Machine learning: advanced

Language modeling and lexical representation

Tommi Jaakkola

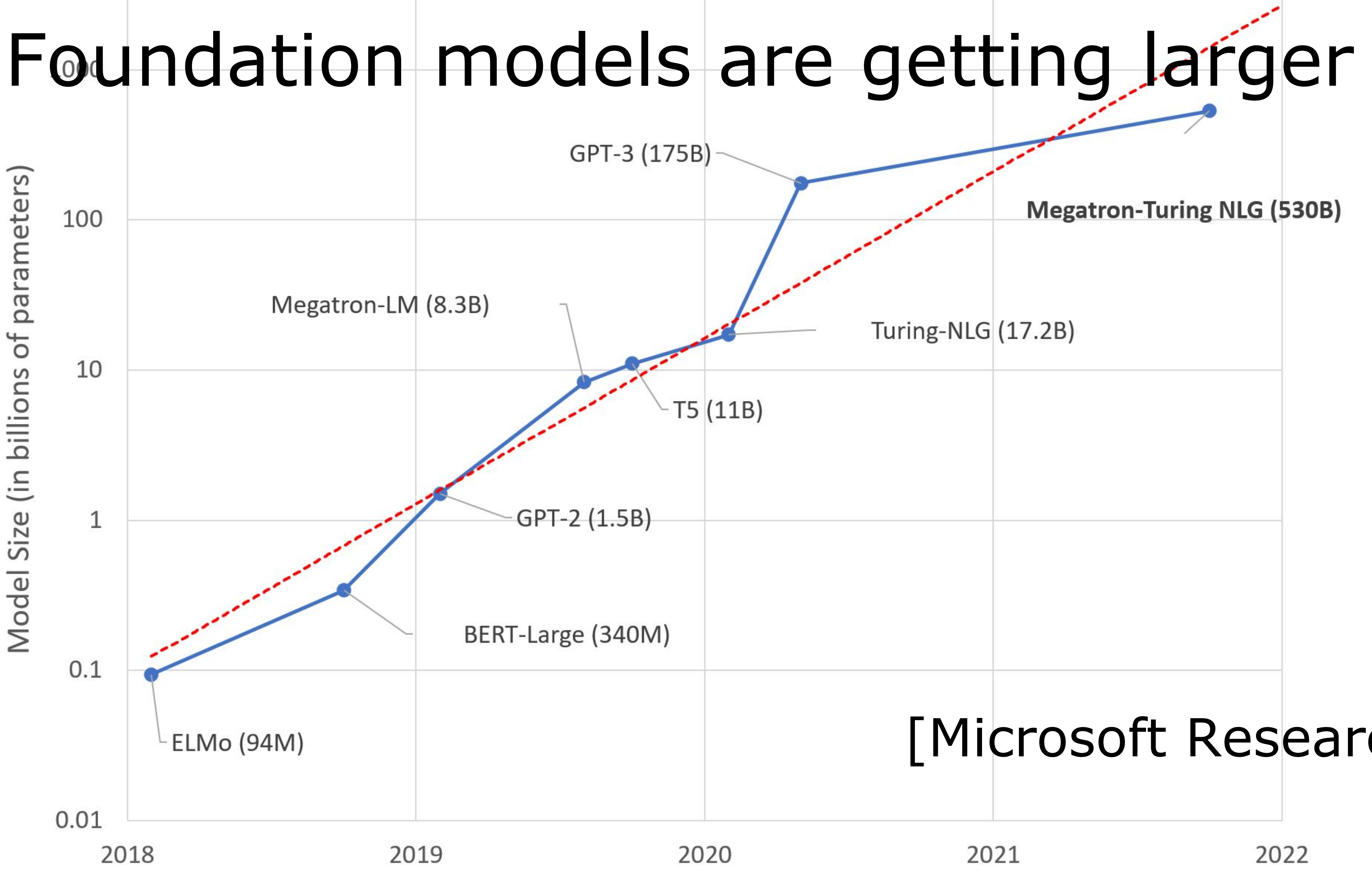
June 7, 2023

Outline

- Continuous embedding a la matrix factorization
- Word vectors from co-occurence data
- Masked language models, fine-tuning, contextual embeddings
- (Transformer language generation)

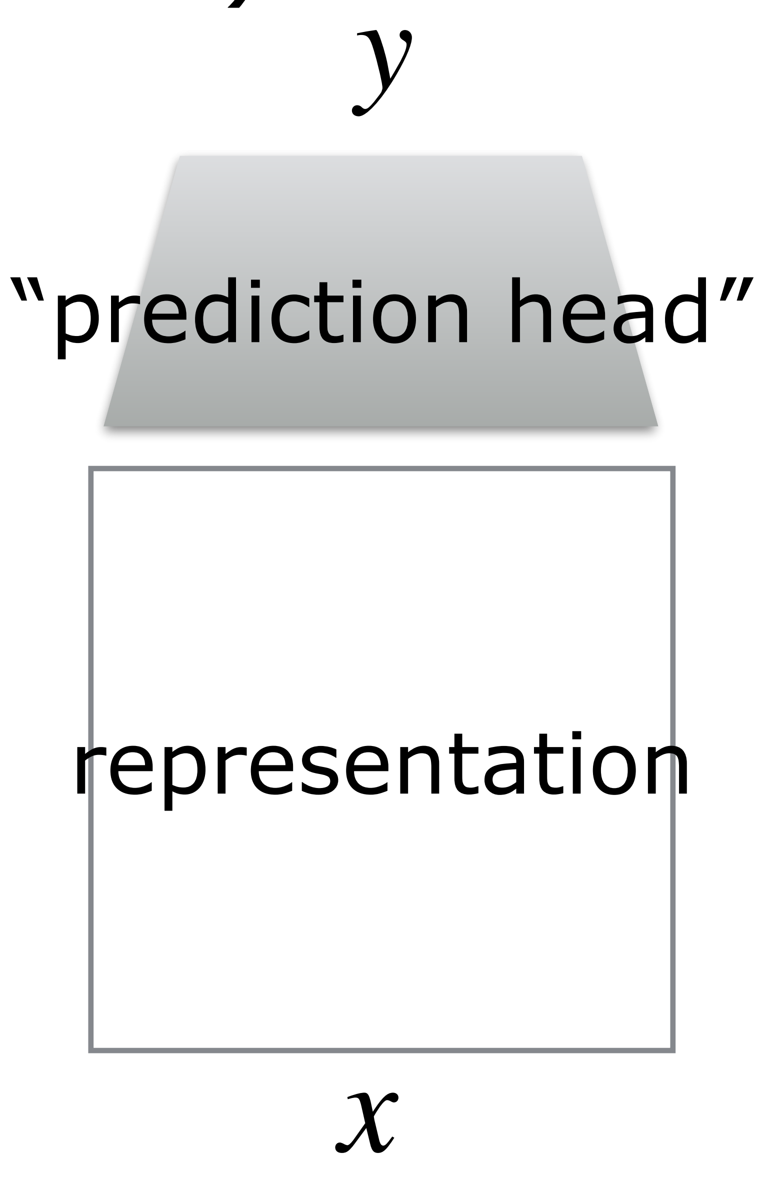
Foundation models

- Foundation models are trained from large amounts of data using auxiliary tasks, then adapted (parts of them reused) for the intended task(s)
- There are many ways to adapt/use these models to new tasks
  - extract useful, broadly useable representations
  - fine-tune (supervised or unsupervised, with or without a new prediction head)
  - use in a few shot prediction (e.g., a few examples available per category)
  - zero shot prediction (no new examples about the intended task; see example later)
- Foundation models are getting larger and larger…



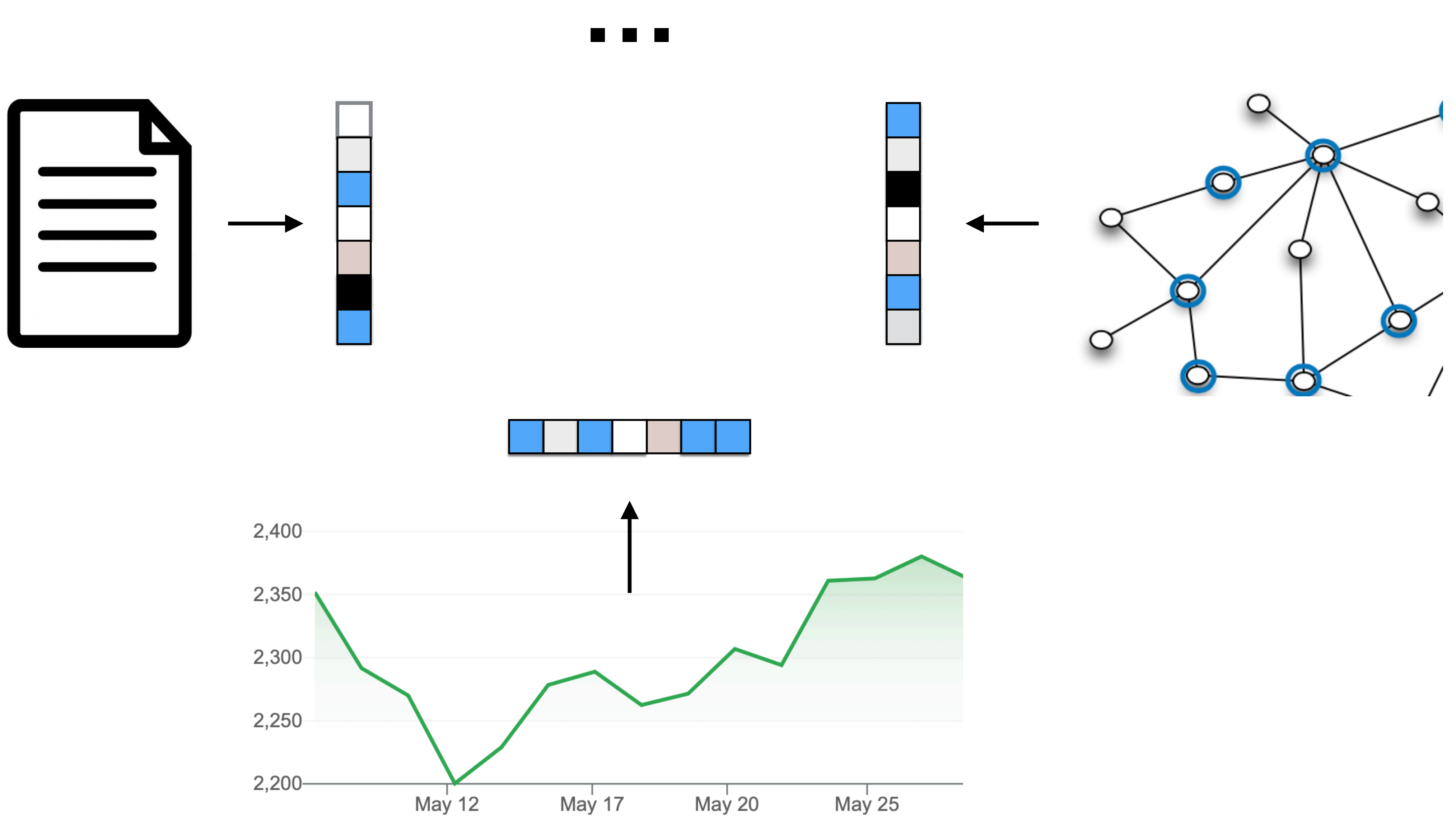
[Microsoft Research Blog 2021]

e.g., fine-tune by training a new prediction head



A step forward: continuous embedding

- We can turn any input into a vector representation (images, text, graphs, time course signals, etc)
- These vector representations can be optimized to preserve relevant information about the objects, often even without any labels!
- One everything is comparable (as vectors), we can easily combine heterogenous data sources, or predict one type from the others
- Some examples
  - users and products (collaborative filtering)
  - words within/across languages
  - images and text (such as captions)
  - service tickets and experts
  - etc.



Recall: matrix factorization

movies/items

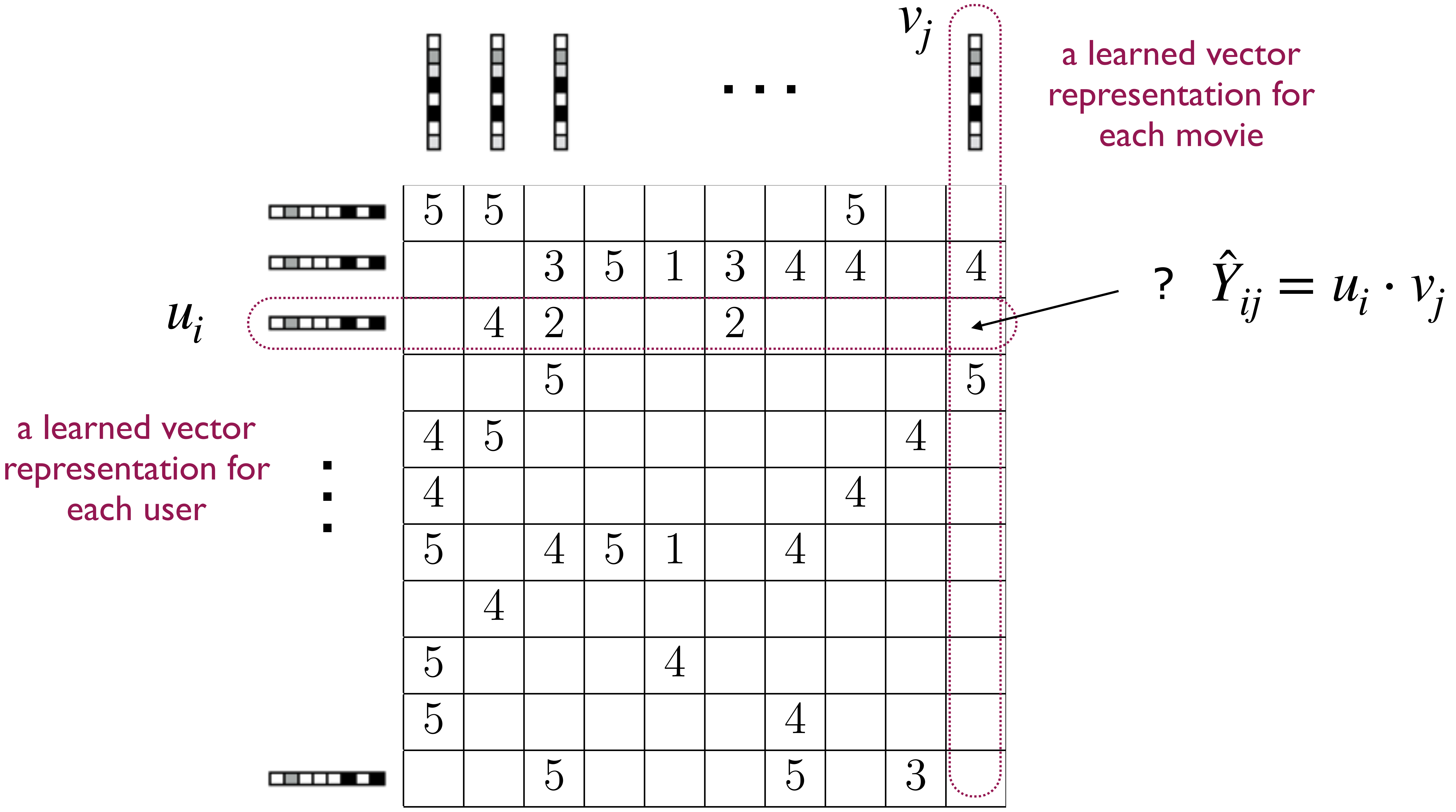
users

5	5						5		
		3	5	1	3	4	4		4
	4	2			2				
		5							5
4	5							4	
4							4		
5		4	5	1		4			
	4								
5				4					
5						4			
		5				5		3	

?

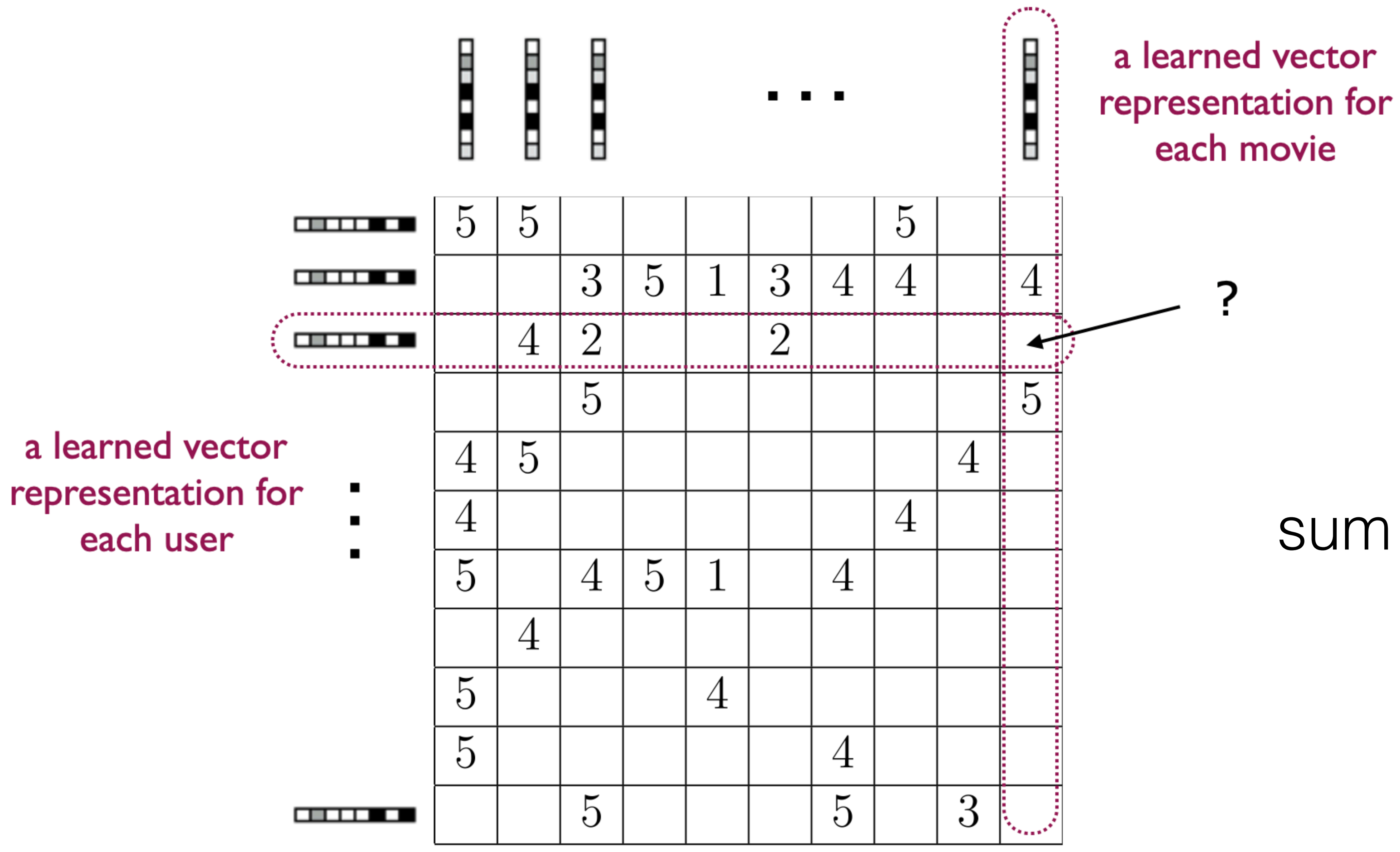
- Vector representations for users and movies are initialized at random, then driven to predict the observed ratings

Recall: matrix factorization



- Vector representations for users and movies are initialized at random, then driven to predict the observed ratings

Supervised embedding

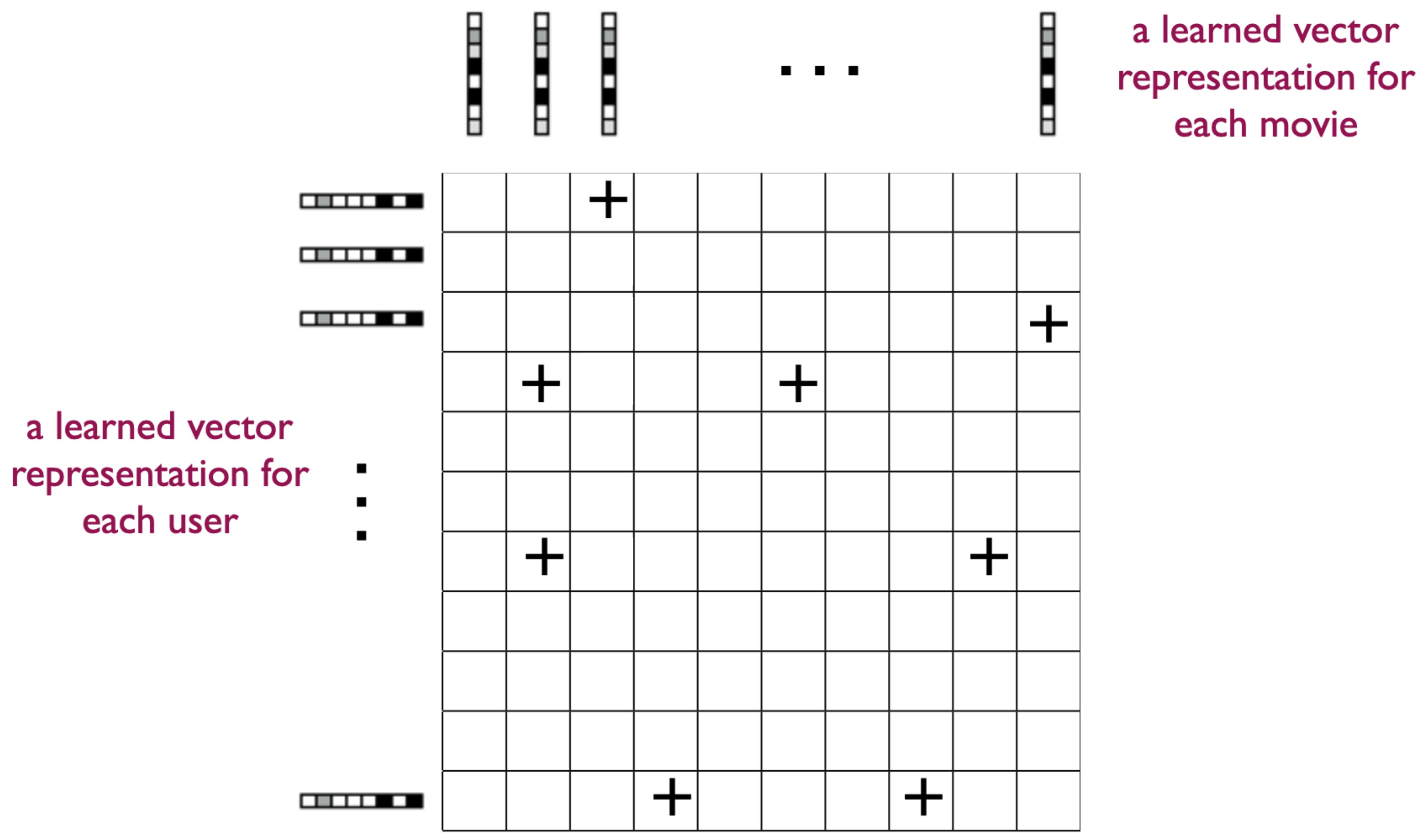


$$\underbrace{\sum_{(i,j)\in D}}_{\text{sum over observations}} (\overset{\text{observed rating}}{Y_{ij}} - u_i \cdot v_j)^2 + \underbrace{\cdots}_{\text{regularization}}$$

- Vector representations for users and movies are initialized at random, then driven to predict the observed ratings

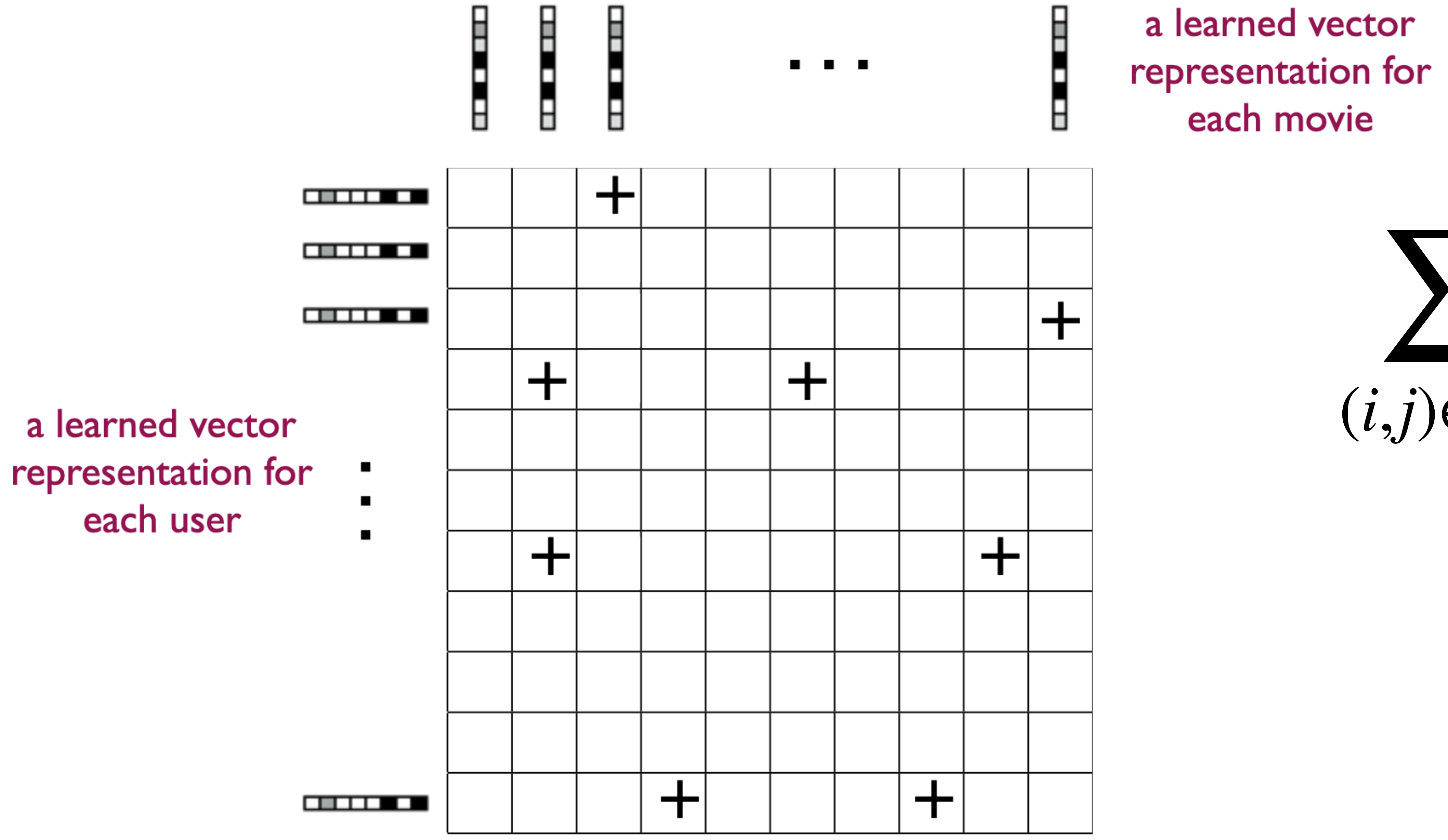
Embedding via co-occurrence

- What if we only have positive “interactions”, not ratings?



Embedding via co-occurrence

- What if we only have positive “interactions”, not ratings?



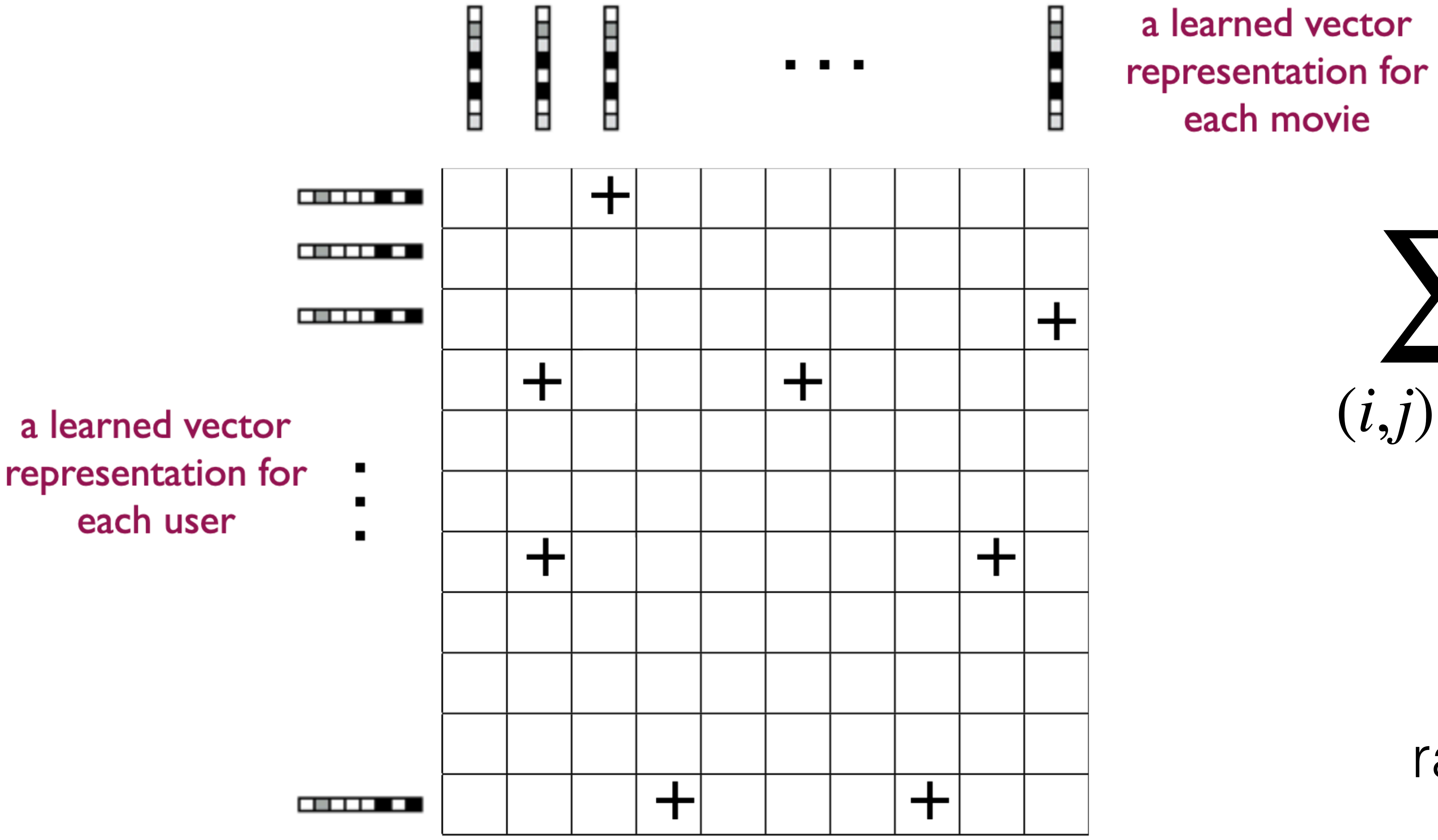
$$\sum_{(i,j)\in D} (1 - u_i \cdot v_j)^2 + \cdots$$

$$\Rightarrow v_i = \vec{1}/\sqrt{d}, \;\; u_j = \vec{1}/\sqrt{d}$$

is a “solution” for all i and j

Embedding via co-occurrence

- What if we only have positive "interactions", not ratings?



$$\sum_{(i,j)\in D} (1 - u_i \cdot v_j)^2 + \cdots$$

$$+ \sum_{(i,j)\in D^-} (0 - u_i \cdot v_j)^2$$

randomly sampled "negatives"

- As a remedy, we could sample and include a set of negatives out of the unobserved entries, and resample these negatives anew for each (gradient) update of u's and v's (contrastive estimation)

Learning to represent words

- One-hot word vectors are not good enough (do not capture similarity)
- E.g., ‘hotel’ and ‘motel’ should likely have a very similar word representation as they are often interchangeable but as one-hot vectors they are always orthogonal!

$$\phi('hotel') = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix} \qquad \phi('motel') = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \cdots \\ 0 \end{bmatrix}$$

Learning to represent words

- One-hot word vectors are not good enough (do not capture similarity)
- Ideally, we’d find low dimensional vectors for words so that their similarity (e.g., cosine similarity) would capture syntactic/semantic relations

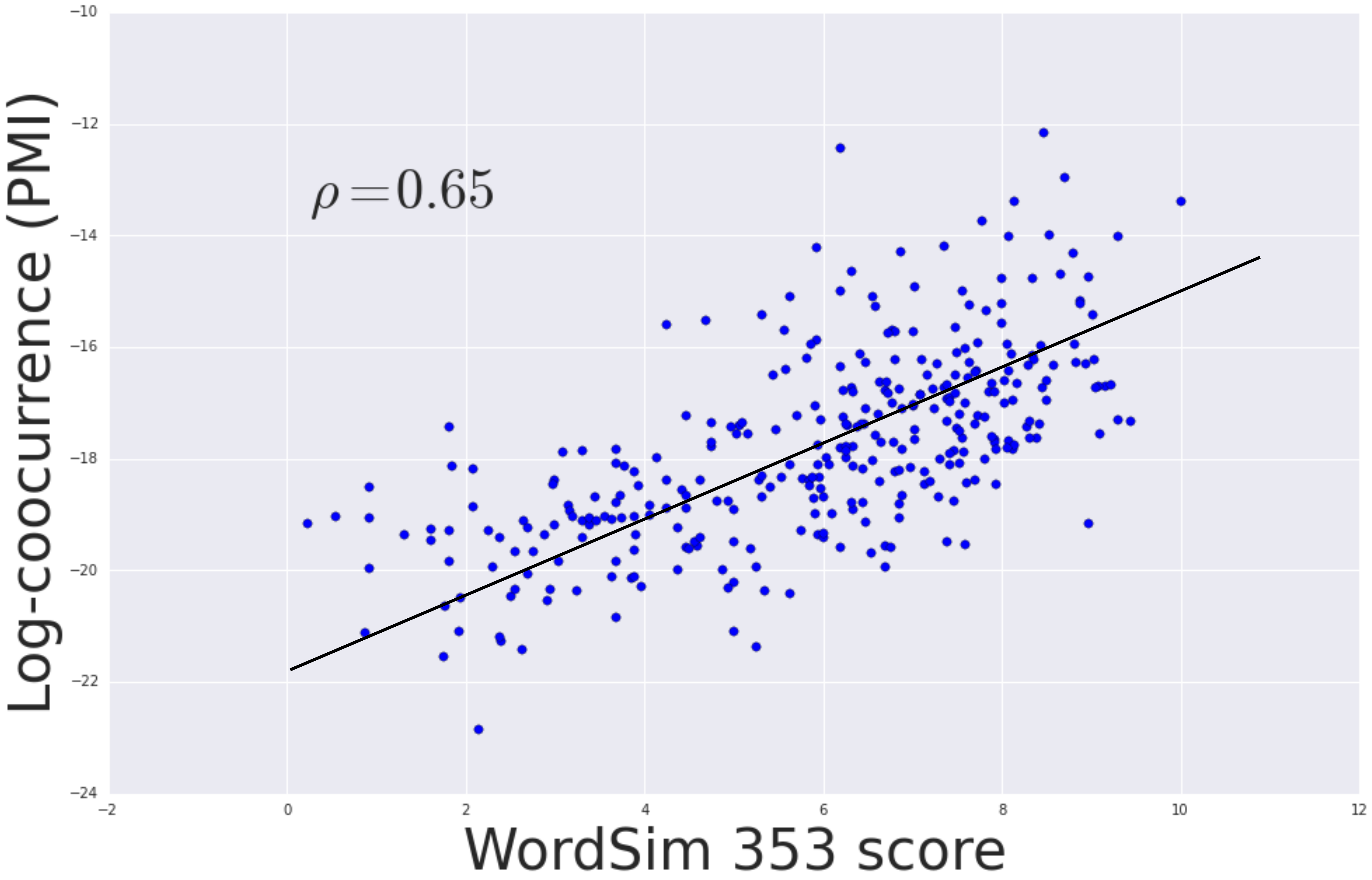
Learning to represent words

- One-hot word vectors are not good enough (do not capture similarity)
- Ideally, we’d find low dimensional vectors for words so that their similarity (e.g., cosine similarity) would capture syntactic/semantic relations
- Co-occurence in sentences across a large corpus provides a sufficient signal to drive these word vector representations to be semantically meaningful

“You shall know a word by the company it keeps”
(Firth, J. R. 1957)

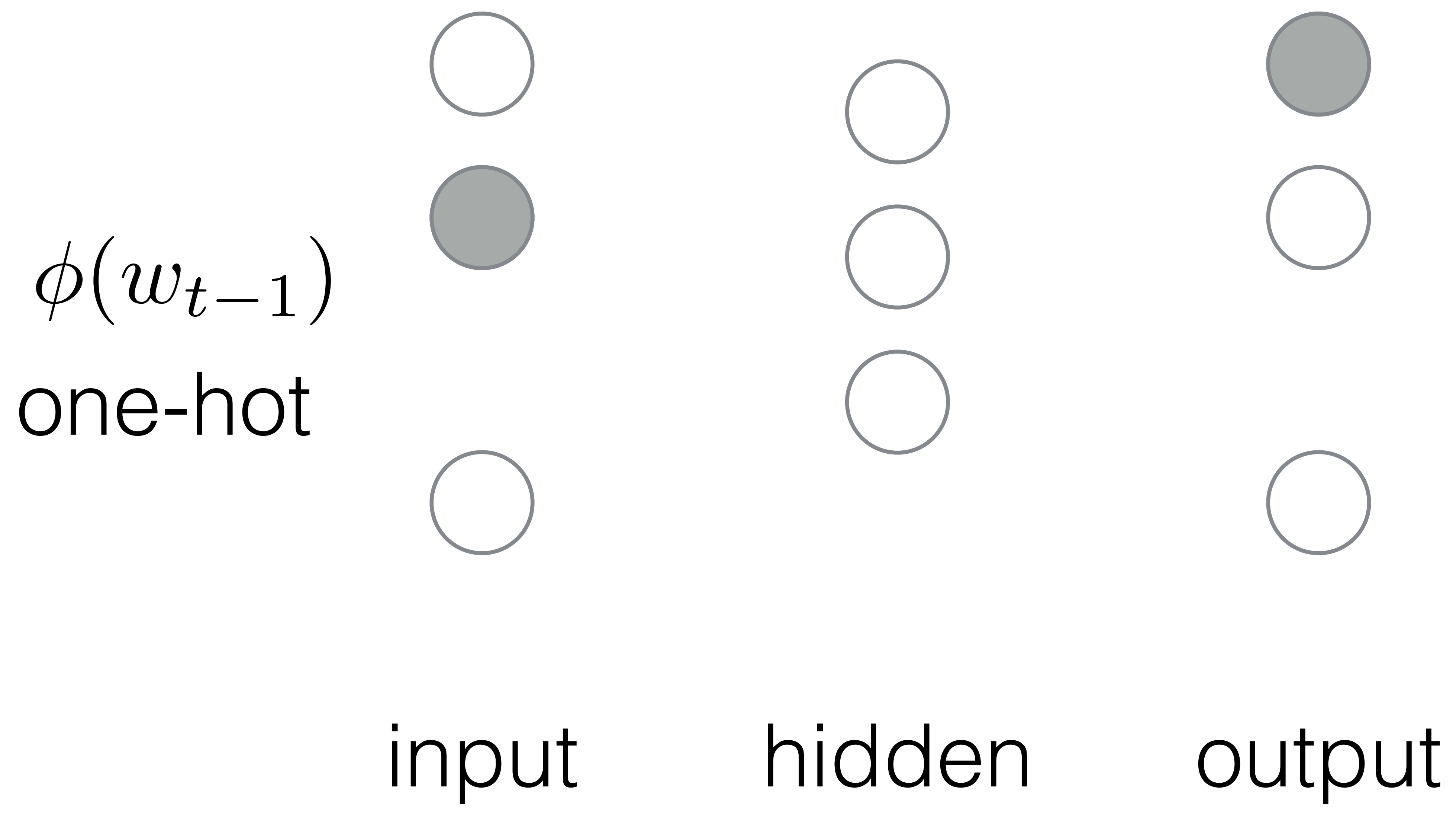
Co-occurence & similarity

- We can obtain semantic similarity assessments between pairs of words from surveys
- Words that co-occur in each other's context in a large corpus tend to be (roughly speaking) semantically similar



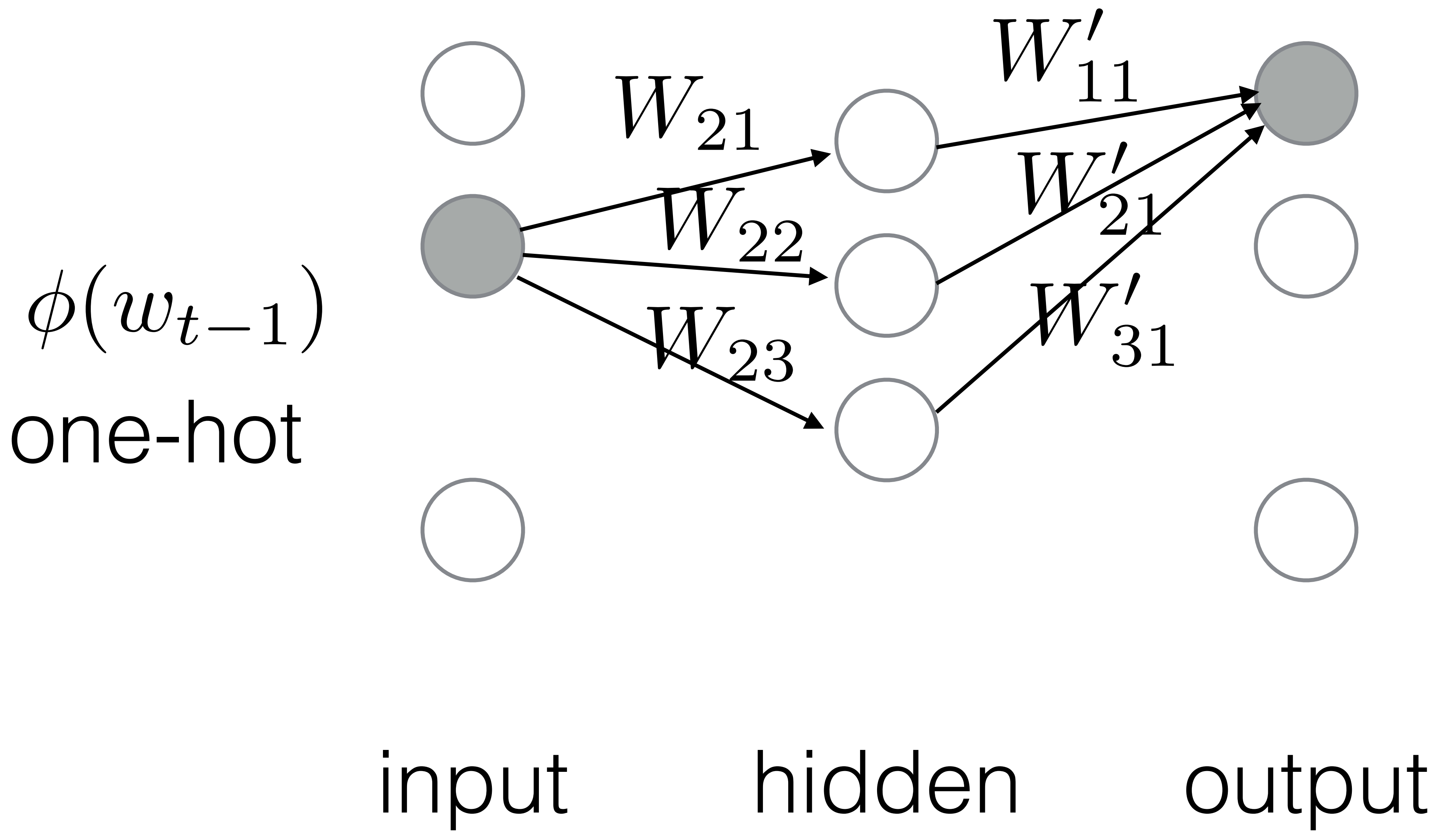
Simple neural bigram model

- Predict each word based on the previous word in the sentence



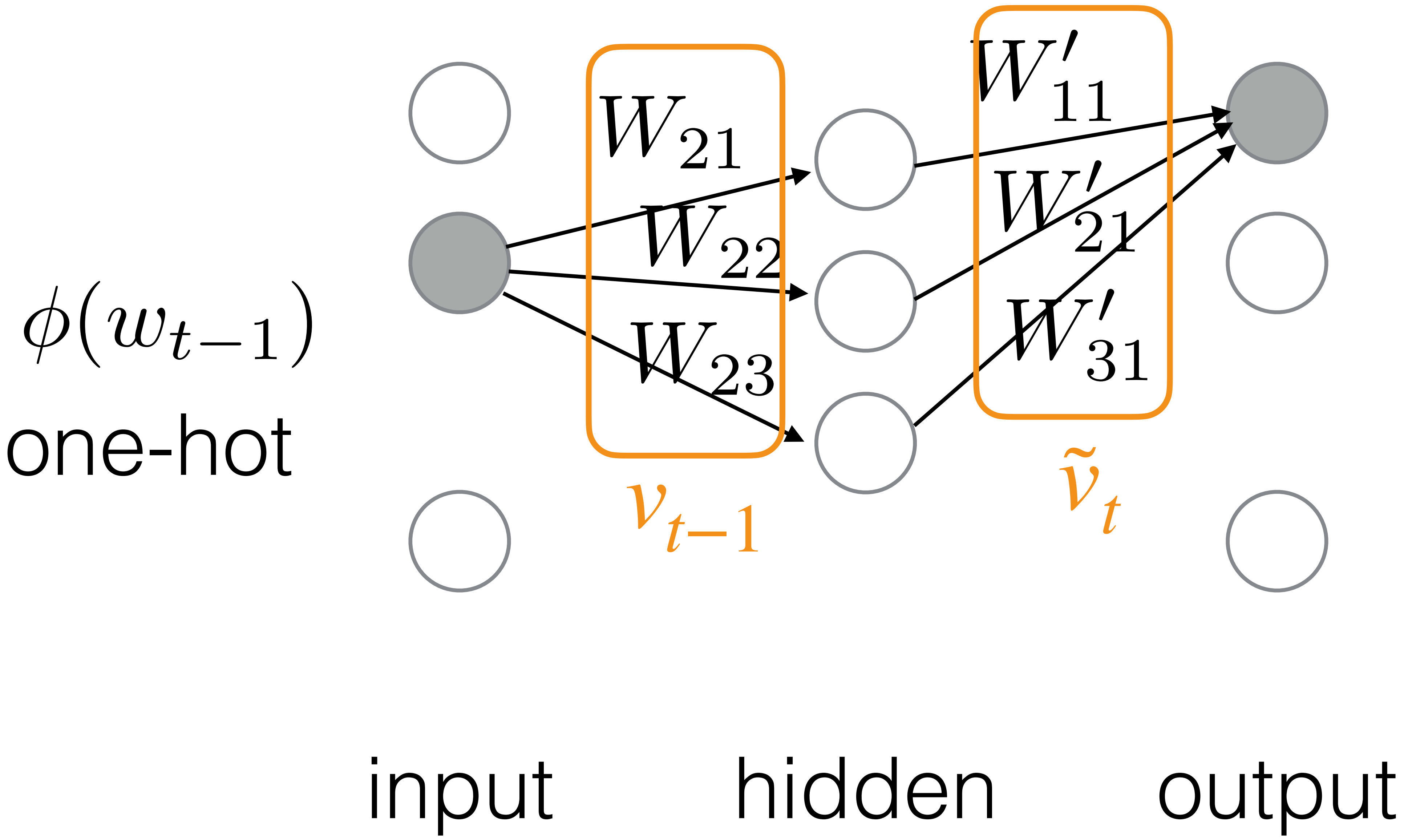
Word vectors from the model

- Predict each word based on the previous word in the sentence
- This simple model already estimates two different low dimensional dense vectors for each word



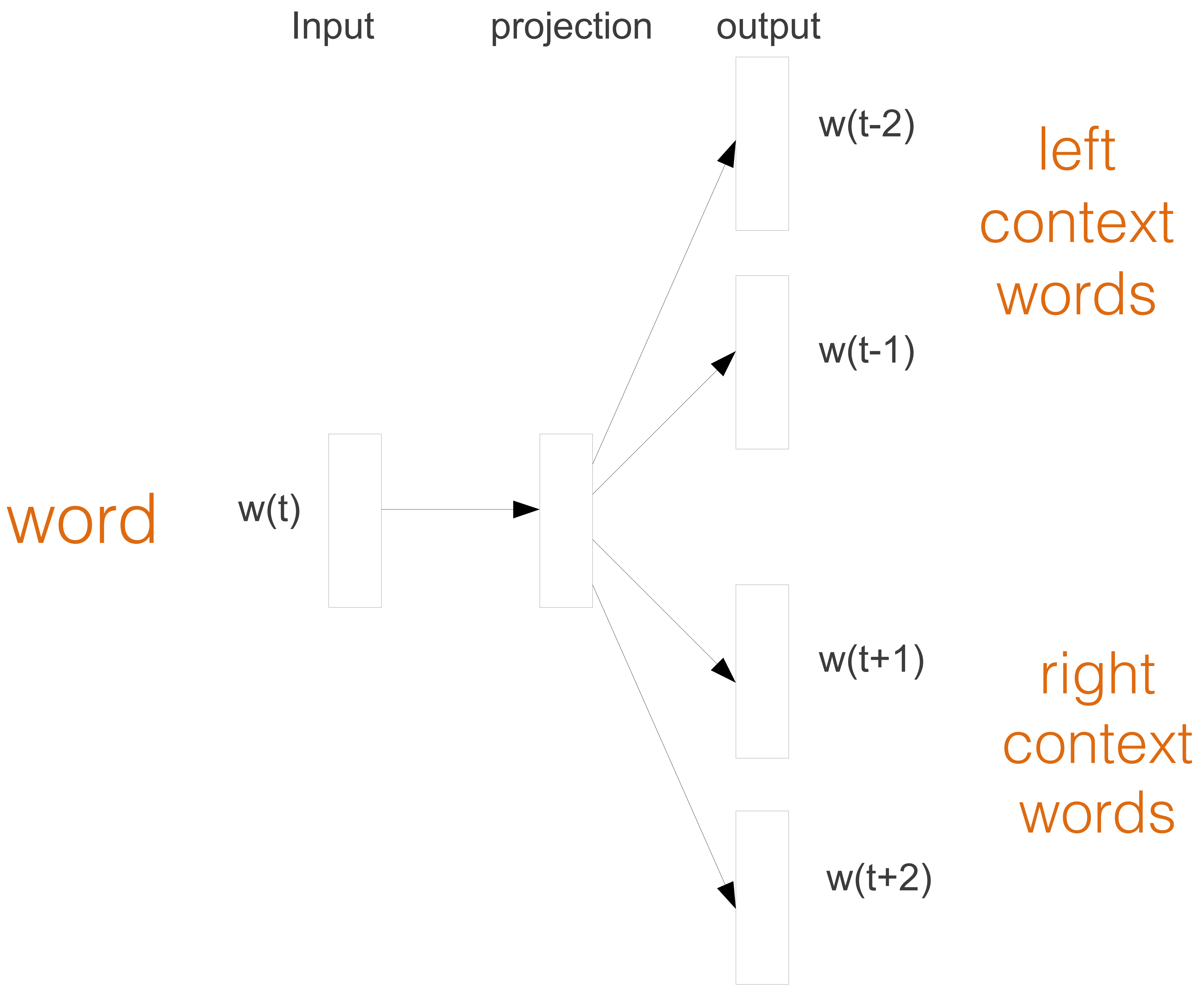
Word vectors from the model

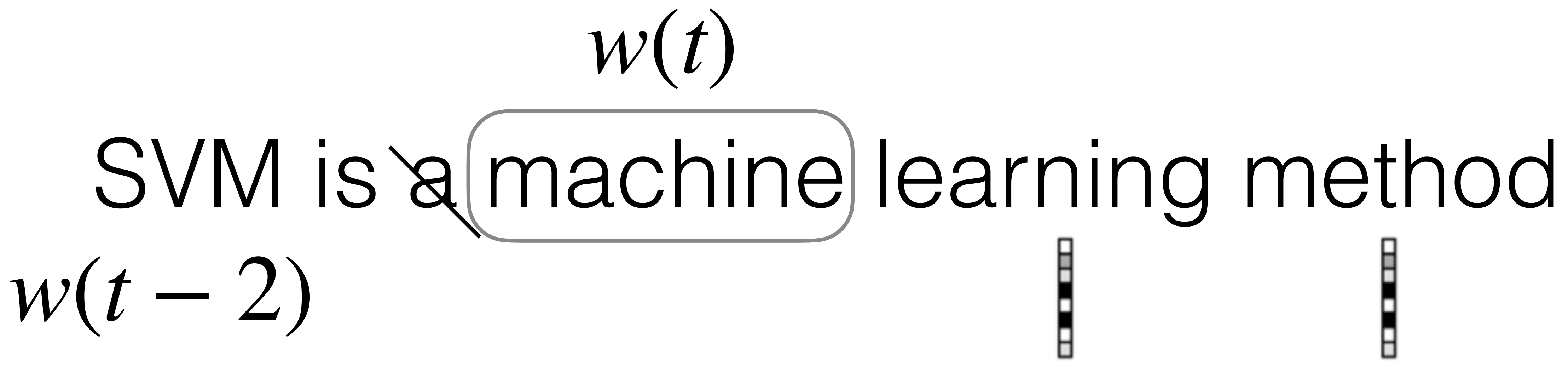
- Predict each word based on the previous word in the sentence
- This simple model already estimates two different low dimensional dense vectors for each word



Example: word2vec

- Skip gram model

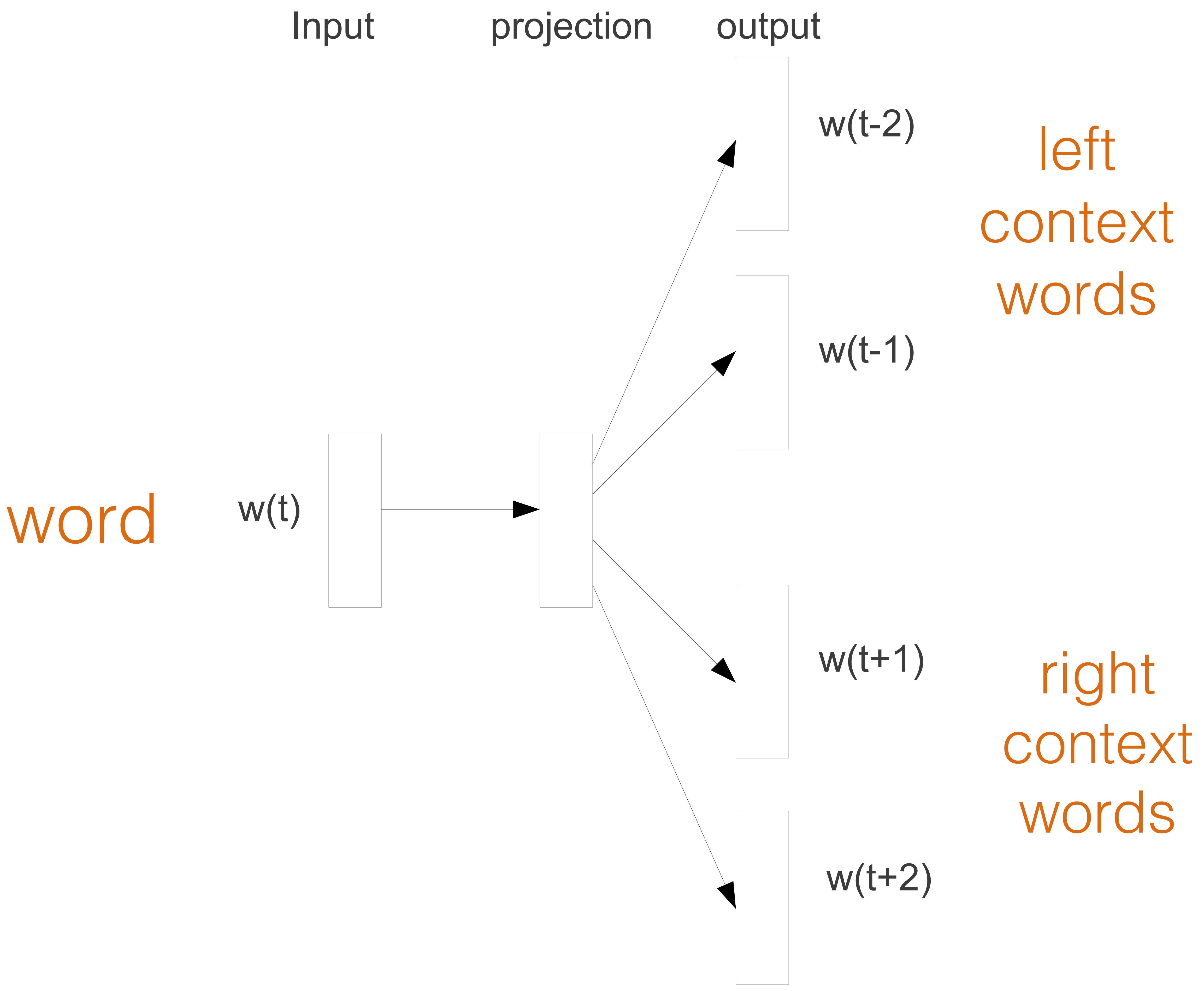




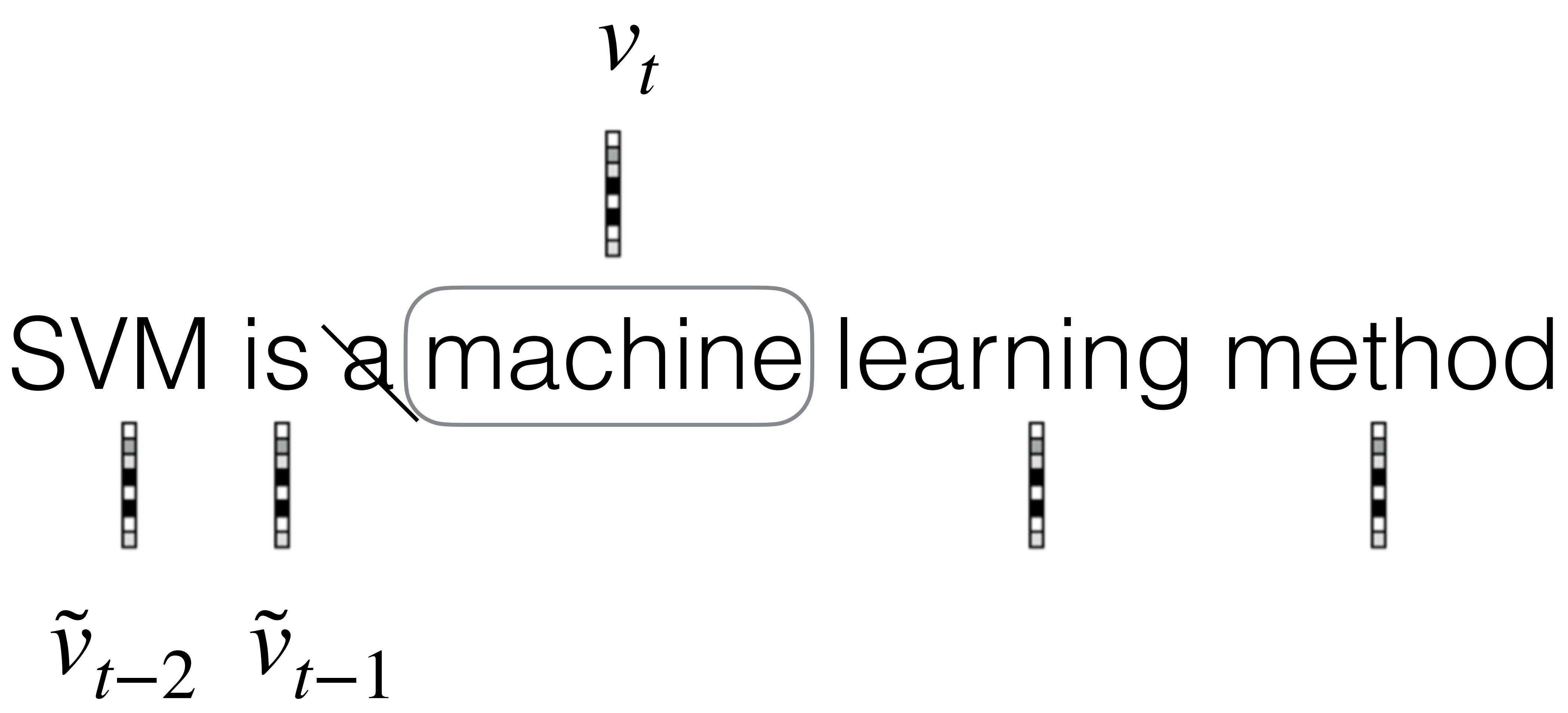
(Mikolov et al 2013)

Example: word2vec

- Skip gram model

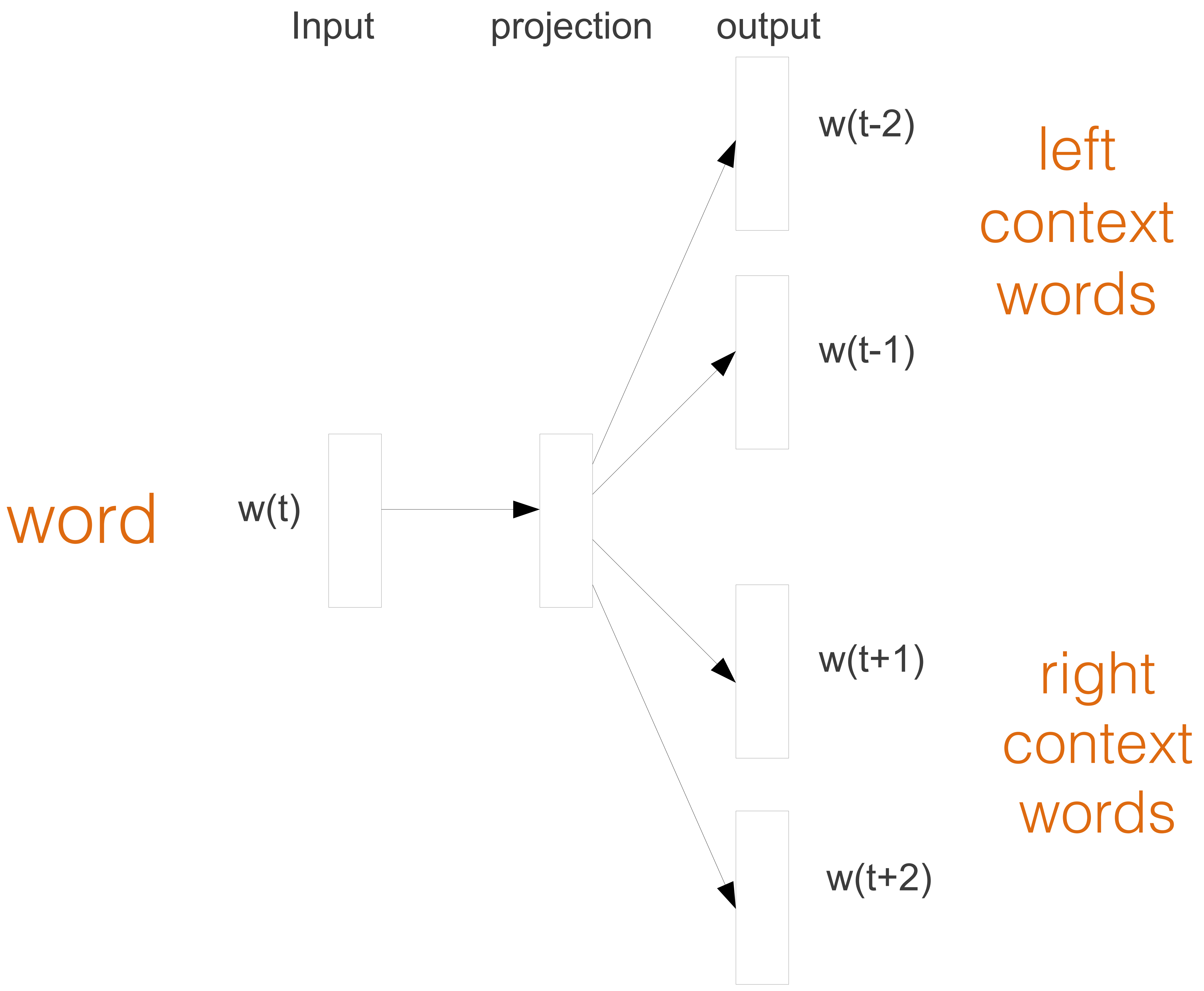


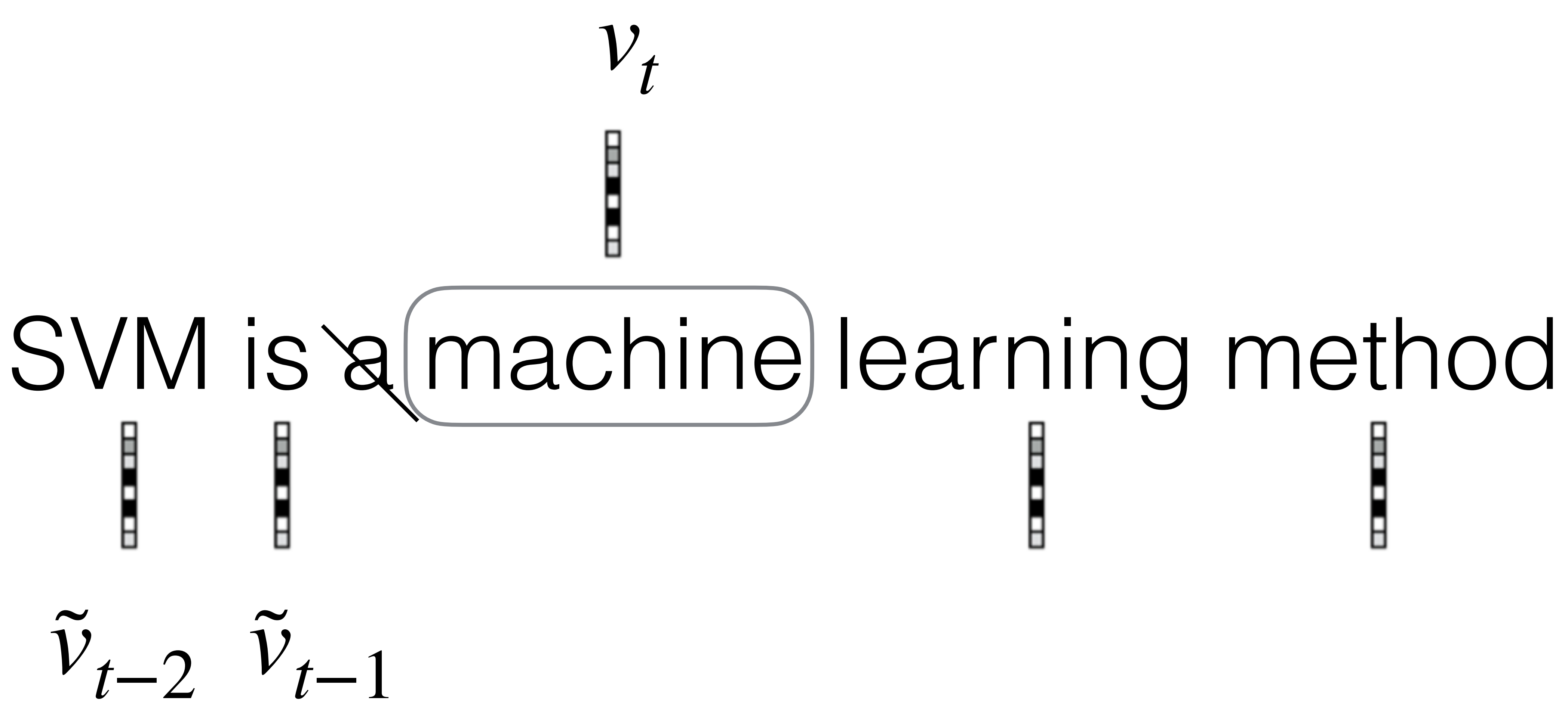
(Mikolov et al 2013)



Example: word2vec

- Skip gram model





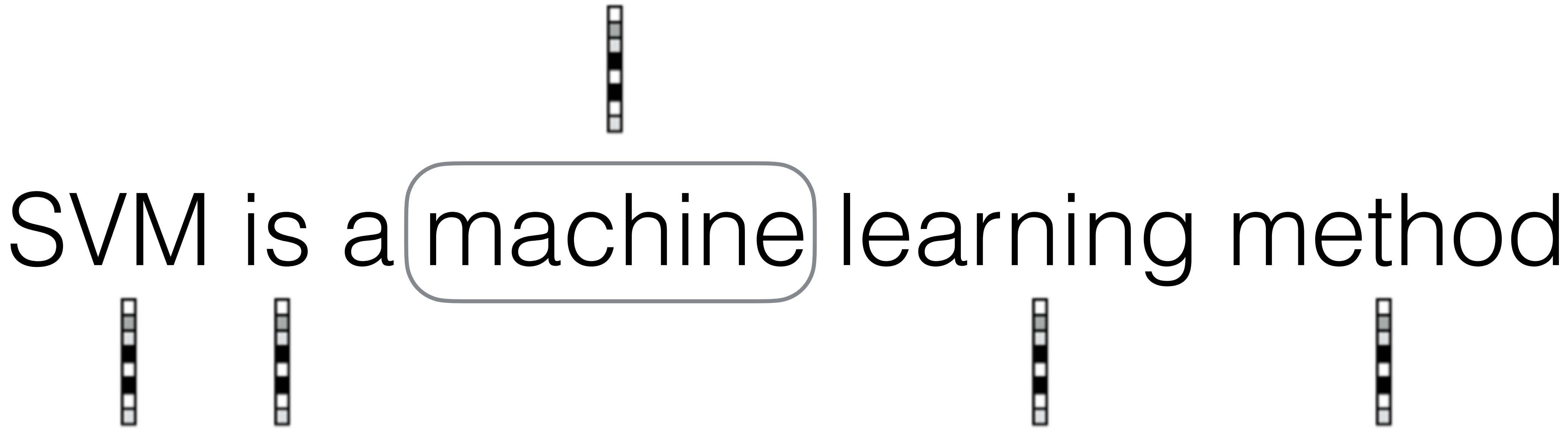
$$P(w(t-2)\,|\,w(t)) = \frac{\exp(v_t \cdot \tilde{v}_{t-2})}{\sum_{\tilde{v}'} \exp(v_t \cdot \tilde{v}')}$$

denominator sums over all the vectors associated with vocab words

(Mikolov et al 2013)

Noise contrastive estimation

- We can also turn the learning problem into a classification task (recognizing words that are in the same context)
- Nearby words are positive examples



- Sampled other K words are negative examples

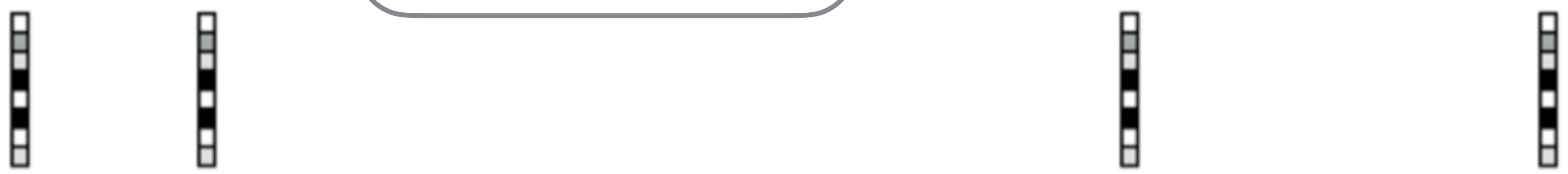
Noise contrastive estimation

- We can also turn the learning problem into a classification task (recognizing words that are in the same context)

$$\sum_i \left[\frac{1}{N(i)} \sum_{j \in N(i)} \log g(v_i \cdot \tilde{v}_j) + \frac{1}{K} \sum_k \log g(-v_i \cdot \tilde{v}_k) \right] \qquad g(z) = \frac{1}{1 + \exp(-z)}$$

- Nearby words are positive examples

SVM is a $\boxed{\text{machine}}$ learning method

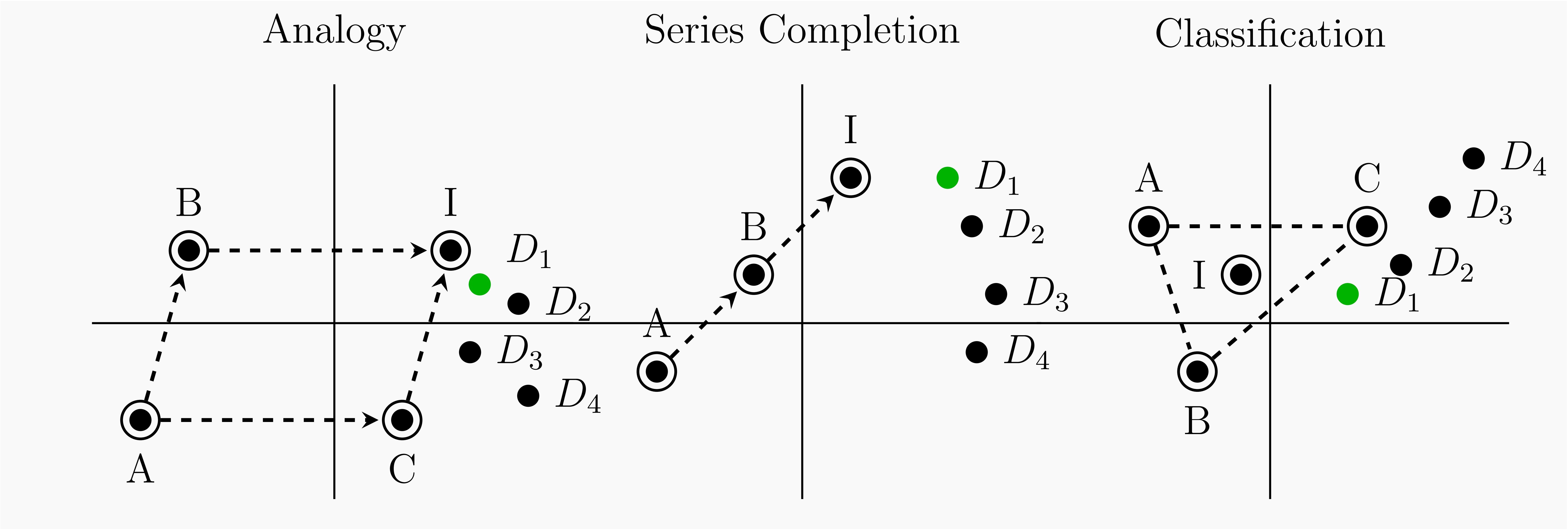


- Sampled other K words are negative examples

Cell biology covers somewhat different material…

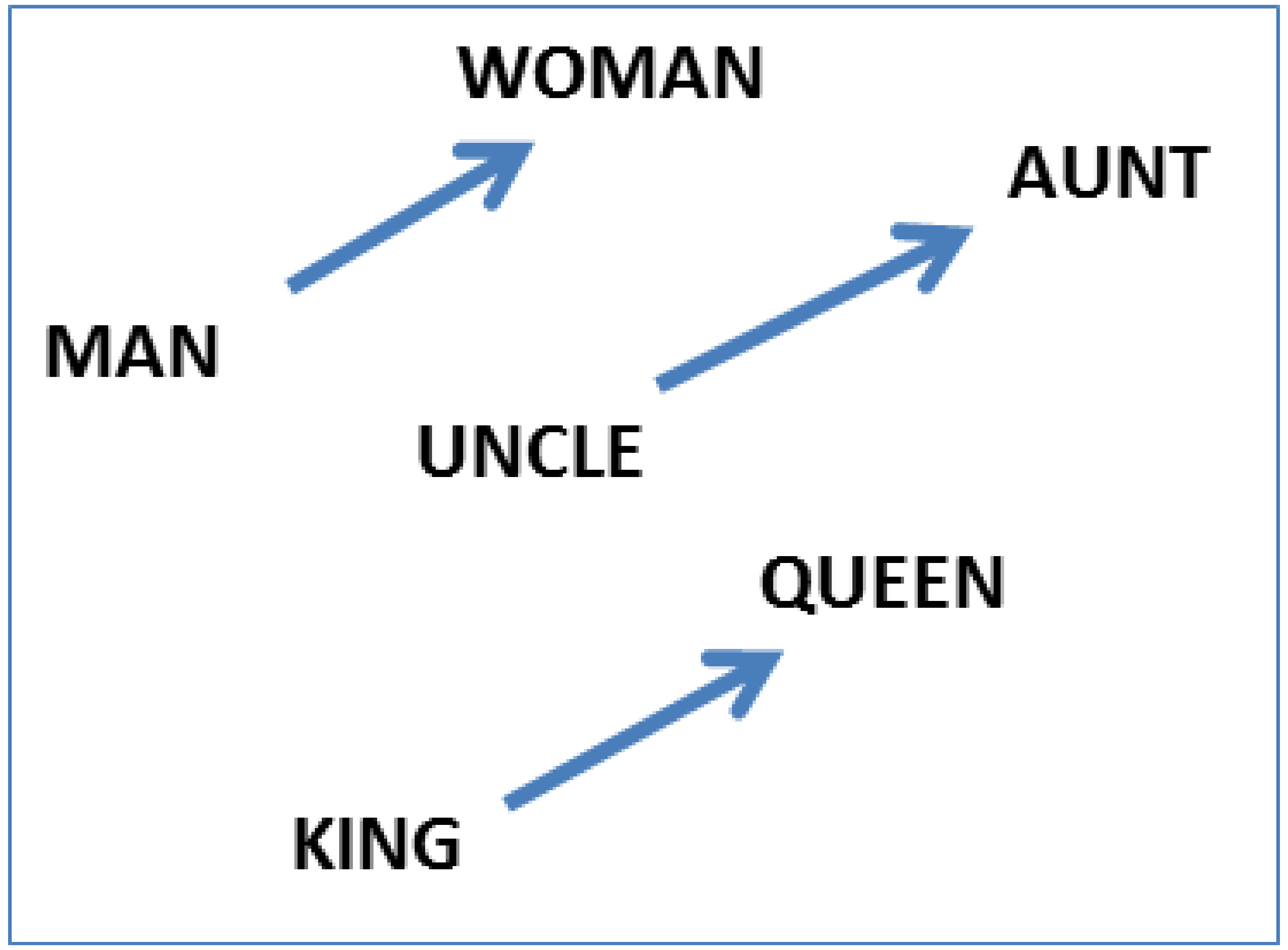
Evaluation: from psychometrics literature

- Word vectors/spaces have long history (e.g., Rumelhart and Abrahamson 73'; Sternberg and Gardner 83')

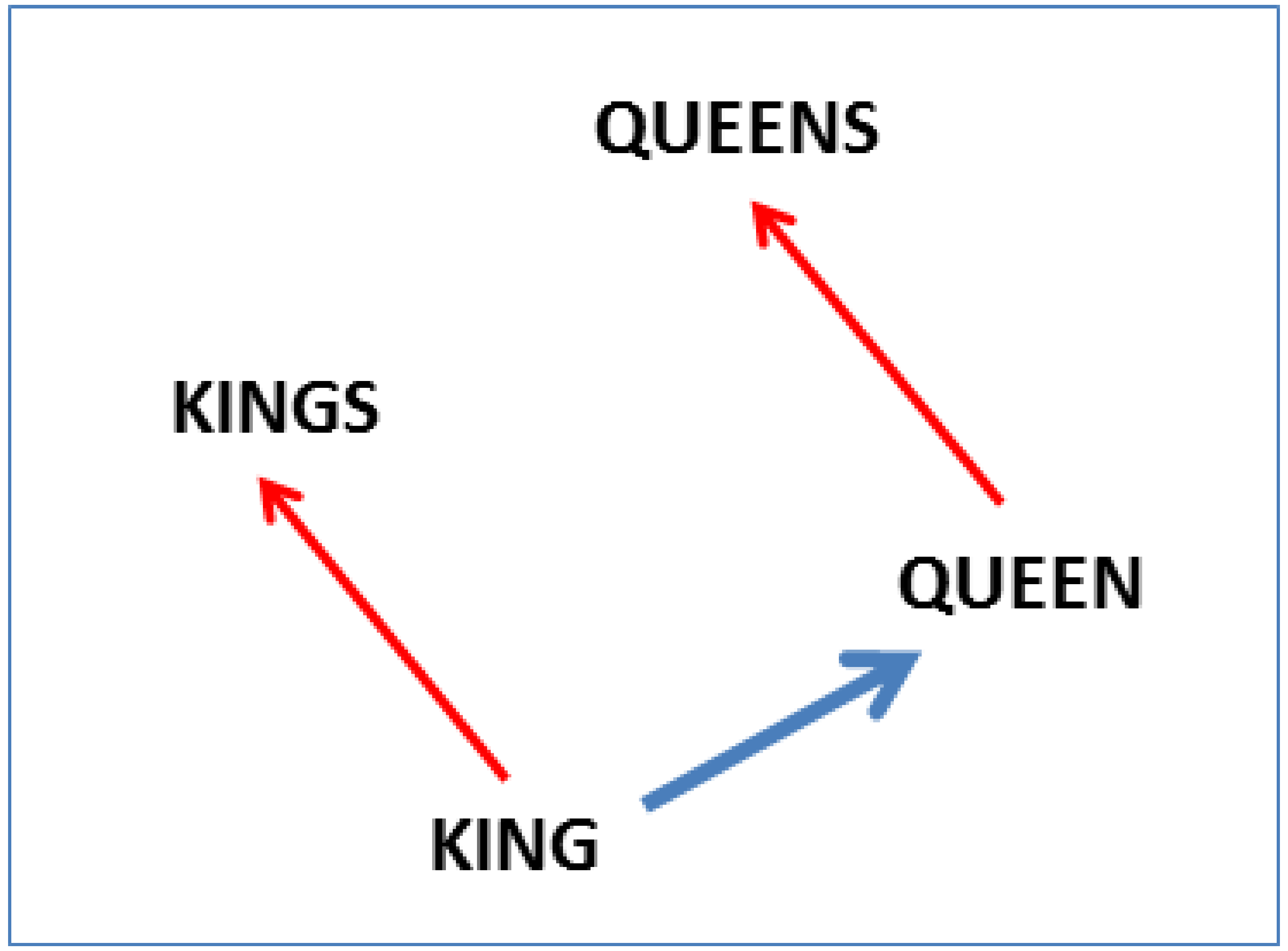


- **analogy**: a king is to man as queen is to ?
- **series completion**: penny, nickel, dime, ?
- **classification**: zebra, giraffe, ? (out of dog, cat, deer)

Testing word vector: analogies



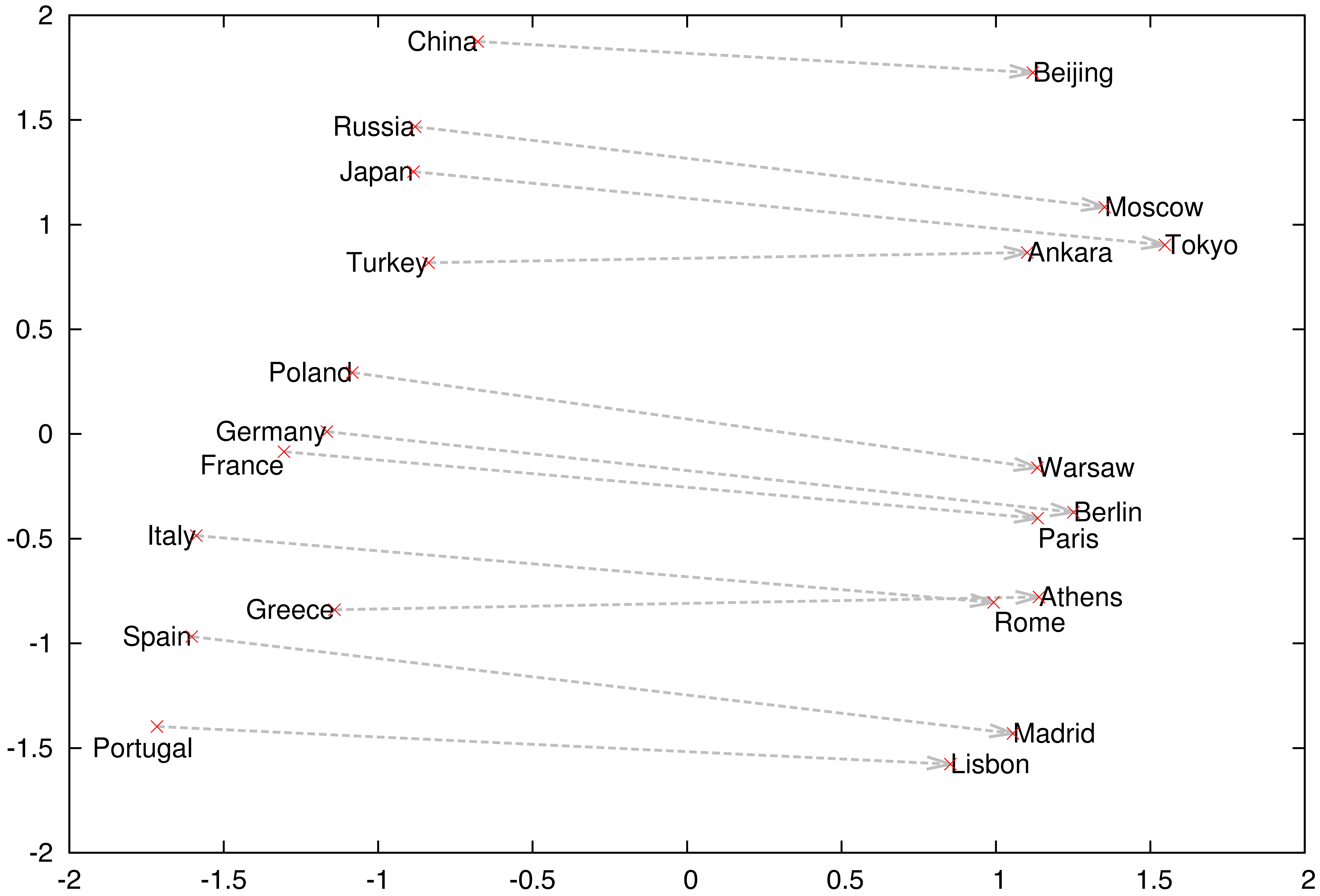
semantic



syntactic

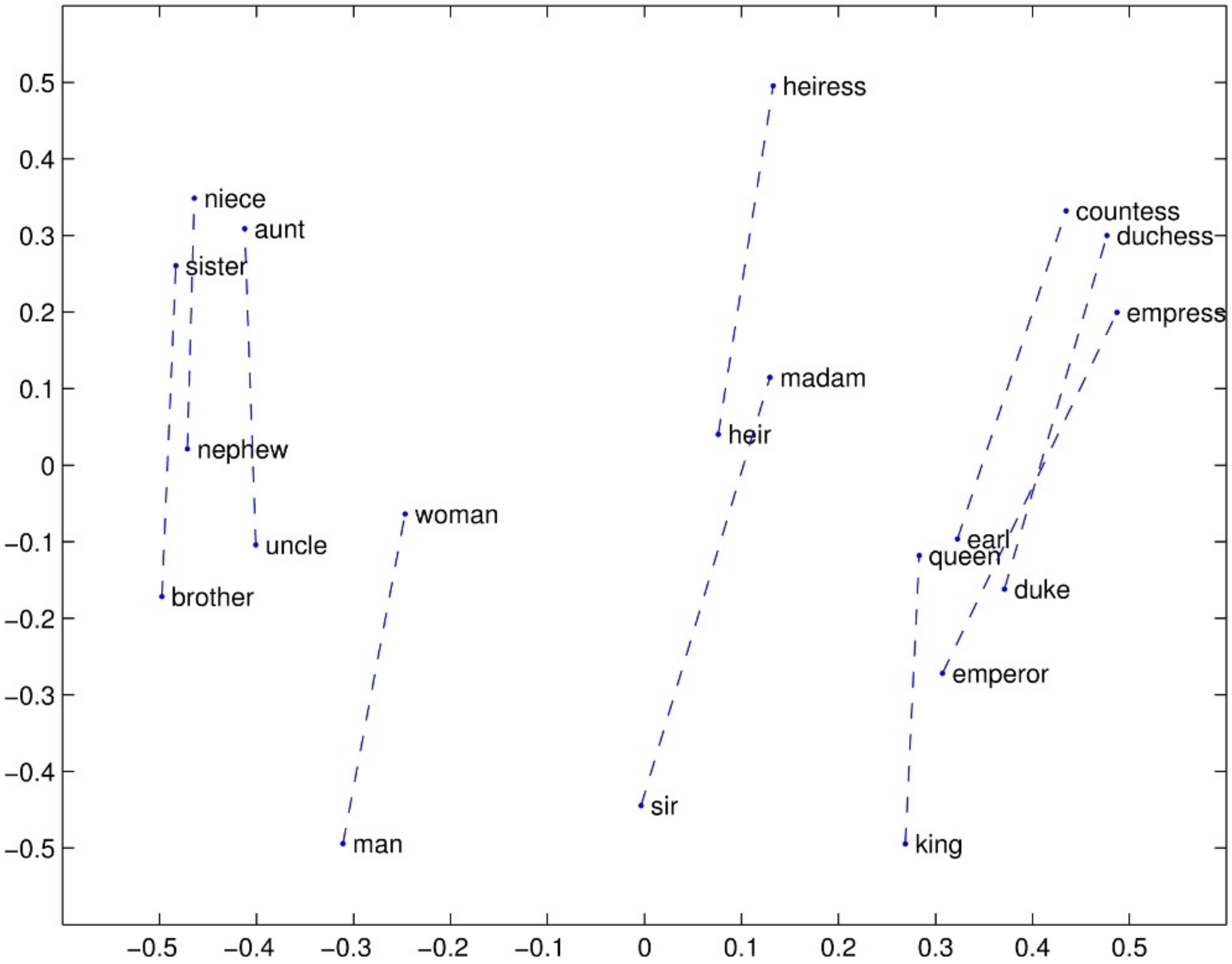
(Mikolov et al., 2013)

Word vector analogies (semantic)



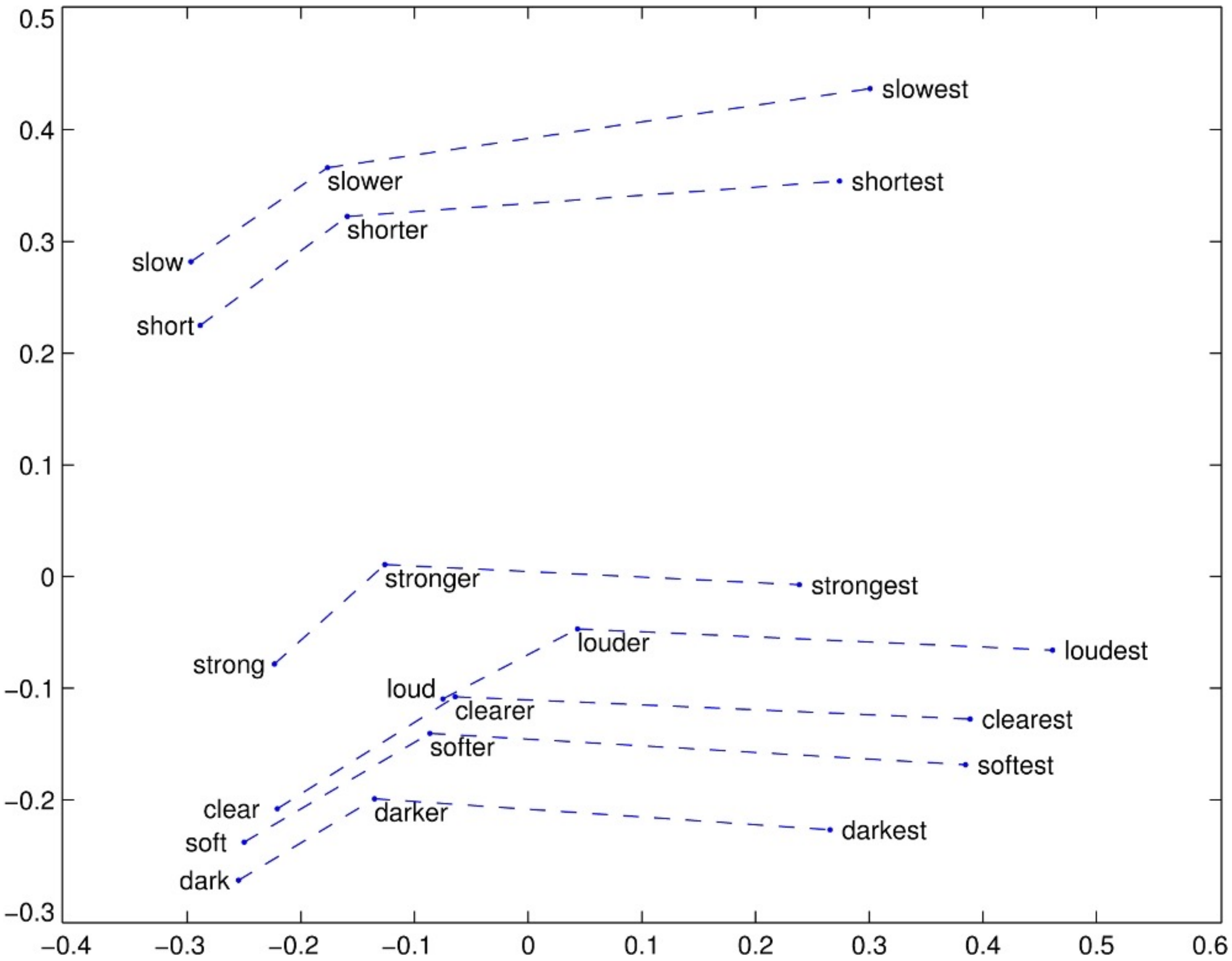
(Mikolov et al., 2013)

Word vector analogies (semantic)



(Pennington et al., 2014)

Word vector analogies (syntactic)



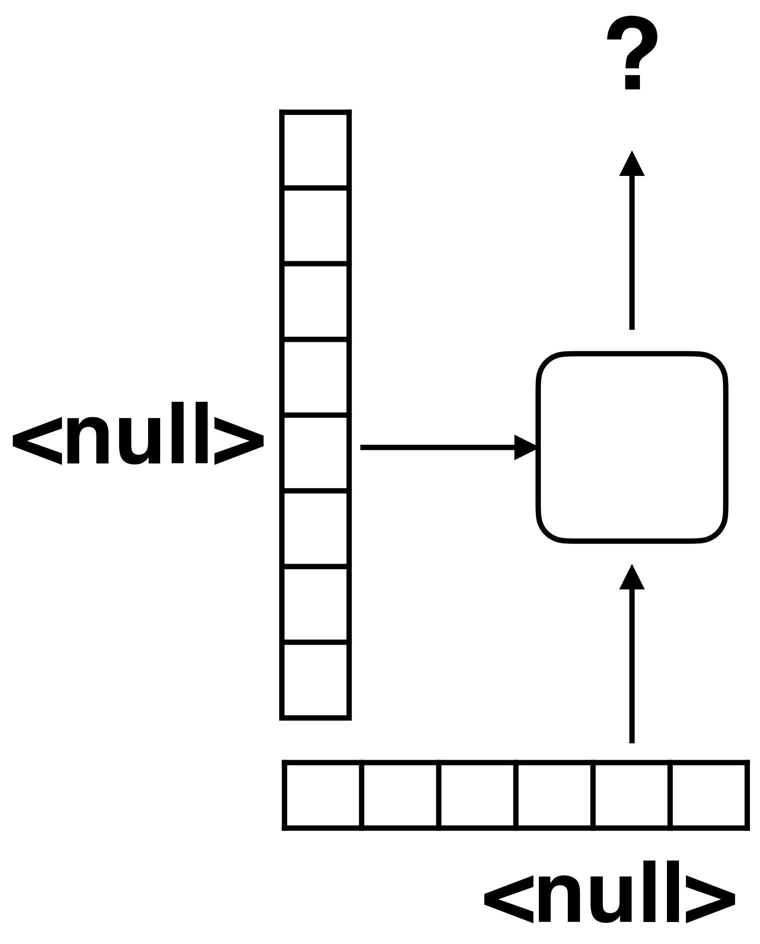
(Pennington et al., 2014)

Beyond bigram/skip gram models

- Autoregressive models for language generate one word at a time, conditioning the generation of each word on the previously generated text

$$P(x_1 \,|\, x_0)$$

E.g., RNN



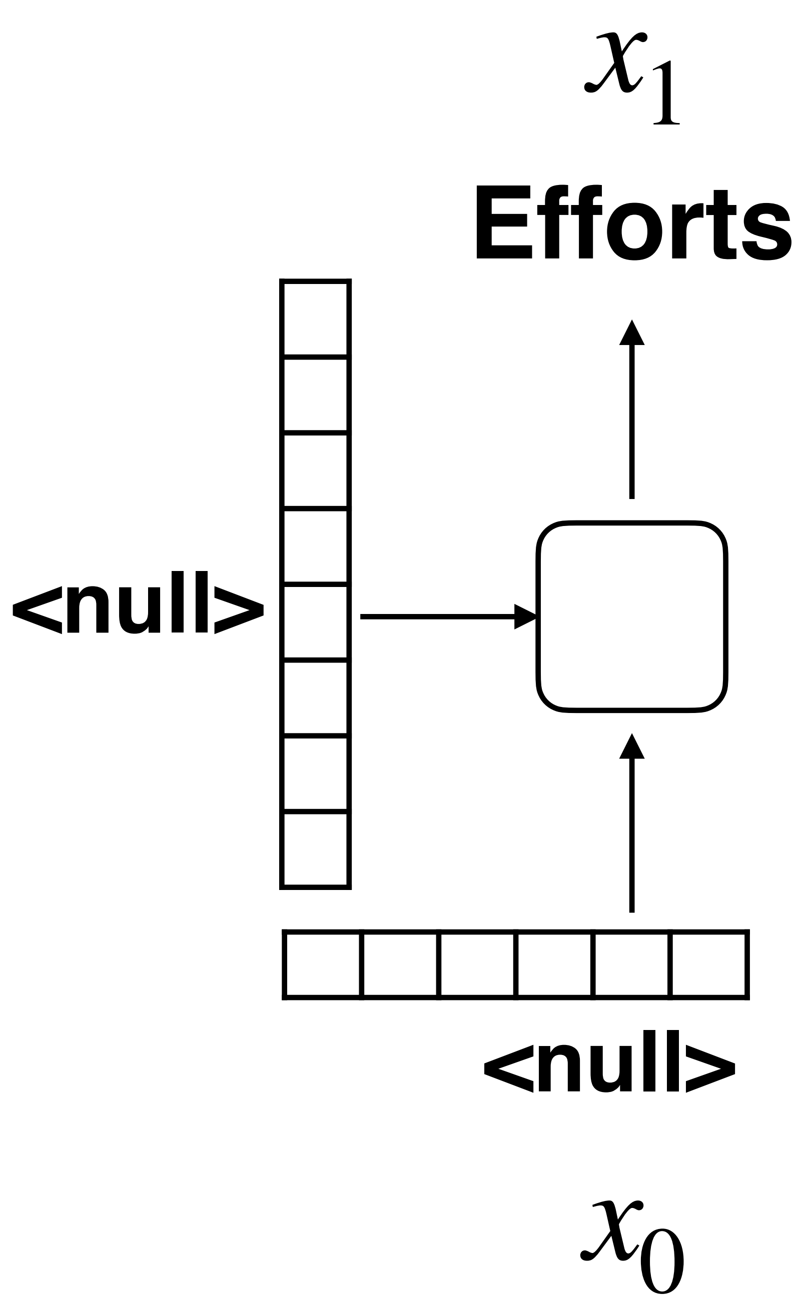
x_0

Beyond bigram/skip gram models

- Autoregressive models for language generate one word at a time, conditioning the generation of each word on the previously generated text

$$P(x_1 \mid x_0)$$

E.g., RNN

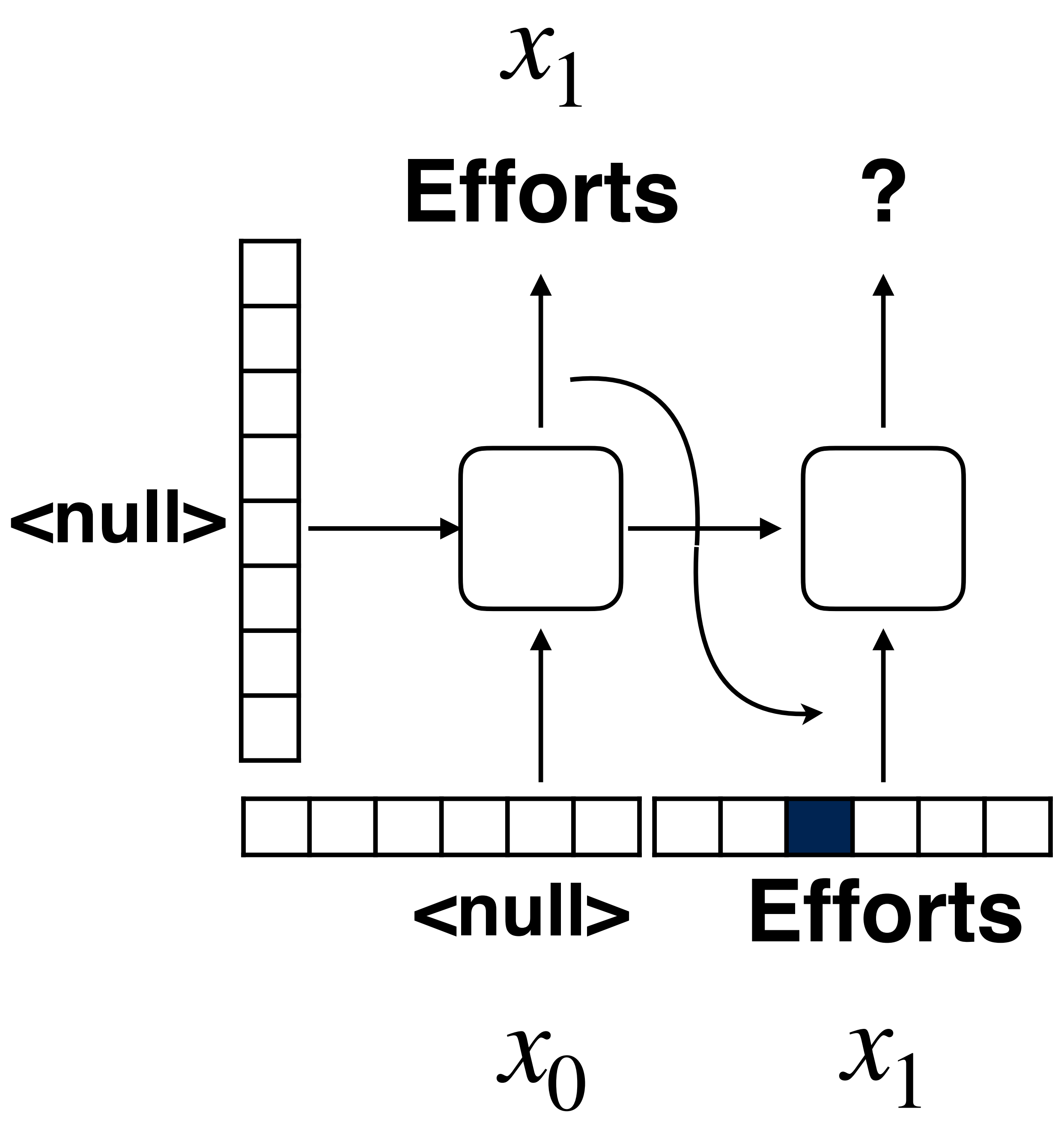


Beyond bigram/skip gram models

- Autoregressive models for language generate one word at a time, conditioning the generation of each word on the previously generated text

$$P(x_1 \mid x_0)P(x_2 \mid x_1, x_0)$$

E.g., RNN

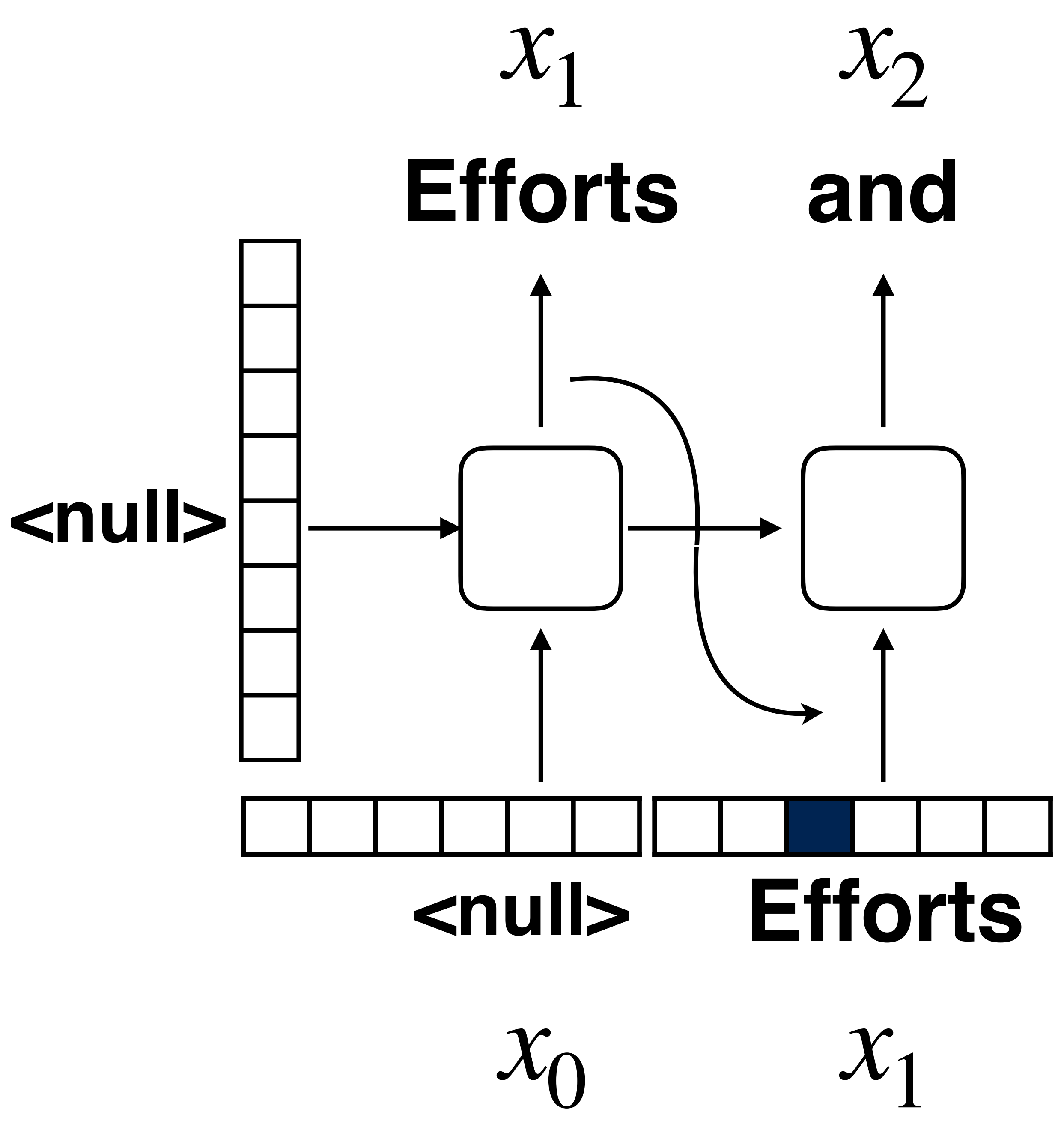


Beyond bigram/skip gram models

- Autoregressive models for language generate one word at a time, conditioning the generation of each word on the previously generated text

$$P(x_1 | x_0) P(x_2 | x_1, x_0)$$

E.g., RNN

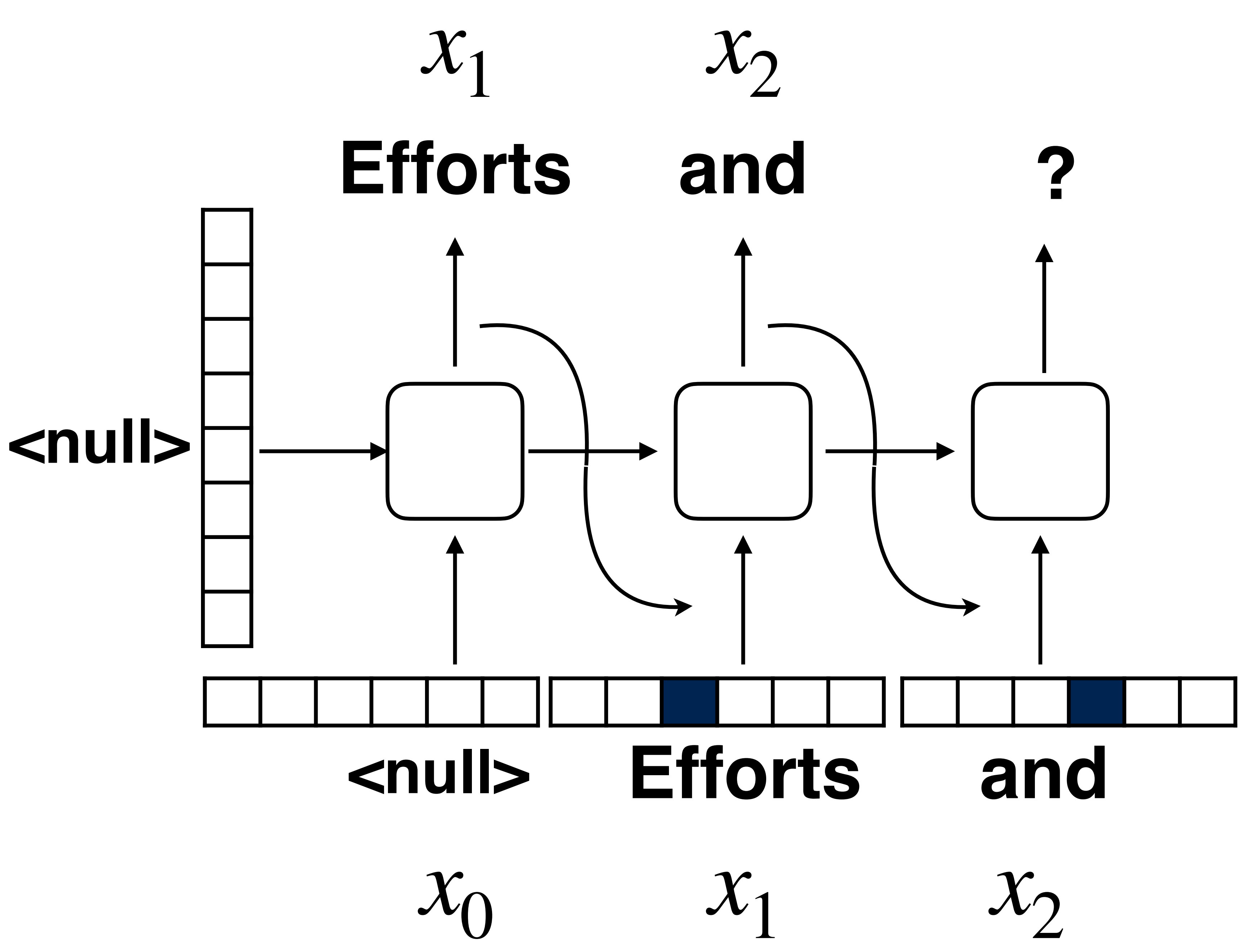


Beyond bigram/skip gram models

- Autoregressive models for language generate one word at a time, conditioning the generation of each word on the previously generated text

$$P(x_1 | x_0)P(x_2 | x_1, x_0)P(x_3 | x_2, x_1, x_0)$$

E.g., RNN

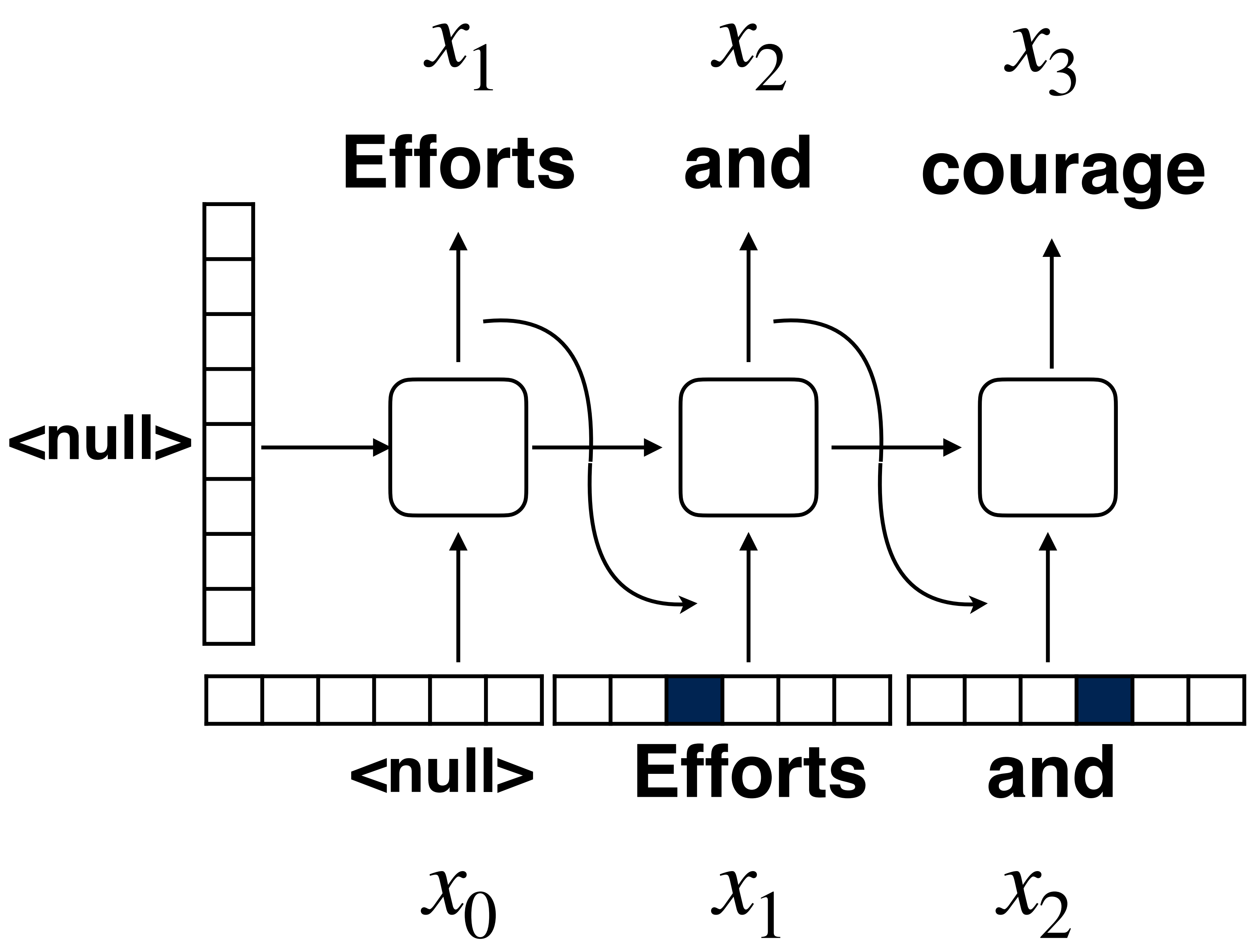


Beyond bigram/skip gram models

- Autoregressive models for language generate one word at a time, conditioning the generation of each word on the previously generated text

$$P(x_1 | x_0)P(x_2 | x_1, x_0)P(x_3 | x_2, x_1, x_0)$$

E.g., RNN

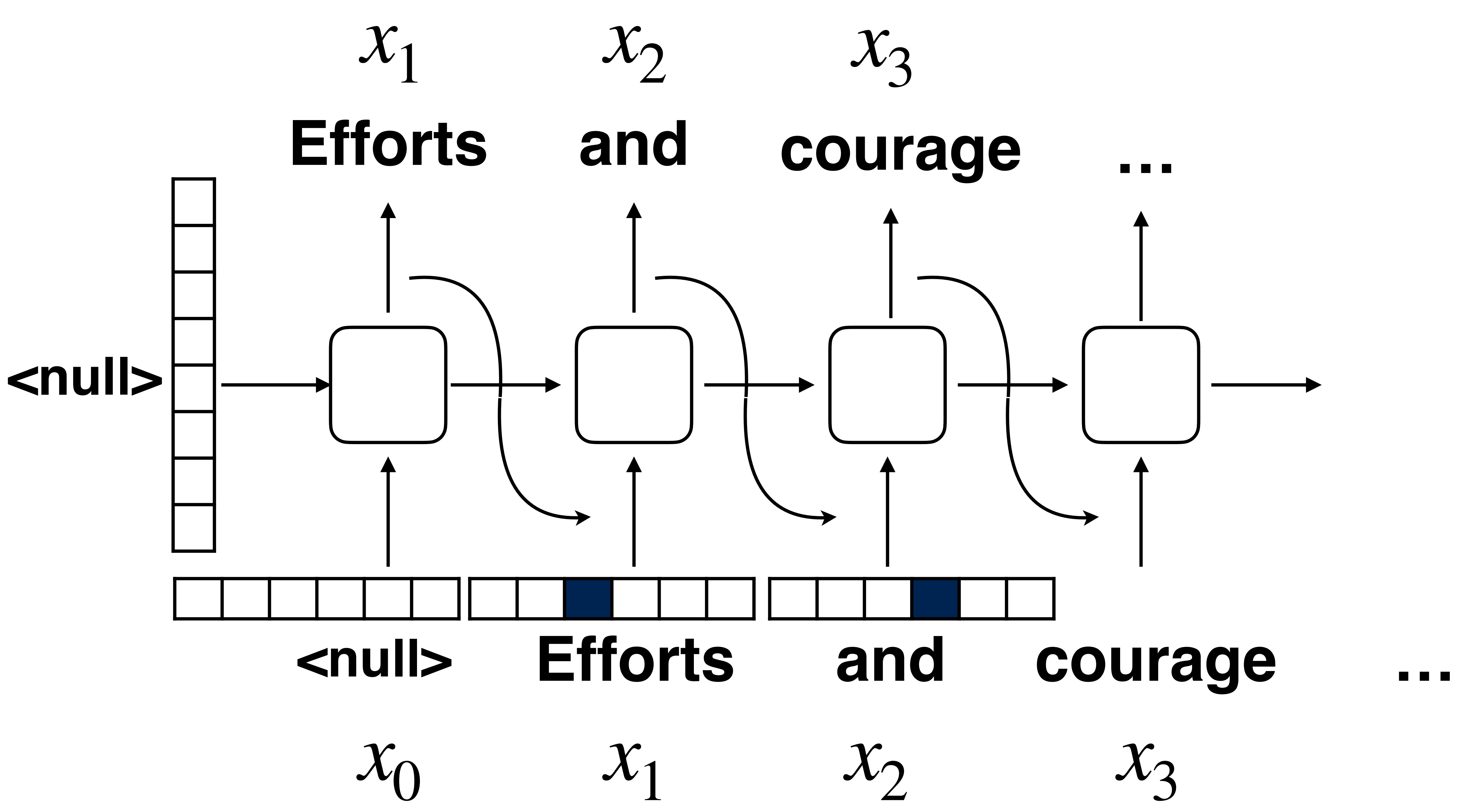


Beyond bigram/skip gram models

- Autoregressive models for language generate one word at a time, conditioning the generation of each word on the previously generated text

$$P(x_1 \mid x_0)P(x_2 \mid x_1, x_0)P(x_3 \mid x_2, x_1, x_0)P(x_4 \mid x_3, x_2, x_1, x_0)\cdots$$

E.g., RNN

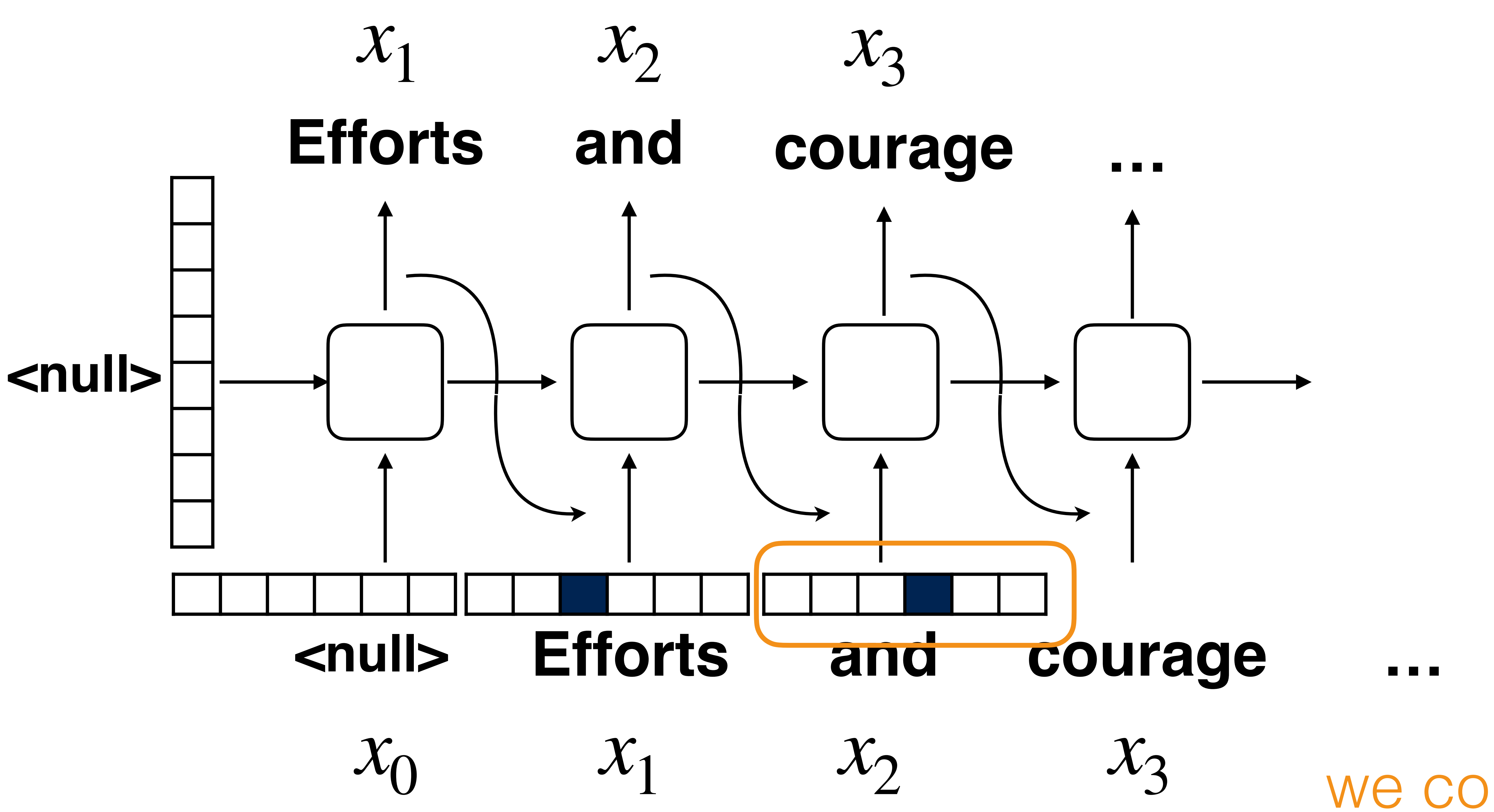


Beyond bigram/skip gram models

- Autoregressive models for language generate one word at a time, conditioning the generation of each word on the previously generated text

$$P(x_1 | x_0) P(x_2 | x_1, x_0) P(x_3 | x_2, x_1, x_0) P(x_4 | x_3, x_2, x_1, x_0) \cdots$$

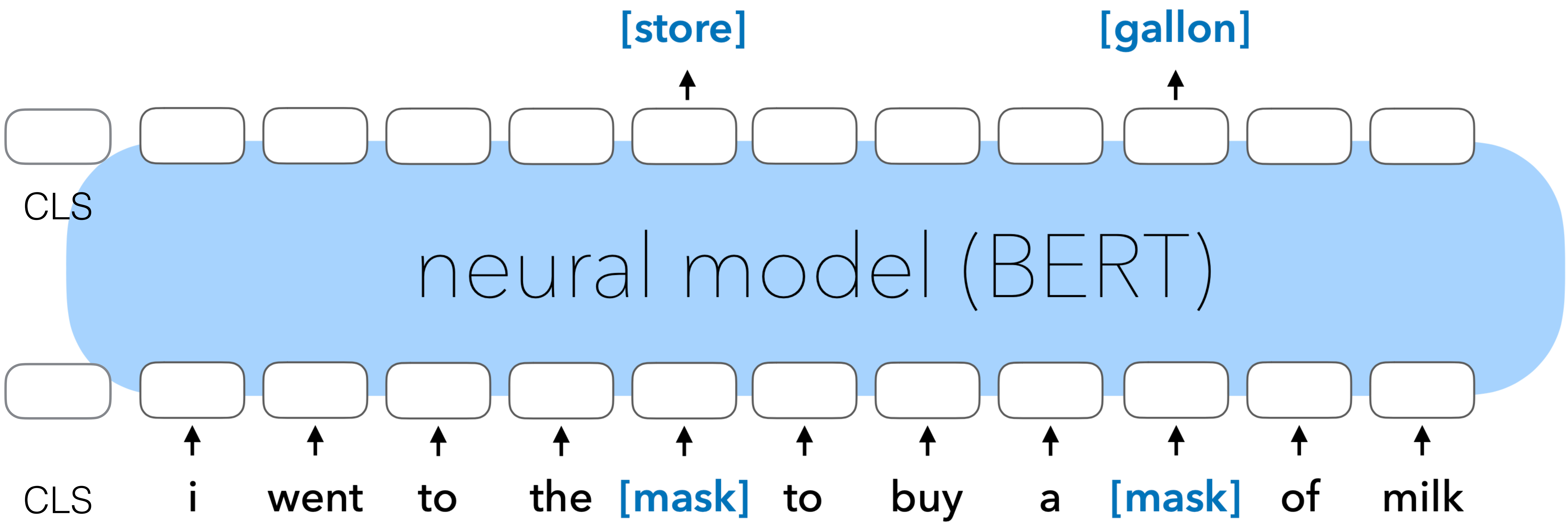
E.g., RNN



we could use the model
to learn better word embeddings
but this is comp. expensive

Masked Language Models (MLMs)

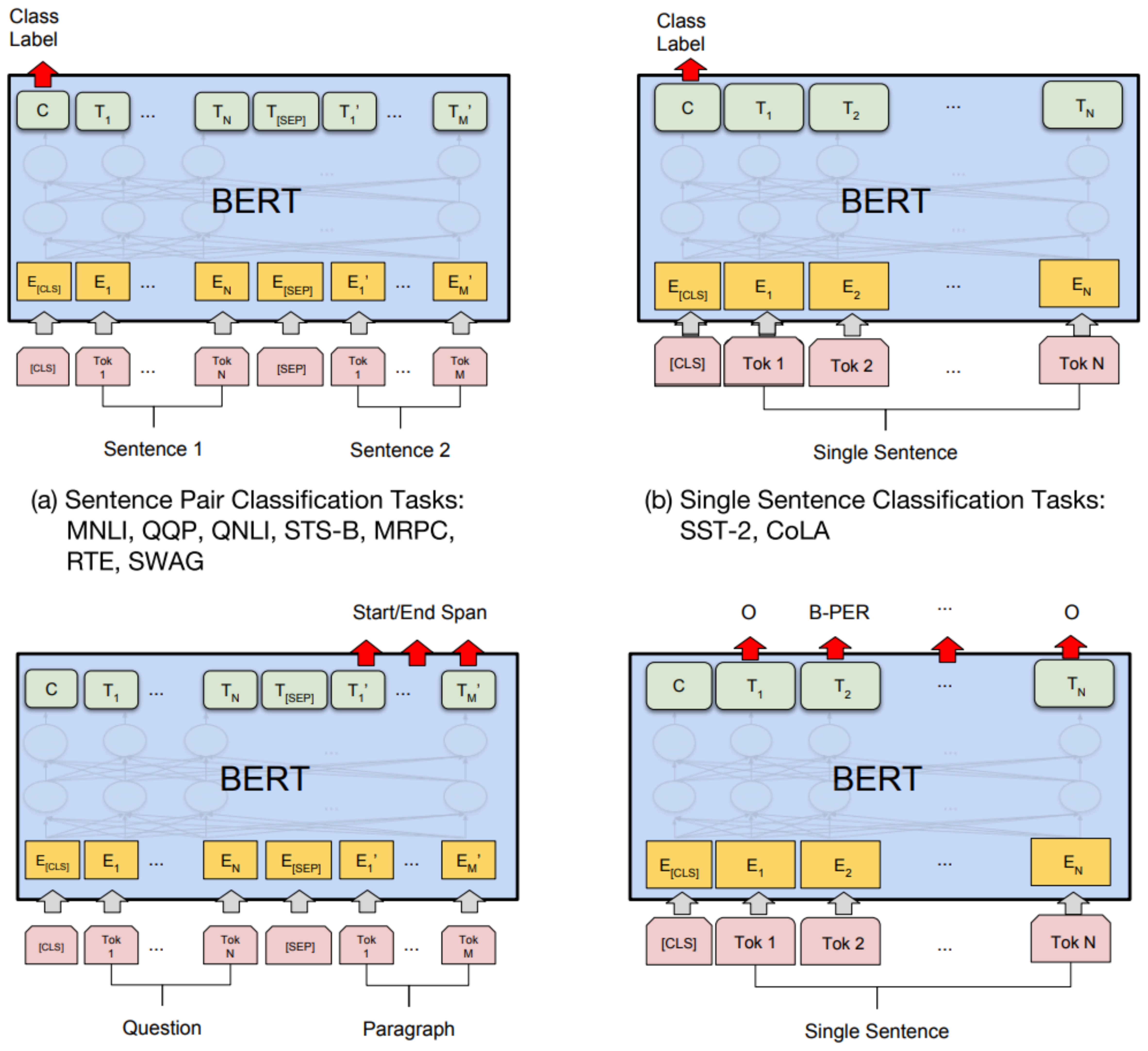
- We can estimate a complex transformer model from large text corpora by setting up a simple self-supervised masking task (masked language model)
- The resulting representation* can substantially help many follow-on tasks
- Some parts, e.g., the output prediction heads, can be stripped before reusing or adapting simple new prediction heads; the rest can be also fine-tuned



Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019

MLM (Bert) fine tuning tasks

- We can fine-tune a BERT model to a specific downstream task by adopting different prediction heads and input/output arrangements
- E.g.,
  - sentiment classification
  - pair sentence classification
  - sentence tagging
  - question-answering
  - Etc.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

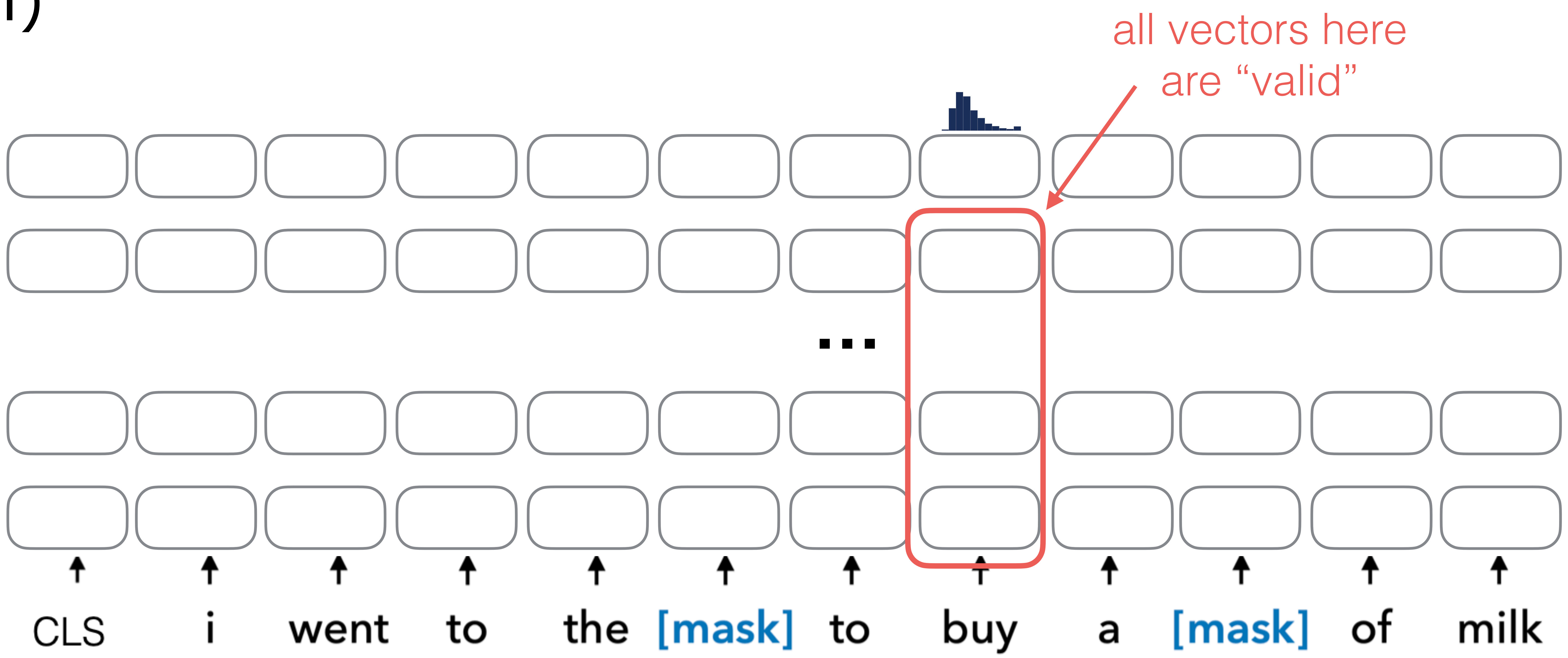
(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019

Contextual word vectors from MLM

- We can extract a valid contextually guided word vector from any BERT unit at any layer except the last one (in the original Bert model these vectors were 768 dimensional)



- Note that the input/output layer uses an automatic "word pieces" tokenizer to handle large vocabularies. E.g., you could see 'Teaching' → ['_Teach', 'ing']