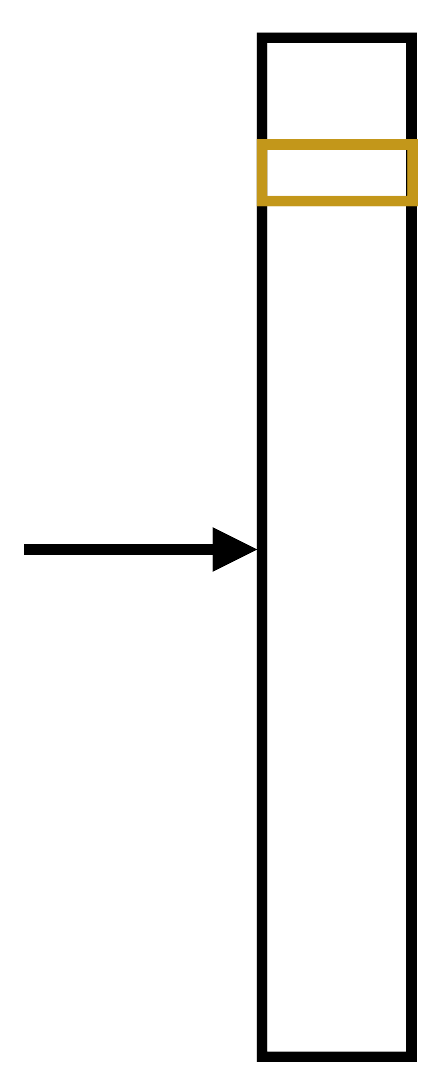
Mikolov et al. (2013)

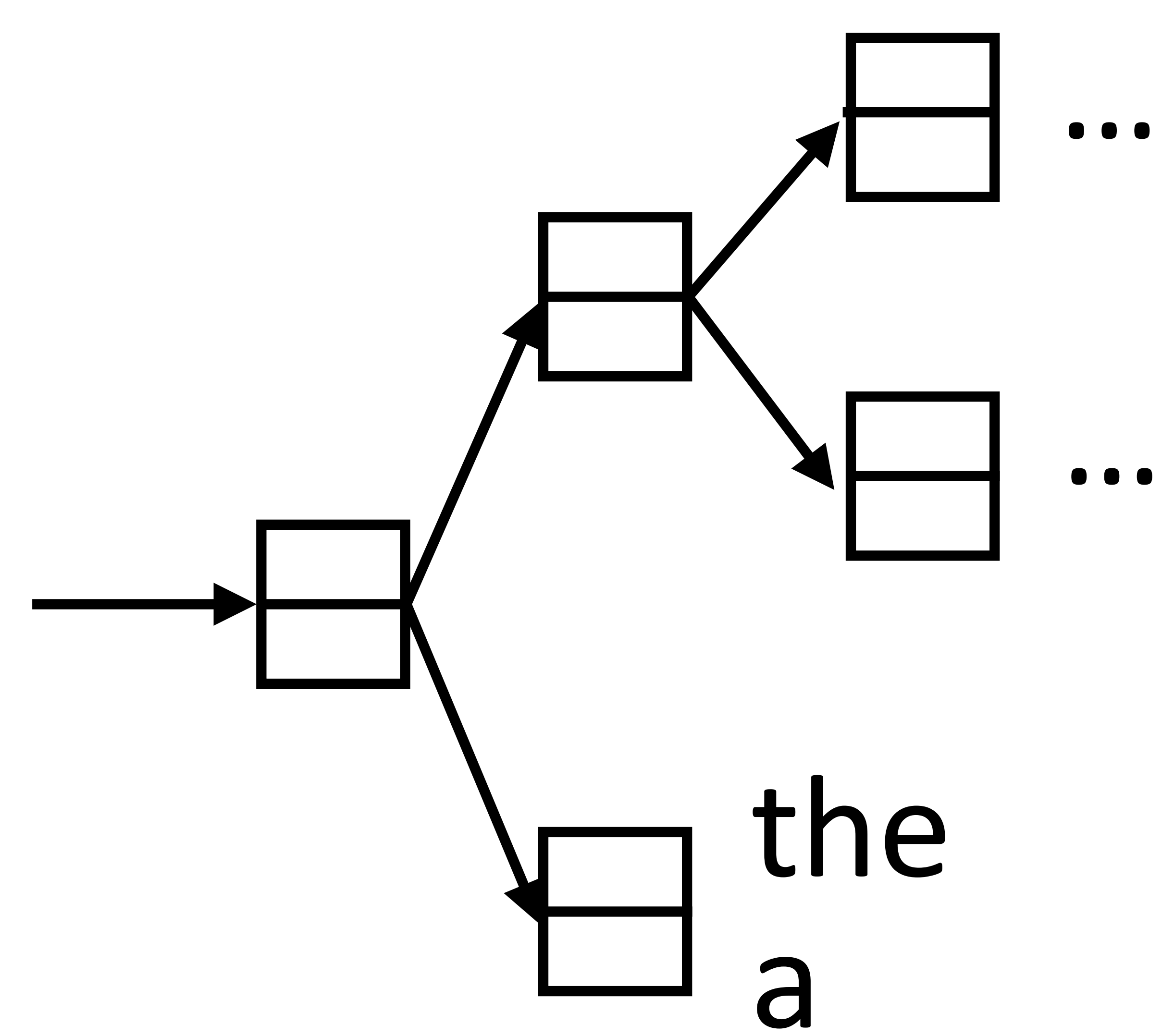
Hierarchical Softmax

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

Skip-gram: $P(w'|w) = \text{softmax}(We(w))$

- Matmul + softmax over |V| is very slow to compute for both techniques





- Huffman encode vocabulary, use binary classifiers to decide which branch to take
- log(|V|) binary decisions

- Standard softmax: |V| dot products of size *d*

- Hierarchical softmax:

log(|V|) dot products of size *d,*

|V| x *d* parameters

Mnih and Hinton (2008)

Skip-gram with Negative Sampling

- Are there alternative ways to learn vectors while avoiding O(|V|) term?
- 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

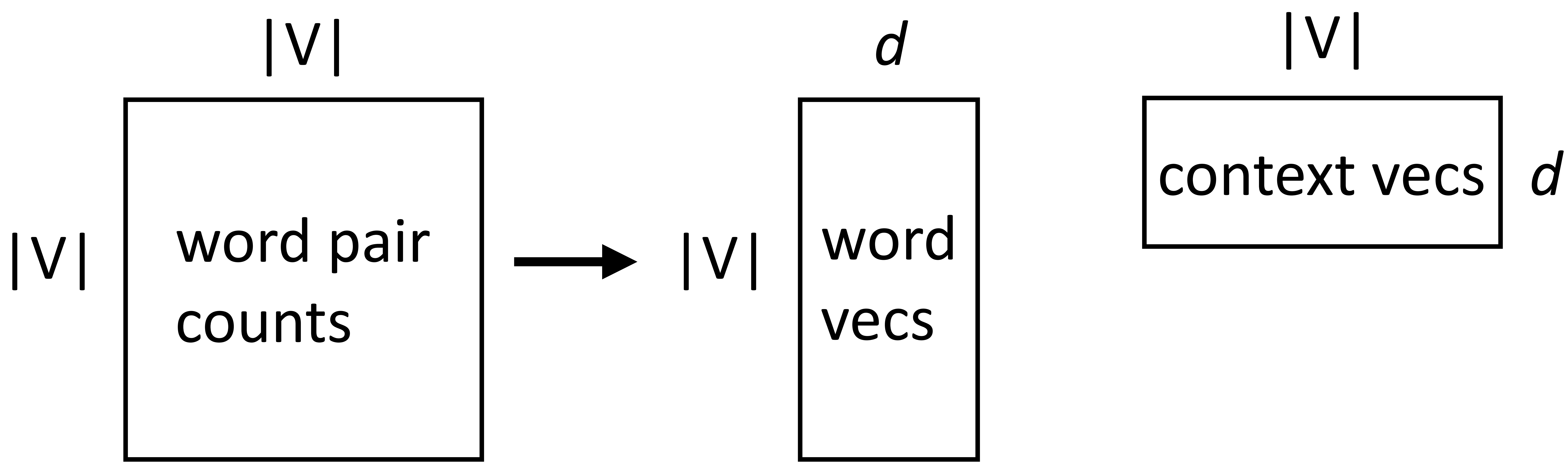
- *d* x |V| vectors, *d* x |V| context vectors (same # of params as before)
- Objective = $\log P(y=1|w,c) + \frac{1}{k}\sum_{i=1}^{n} \log P(y=0|w_i,c)$

sampled

Mikolov et al. (2013)

Connections with Matrix Factorization

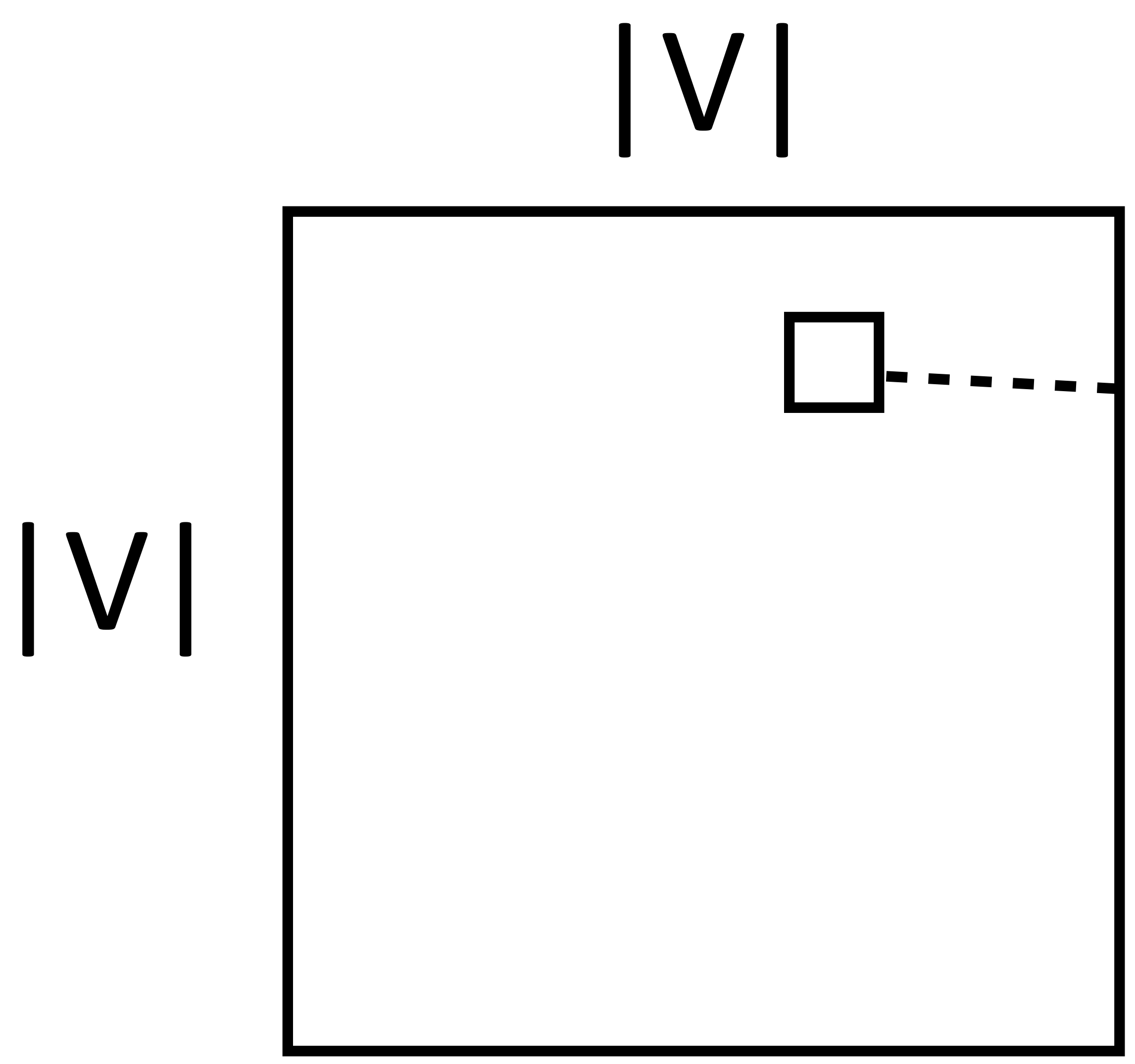
- Skip-gram model looks at word-word co-occurrences and produces two types of vectors



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

Levy et al. (2014)

Skip-gram as Matrix Factorization



num negative samples

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

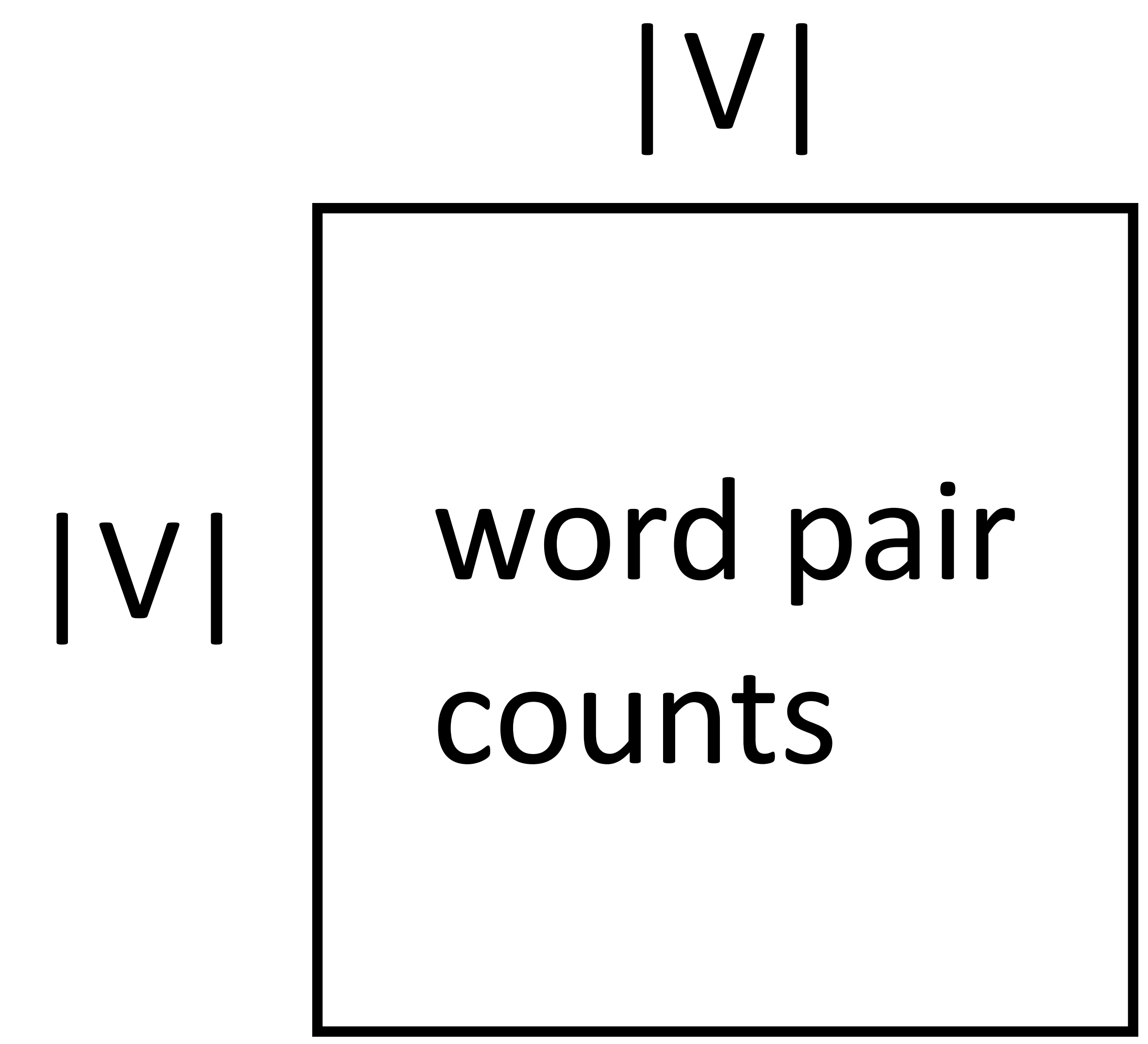
$$\mathrm{PMI}(w_i, c_j) = \frac{P(w_i, c_j)}{P(w_i)P(c_j)} = \frac{\frac{\mathrm{count}(w_i,c_j)}{D}}{\frac{\mathrm{count}(w_i)}{D}\,\frac{count(c_j)}{D}}$$

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

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

Levy et al. (2014)

GloVe (Global Vectors)



- Also operates on counts matrix, weighted regression on the log co-occurrence matrix
- Objective = $\sum_{i,j} f(\mathrm{count}(w_i, c_j)) \left(w_i^\top c_j + a_i + b_j - \log \mathrm{count}(w_i, c_j)\right)^2$
- Constant in the dataset size (just need counts), quadratic in voc size
- By far the most common word vectors used today (5000+ 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 $\left(\sum_{g \in \text{ngrams}} w_g \cdot c \right)$

Bojanowski et al. (2017)

Pre-trained Models: ELMo, GPT, BERT

- These encode "subwords" rather than words. Underscore indicates that the following token continues the existing word

  and there were no re_ fueling stations anywhere

  one of the city 's more un_ princi_ pled real estate agents

- Any word is either in the subword vocabulary or can be expressed as a sequence of subwords in the vocabulary

- Embeddings are computed using RNNs and Transformers. We can't just look up an embedding for each word, but actually need to run a model

- Learn embeddings through language modeling (discussed in the second half of the course)