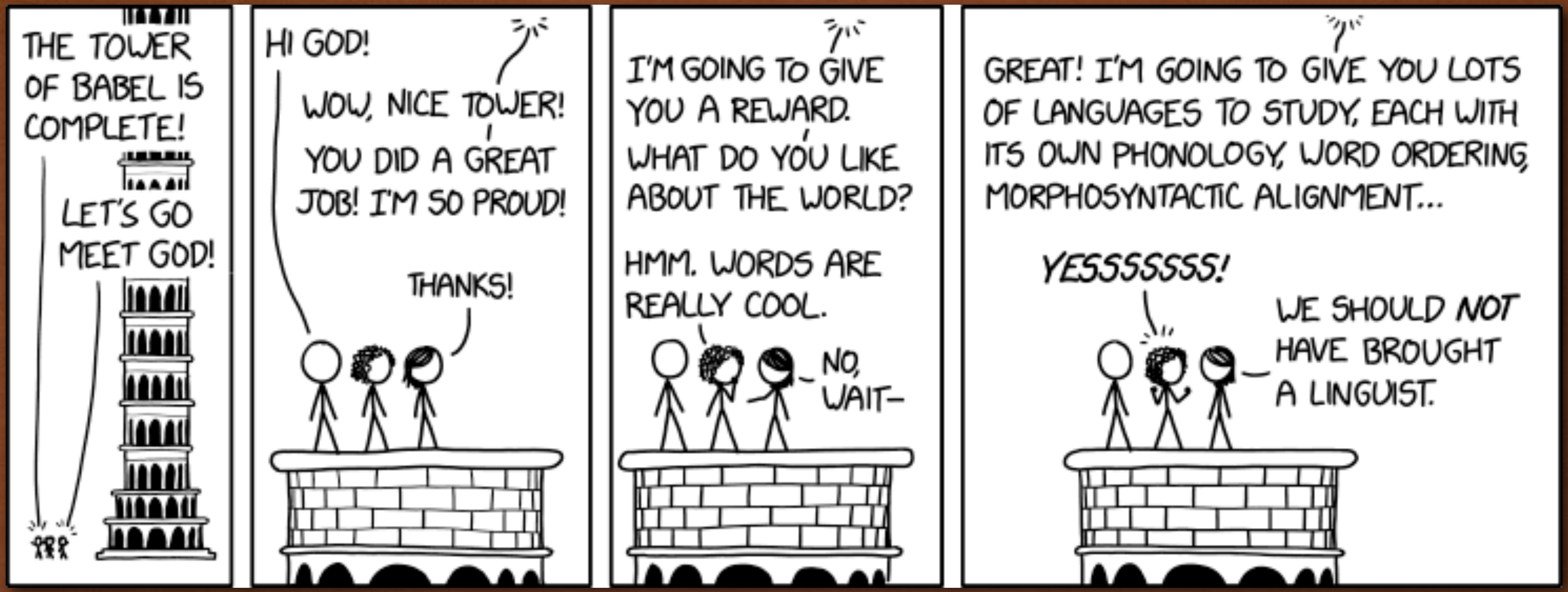


Soon, linguists will be wandering around everywhere, saying things like "colorless green ideas sleep furiously" and "more people have been to Russia than I have," and speech will become unintelligible.

COMMON ERRORS IN ASSIGNMENT 1

```
hamlet = ''.join(lines)
```

```
for x in sentences:
    x.lower()
    tok_again = nltk.word_tokenize(x)
    lower_case_words.append(tok_again)
    #This gets us total set count aka types
    lower_count = lower_count + len(set(tok_again))
```

```
tokens = nltk.word_tokenize(data.lower())
    get_types = nltk.pos_tag(tokens)
    types = []
    for t in get_types:
        if not types.__contains__(t[1]):
            types.append(t[1])
```

- READ AND FOLLOW THE INSTRUCTIONS
    - In the future, I will not grade anything that does not follow the instructions
- Your report should be self-sufficient
    - Provide answers to all questions
- Put your name in the PDF

ANTONIS ANASTASOPOULOS
CS499 INTRODUCTION TO NLP

NEURAL EMBEDDINGS

https://cs.gmu.edu/~antonis/course/cs499-spring21/

With adapted slides by David Mortensen and Alan Black

DENSE VECTORS (NEURAL EMBEDDINGS)

WHY DENSE VECTORS?

PPMI vectors are:

- long: length $|V| = [20{,}000 - 50{,}000]$
- sparse: most elements are 0

The alternative is to learn vectors which are:

- short: length 200-1000
- dense: most elements are non-zero

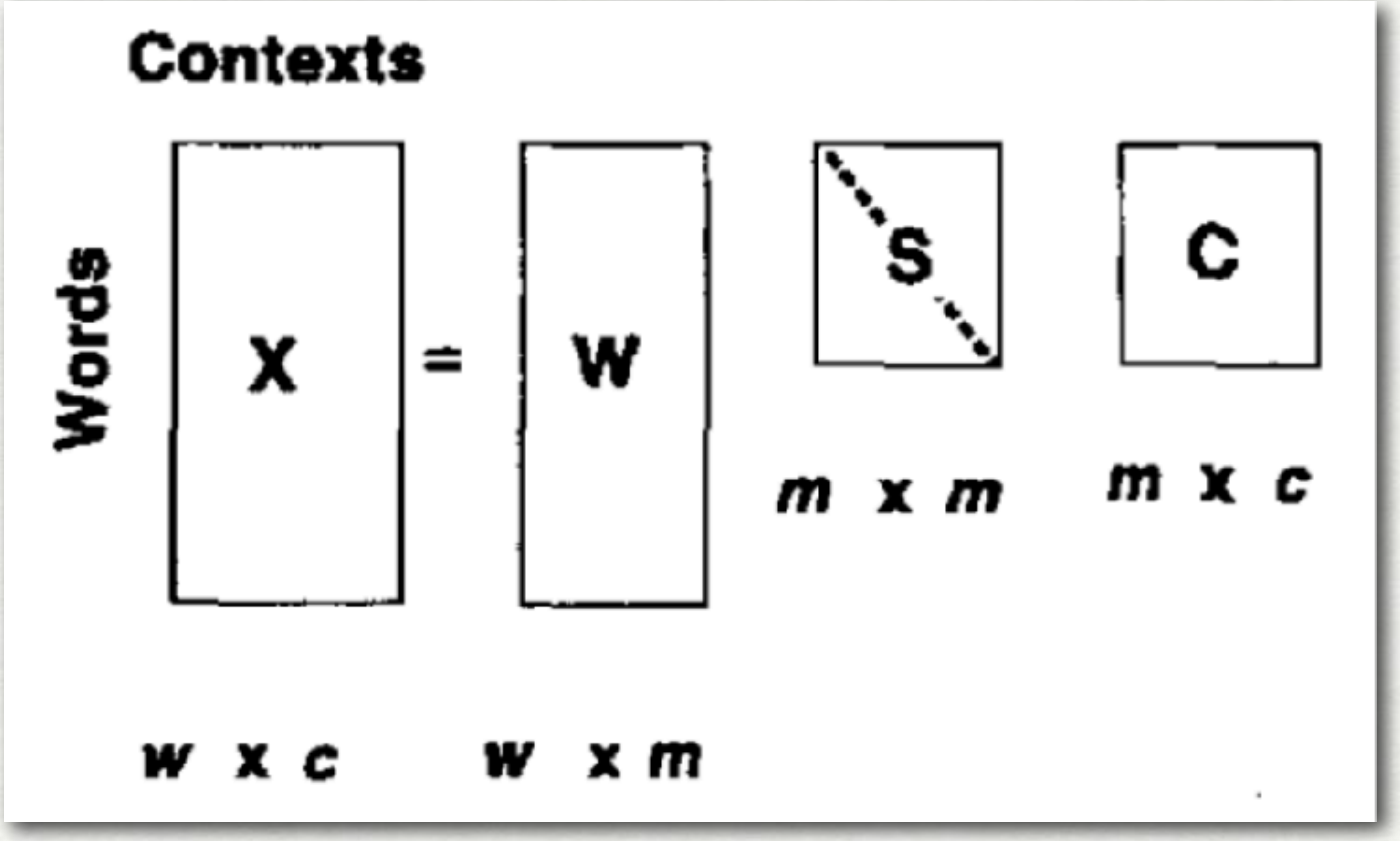
Advantages

1. Easier to use in ML
2. Might generalize better than storing explicit counts
3. May do better at capturing synonymy:
   - *car* and *automobile* are synonyms
   - but in PPMI they are separate dimensions
   - a word with *car* as neighbor, and a different word with *automobile* as neighbor are not captured as similar

OPTION 1: SINGULAR VALUE DECOMPOSITION

Intuition: approximate the big matrix using fewer dimensions

SVD: Any $w \times c$ matrix X can be written as a product of three matrices:

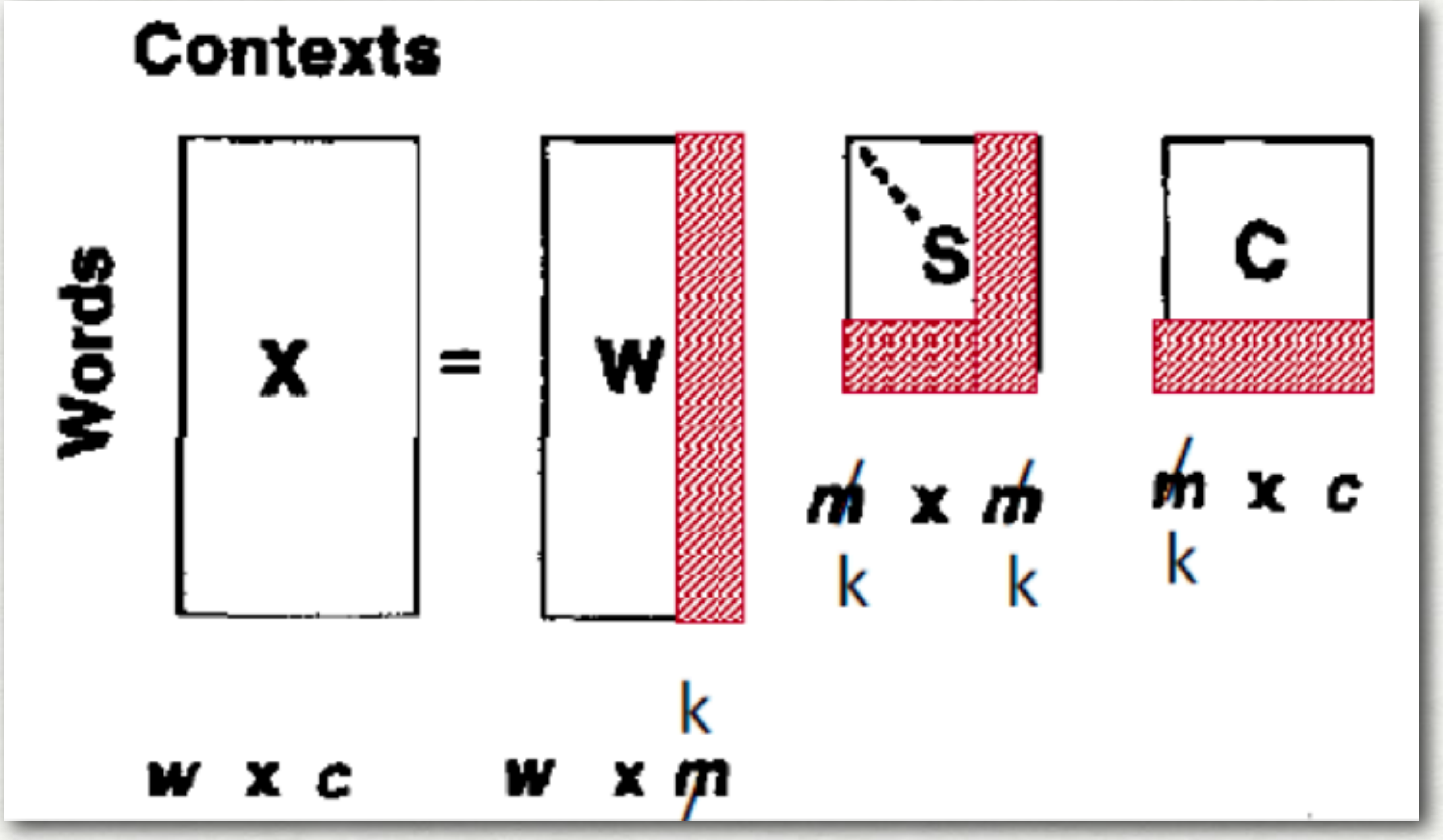


- $\mathbf{W}$: rows equal to original, but $m < k$ columns are dimensions in the new latent space
- $\mathbf{S}$: diagonal $m \times m$ matrix of **singular values** expressing the importance of each dimension
- $\mathbf{C}$: columns equal to original, but $m < k$ rows correspond to singular values

OPTION 1: SINGULAR VALUE DECOMPOSITION

Intuition: approximate the big matrix using fewer dimensions

SVD: Any $w \times c$ matrix X can be written as a product of three matrices:



Instead of keeping all m dimensions, we only keep the top k singular values.

Now, each row of truncated $\mathbf{W}$ is a k-dimensional vector representing a word w.

OPTION 2: LEARN (NEURAL) DENSE VECTORS FROM SCRATCH

Prediction-based models

Idea: learn embeddings as part of the process of word prediction

Implementation:

train a neural network to predict the context given a word

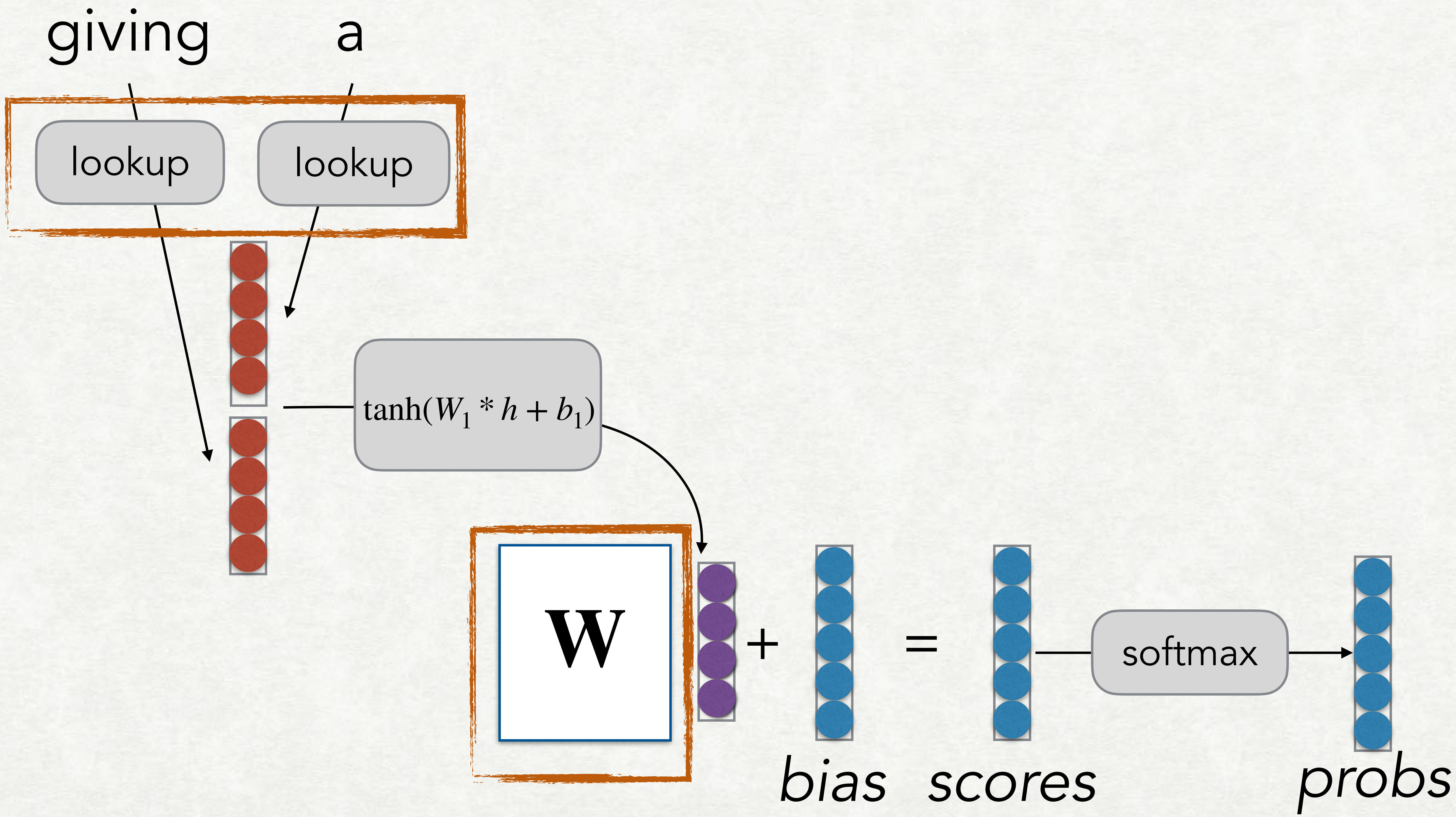
or the opposite: train a NN to predict a word given the context (LM?)

Advantages:

fast, easy to train

pre-trained embeddings for 100s of languages are available online!

WORD EMBEDDINGS FROM LANGUAGE MODELS



CONTEXT WINDOW METHODS

If we don't need to calculate the probability of the sentence, other methods possible!

These can move **closer to the contexts used in count-based methods**

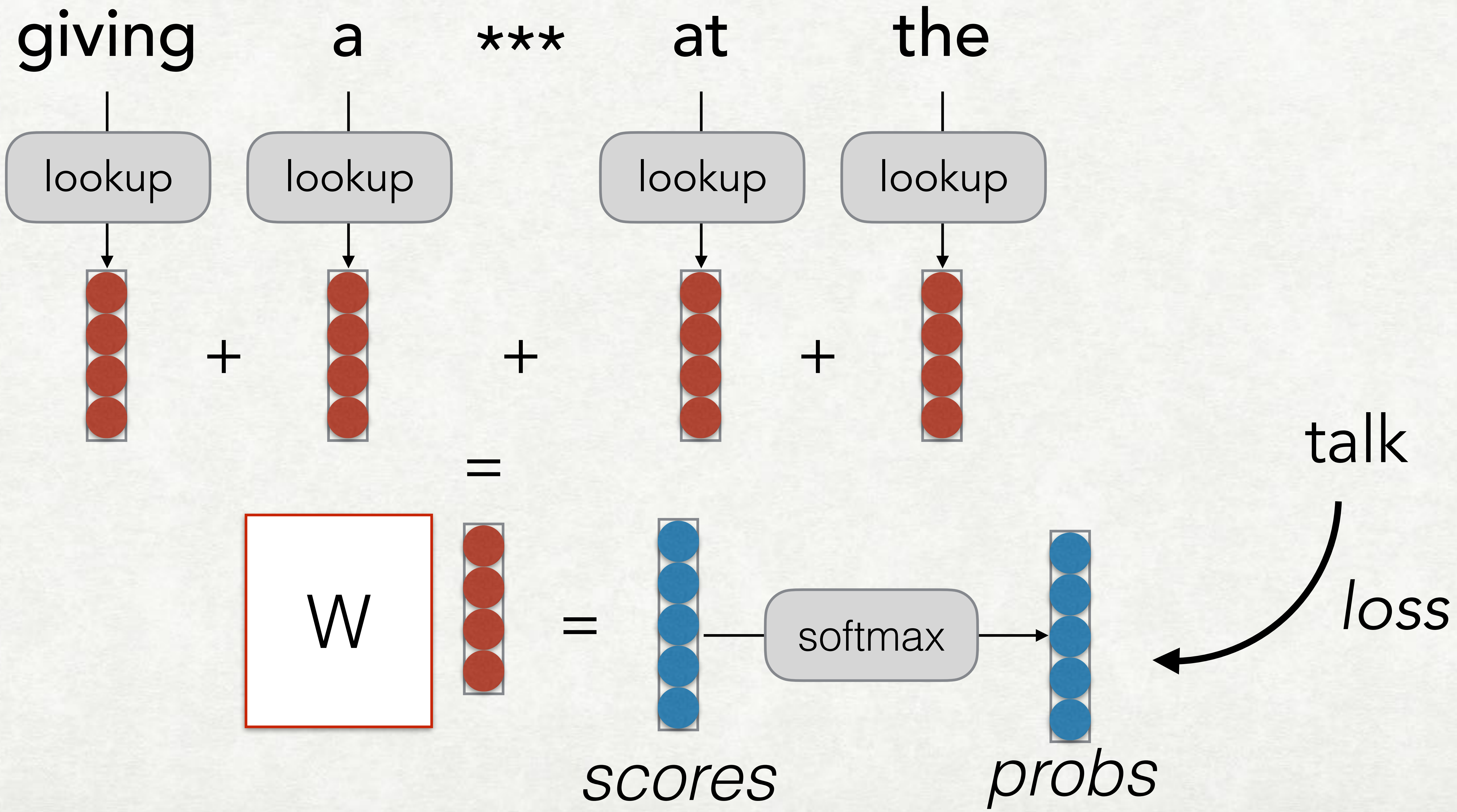
These drive word2vec, etc.

CONTINUOUS BAG-OF-WORDS (CBOW; MIKOLOV ET AL. 2013)

Predict word based on sum of surrounding embeddings

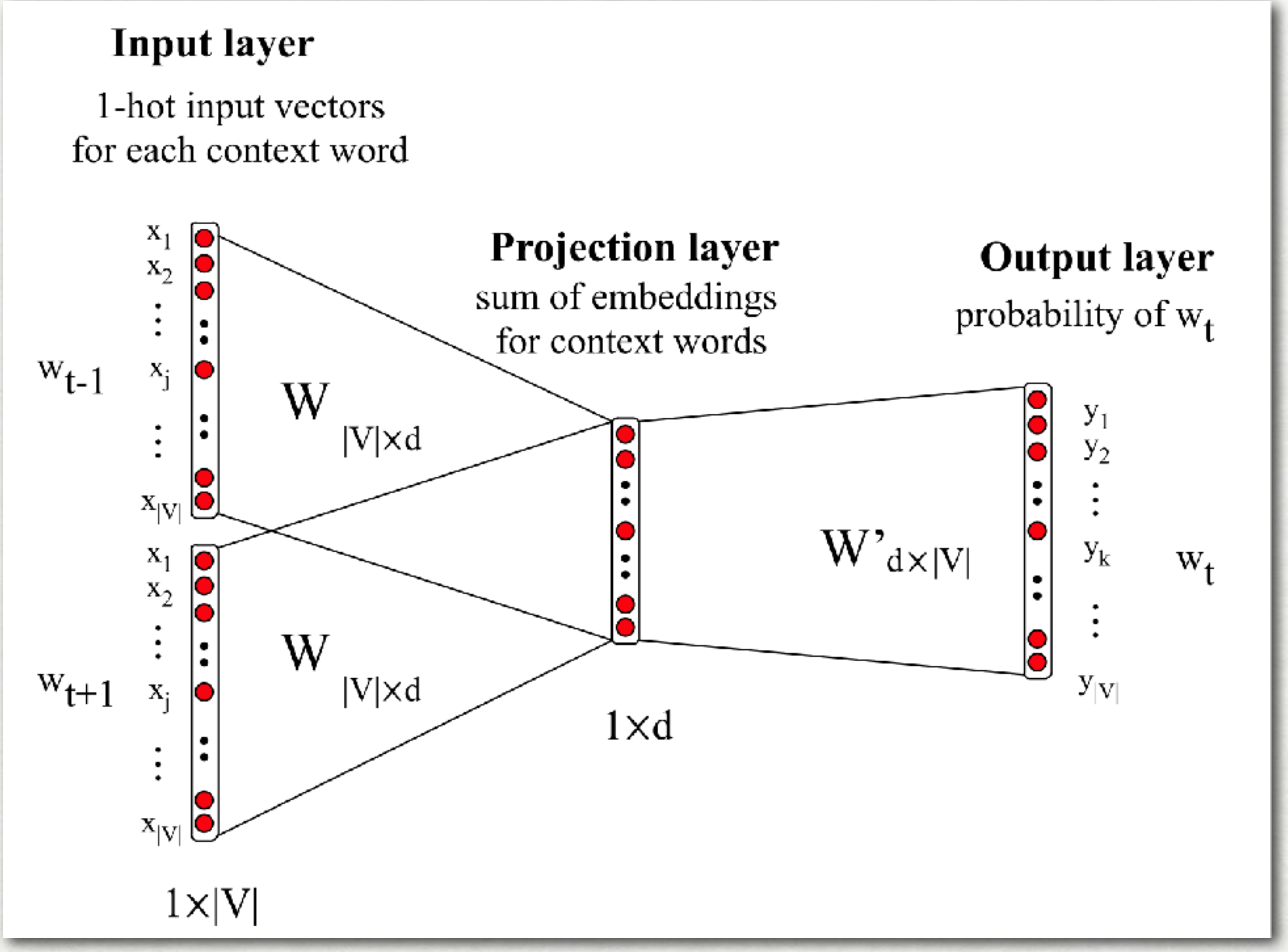
CONTINUOUS BAG-OF-WORDS (CBOW; MIKOLOV ET AL. 2013)

Predict word based on sum of surrounding embeddings



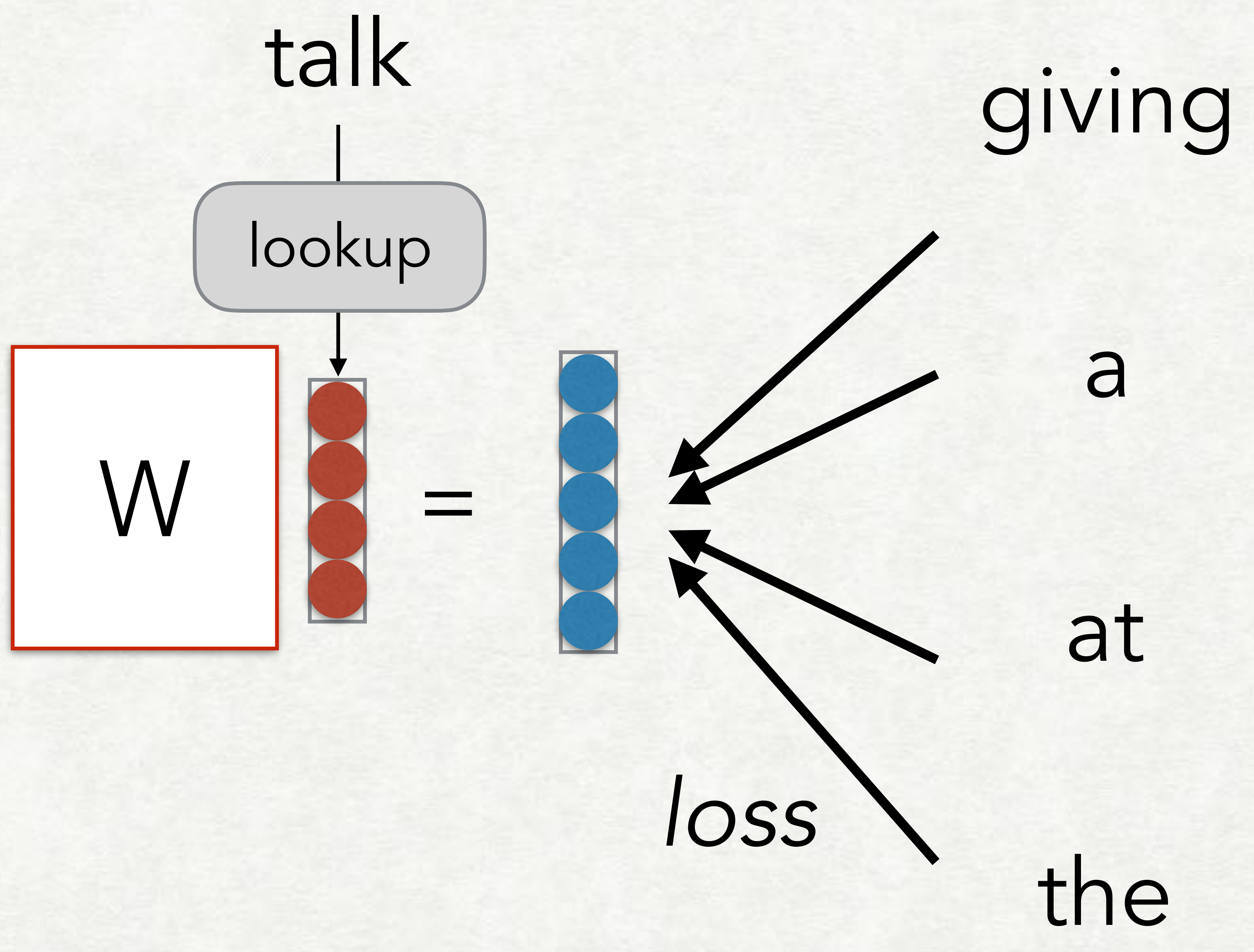
CONTINUOUS BAG-OF-WORDS (CBOW; MIKOLOV ET AL. 2013)

Predict word based on surrounding embeddings



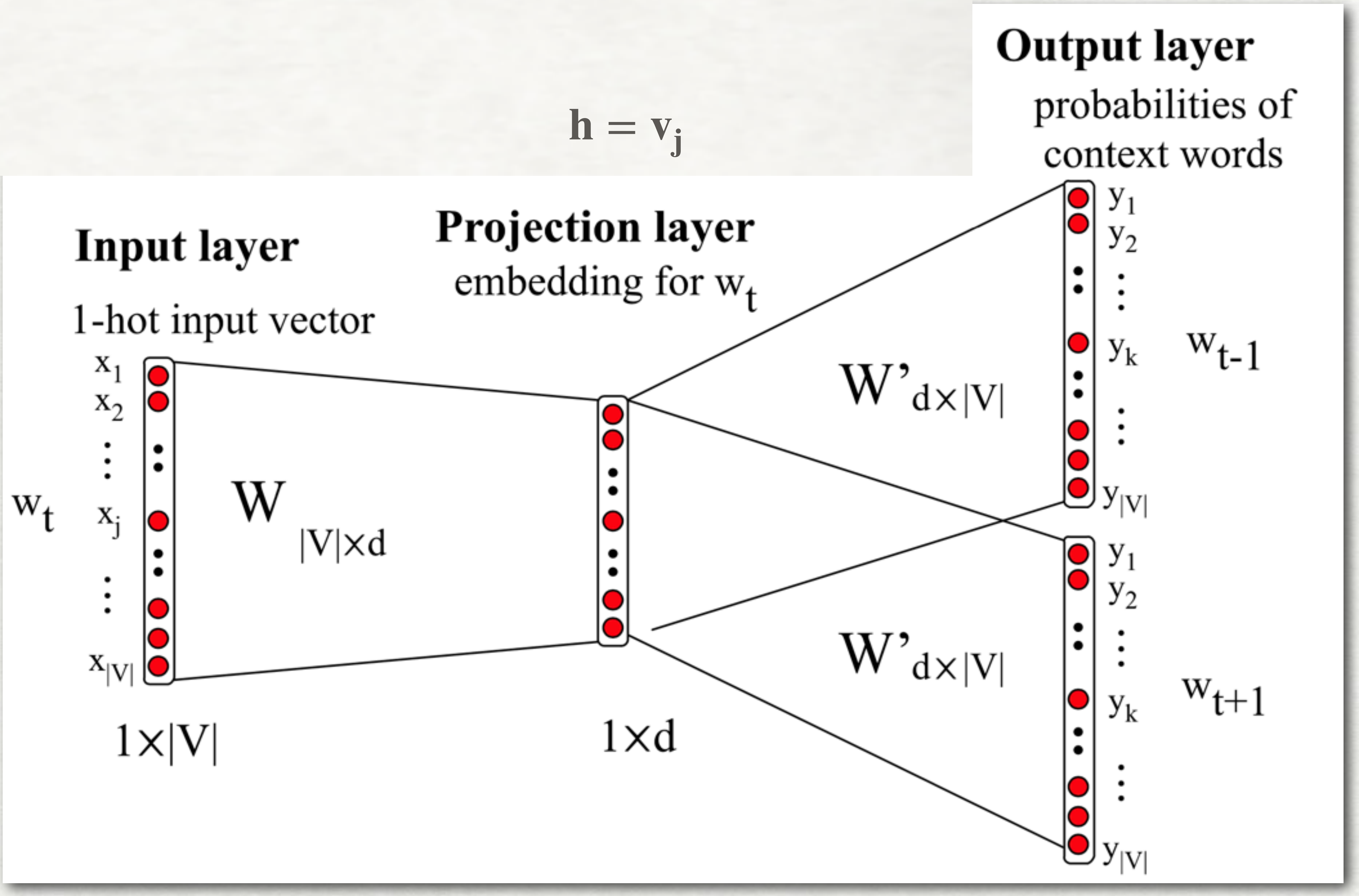
SKIP-GRAM (MIKOLOV ET AL. 2013)

Predict each word in the context given the word



SKIP-GRAM (MIKOLOV ET AL. 2013)

Predict each word in the context given the word



$\mathbf{o} = \mathbf{W'h}$

Use softmax to turn into probabilities:

$p(w_{t-1} \mid w_t) = \text{softmax}(\mathbf{o})$

COUNT-BASED AND PREDICTION-BASED METHODS

Strong connection between count-based methods and prediction-based methods (Levy and Goldberg 2014)

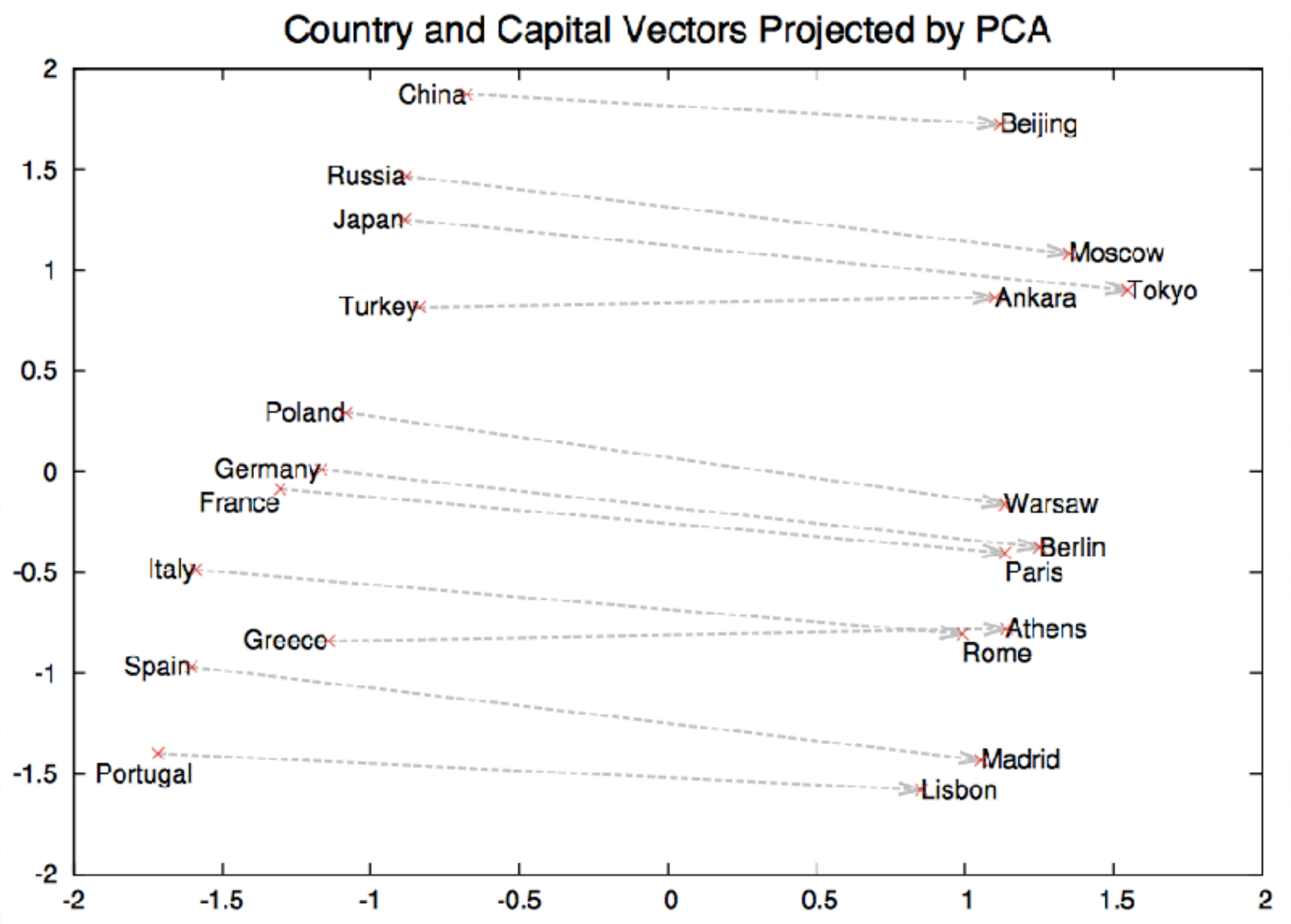
Skip-gram objective is equivalent to matrix factorization with PMI and discount for number of samples *k* (sampling covered next time)

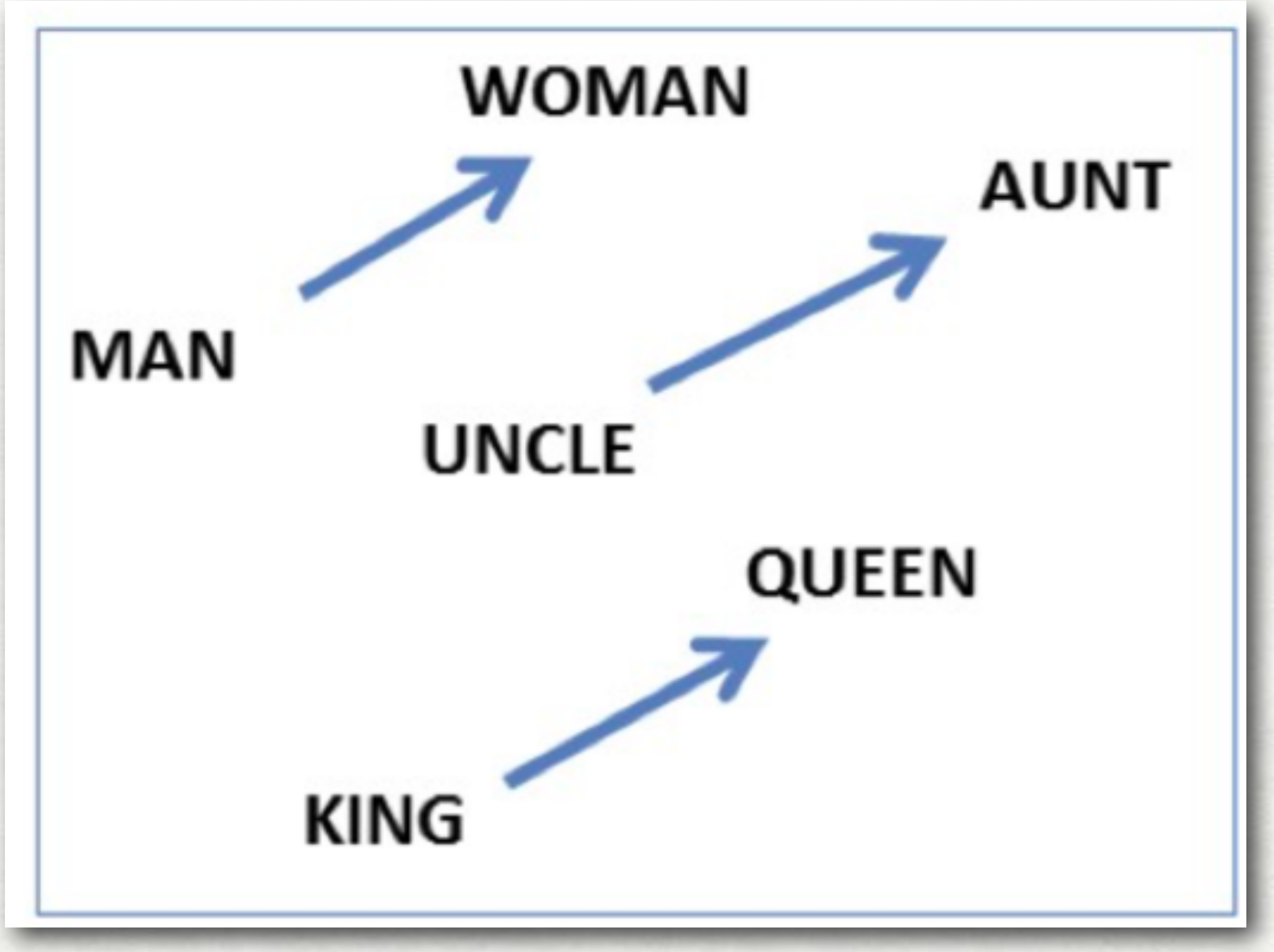
$$M_{w,c} = \mathrm{PMI}(w, c) - \log(k)$$

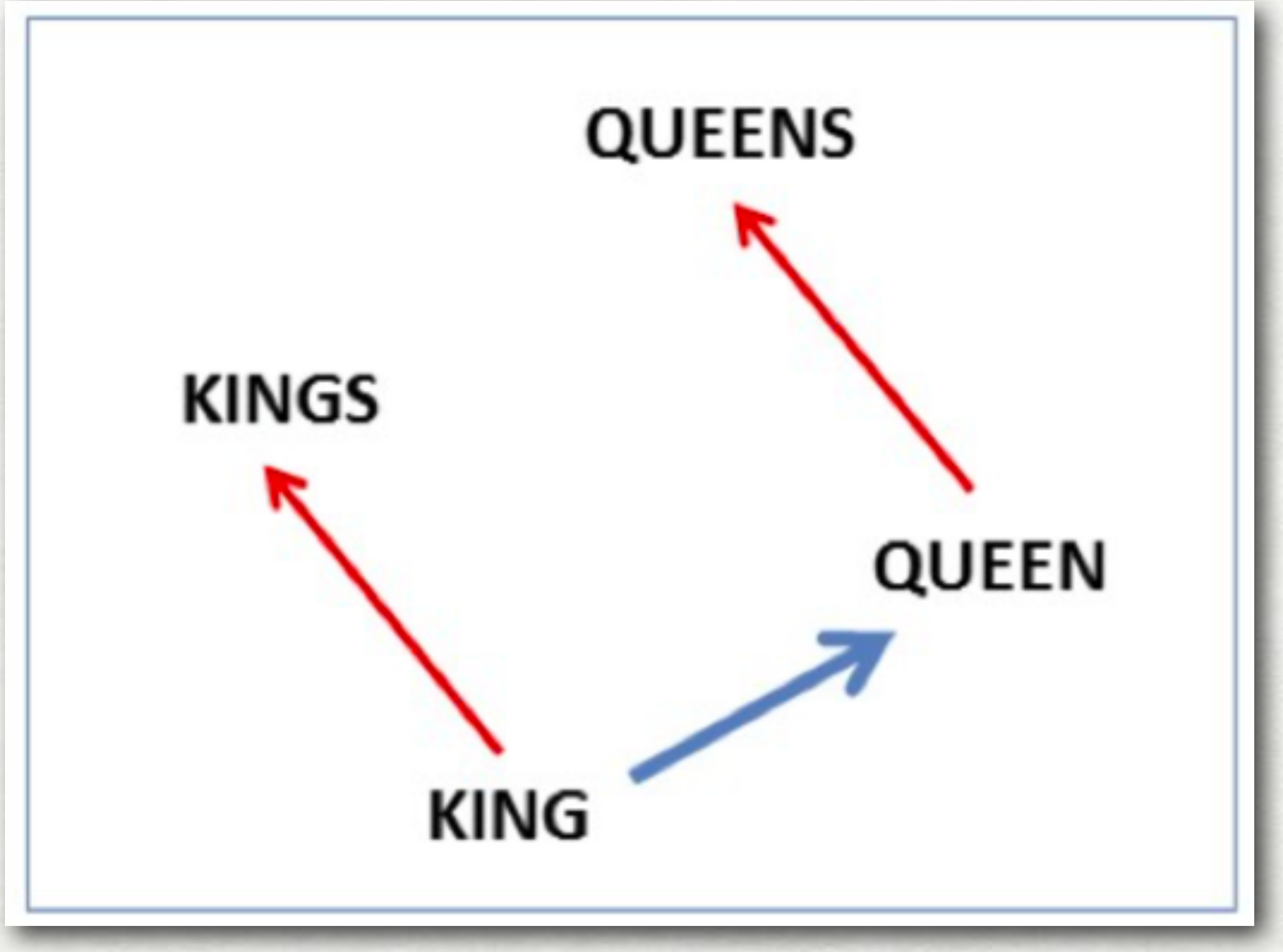
EMBEDDINGS CAPTURE RELATIONAL MEANING

v(Paris) – v(France + v(Italy) ≈ v(Rome)

v(king) – v(man + v(woman) ≈ v(queen)







CAN WE TRAIN EMBEDDINGS ON ALL OF WIKIPEDIA

Yes! In fact, good embeddings need lots of (appropriate) data

There already exist pertained models

- word2vec
- Glove
- FastText

In future classes we'll talk about
contextualized embeddings **(e.g. BERT, ELMo)**

https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html

https://fasttext.cc/docs/en/cheatsheet.html

TYPES OF EVALUATION

Intrinsic vs. Extrinsic

Intrinsic: How good is it based on its features?

Extrinsic: How useful is it downstream?

Qualitative vs. Quantitative

Qualitative: Examine the characteristics of examples.

Quantitative: Calculate statistics

INTRINSIC EVALUATION OF EMBEDDINGS (CATEGORIZATION FROM SCHNABEL ET AL 2015)

Relatedness: The correlation btw. embedding cosine similarity and human eval of similarity?

Analogy: Find x for "*a is to b, as x is to y*".

Categorization: Create clusters based on the embeddings, and measure purity of clusters.

Selectional Preference: Determine whether a noun is a typical argument of a verb.

EXTRINSIC EVALUATION: USING WORD EMBEDDINGS IN SYSTEMS

Initialize w/ the embeddings

Concatenate pre-trained embeddings with learned embeddings

Latter is more expressive, but leads to increase in model parameters