

Natural Language Processing

自然語言處理

黃瀚萱

Department of Computer Science

National Chengchi University

2020 Fall

Lesson 9

Word Embeddings

Schedule

Date	Topic
9/16	Introduction
9/23	Linguistic Essentials
9/30	Collocation
10/7	Language Model
10/14	Performance Evaluation and Word Sense Disambiguation
10/21	Text Classification (HW1 will be assigned)
10/28	Invited Talk: NLP and Cybersecurity (Term Project)
11/4	POS Tagging
11/11	Midterm Exam

Schedule

Date	Topic
11/18	Chinese Word Segmentation
11/25	Word Embeddings
12/2	Neural Networks for NLP
12/9	Parsing
12/16	Discourse Analysis
12/23	Invited Talk
12/30	Final Project Presentation I
1/6	Final Project Presentation II
1/13	Final Exam

Agenda

- Word embeddings
 - CBOW
 - Skipgram
- Analogical reasoning
- Cross-lingual word embeddings
- Applications of word embeddings

One Hot Representation

- Bag-of-words (aka “one hot” representation)
- Treat each word as an individual symbol
- Dimensionality of one-hot vector is same as number of distinct features.
- Features are completely independent from one another.
 - No difference between the concept pair “dog” vs “thinking” and “dog” vs “cat”

1	2	3	4	5	6	7	8	n
dog				cat			thinking				

Drawback of the Bag-of-Word Scheme

- The feature vectors with the bag-of-word scheme are usually sparse.
- The length of the vector is proportional to the vocabulary size and tends to be very large.
- The feature vectors are sparse.
 - Most of the dimensions are zero.

Dense Vectors for Word Representation

- Aka “distributed word representation”
- The word embedding model is learned to represent each word in a dense vector.
 - With a smaller dimension d from 50 to 1,000.
 - Most elements in the feature vector are not zero.
 - Similar words share similar representations

dog	0.53	0.15	0.12	0.78	0.65	0.15	0.44	0.34	0.89
cat	0.43	0.11	0.22	0.82	0.69	0.09	0.49	0.31	0.95
thinking	0.44	0.89	0.65	0.23	0.88	0.13	0.98	0.61	0.45

Advantages of Word Embeddings

- Short feature vectors are easier for machine learning model to train.
- Word embeddings is better at generalization.
- Some semantic properties are captured by word embeddings.
 - Semantic similarity
 - Analogical reasoning

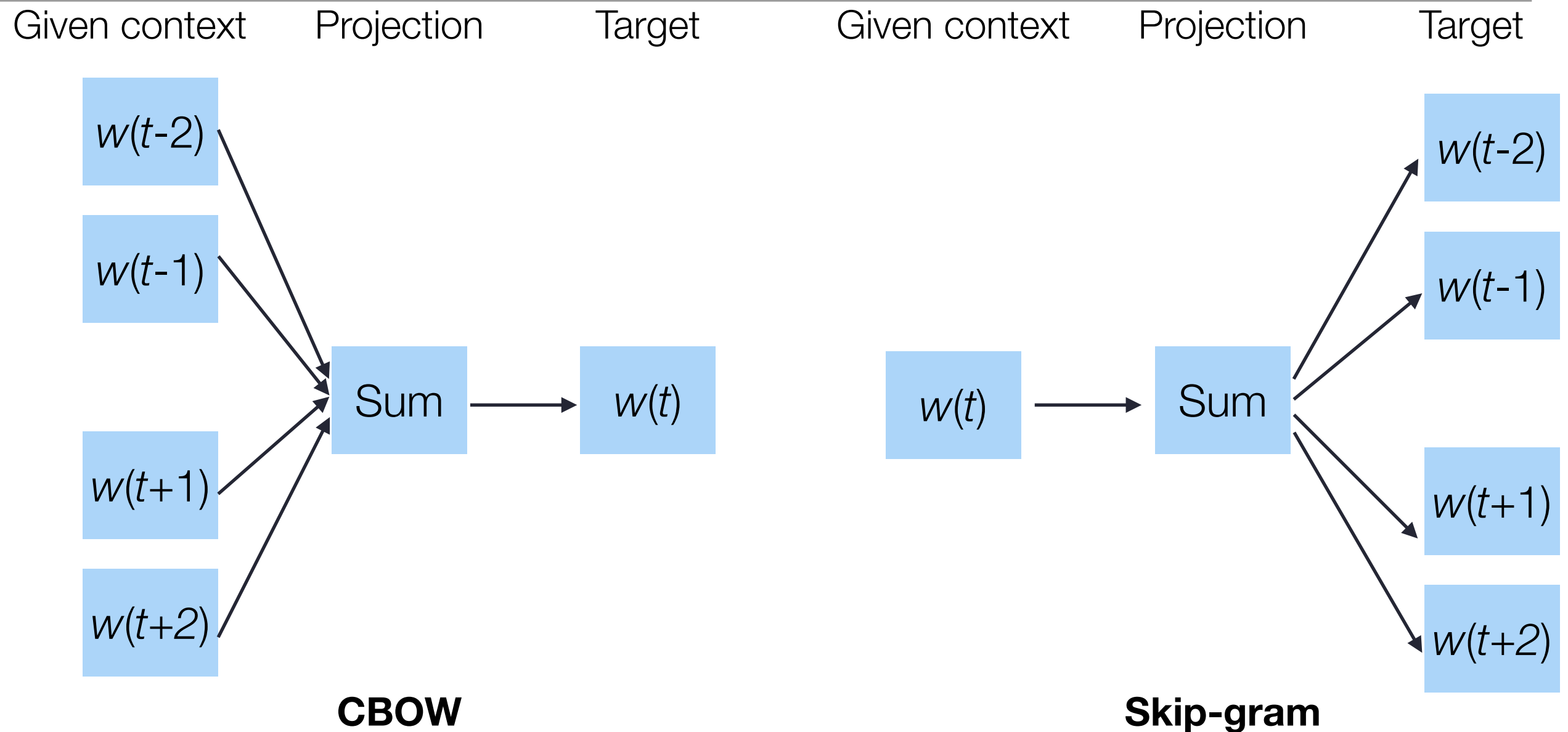
Pre-trained Word Embeddings Publicly Available

- Word2Vec (Mikolov et al.)
 - CBOW
 - Skipgram
 - <https://code.google.com/archive/p/word2vec/>
- GloVe (Pennington et al.)
 - <https://nlp.stanford.edu/projects/glove/>
- Fasttext (Bojanowski et al.)
 - <https://fasttext.cc>

Word2Vec

- The most popular model for word embedding
- Two variations
 - Continuous Bag of Word (CBOW)
 - Skip-gram

CBOW vs Skip-gram

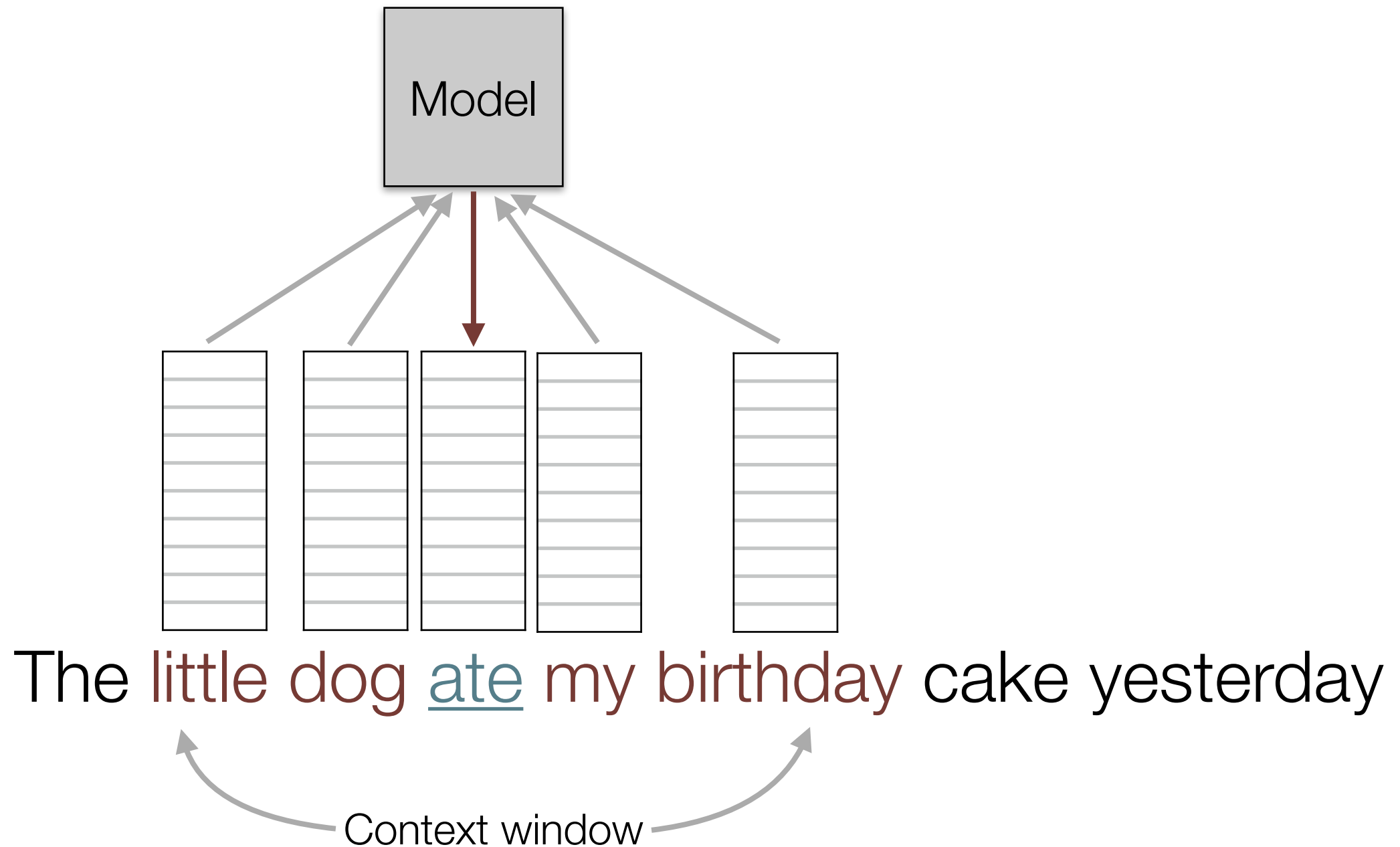


- CBOW predicts the current word based on the context.
- Skip-gram predicts surrounding words given the current word.

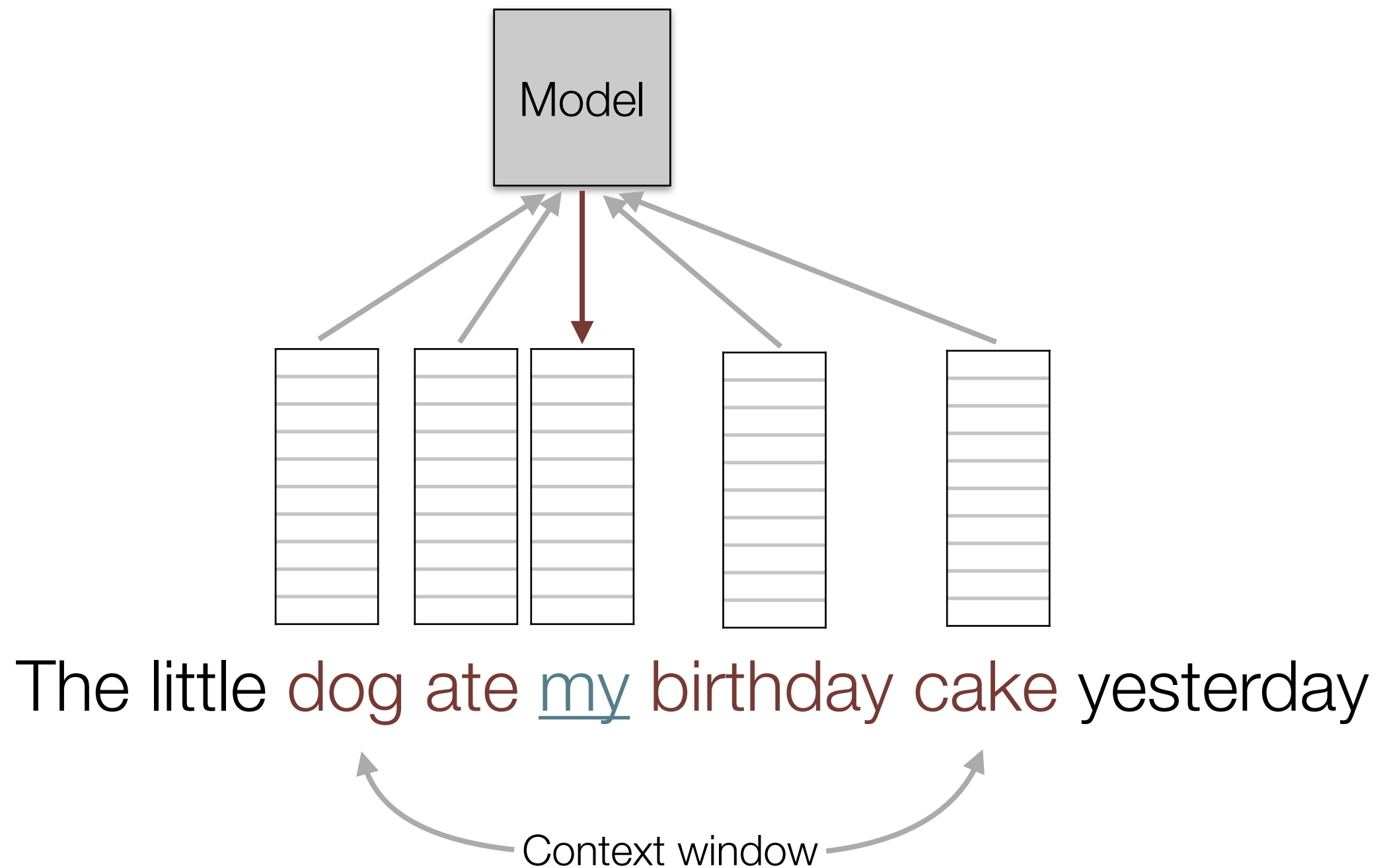
Unsupervised Training

- Basic idea of word2vec
 - Represent each word as a vector of the dimension in 50 to 1,000.
 - Train a classifier to predict the word given the contextual words.

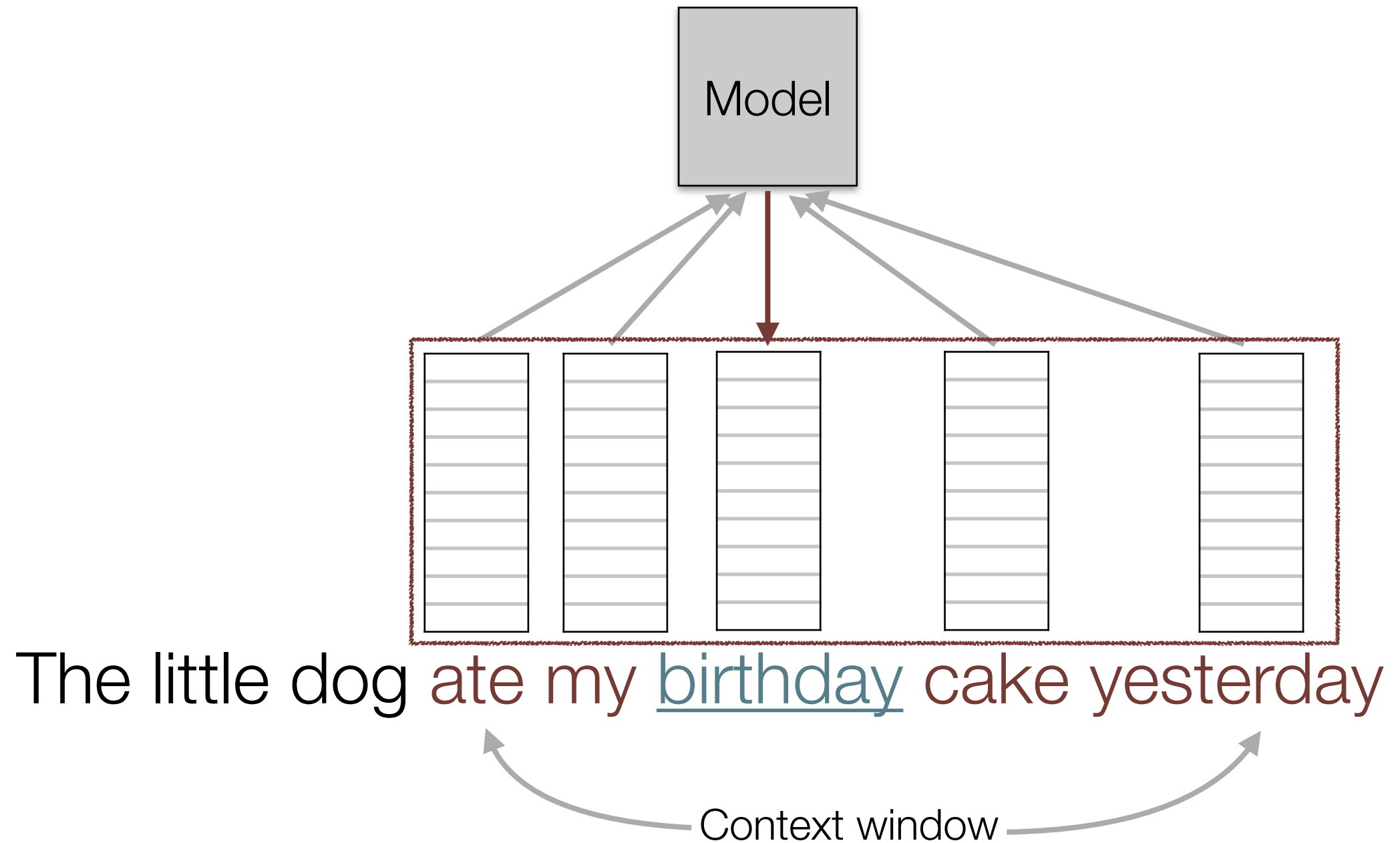
CBOW



CBOW



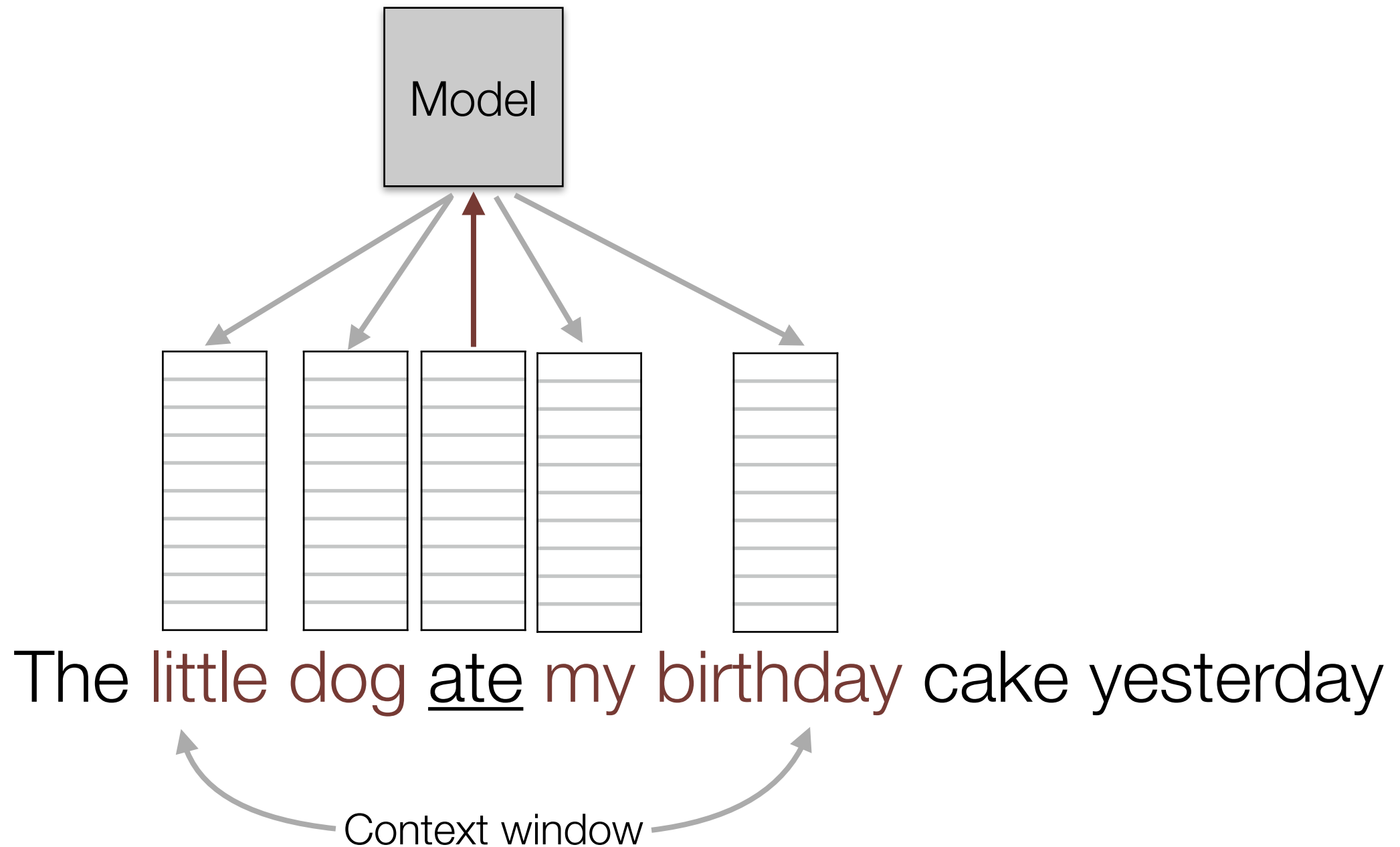
CBOW



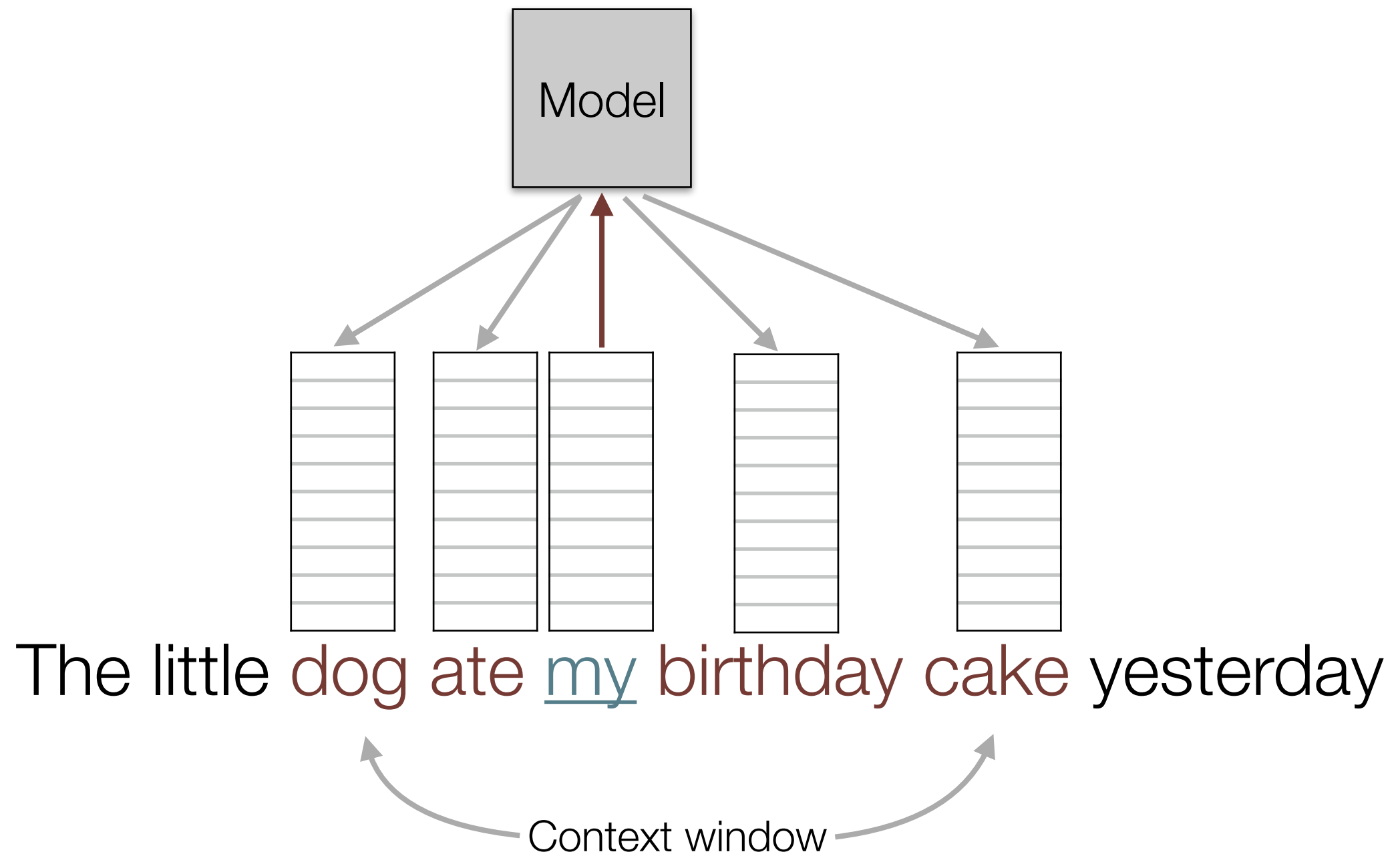
Skip-gram

- Basic idea of word2vec
 - Represent each word as a vector of the dimension in 50 to 1,000.
 - Train a classifier to predict the contextual words given a target word

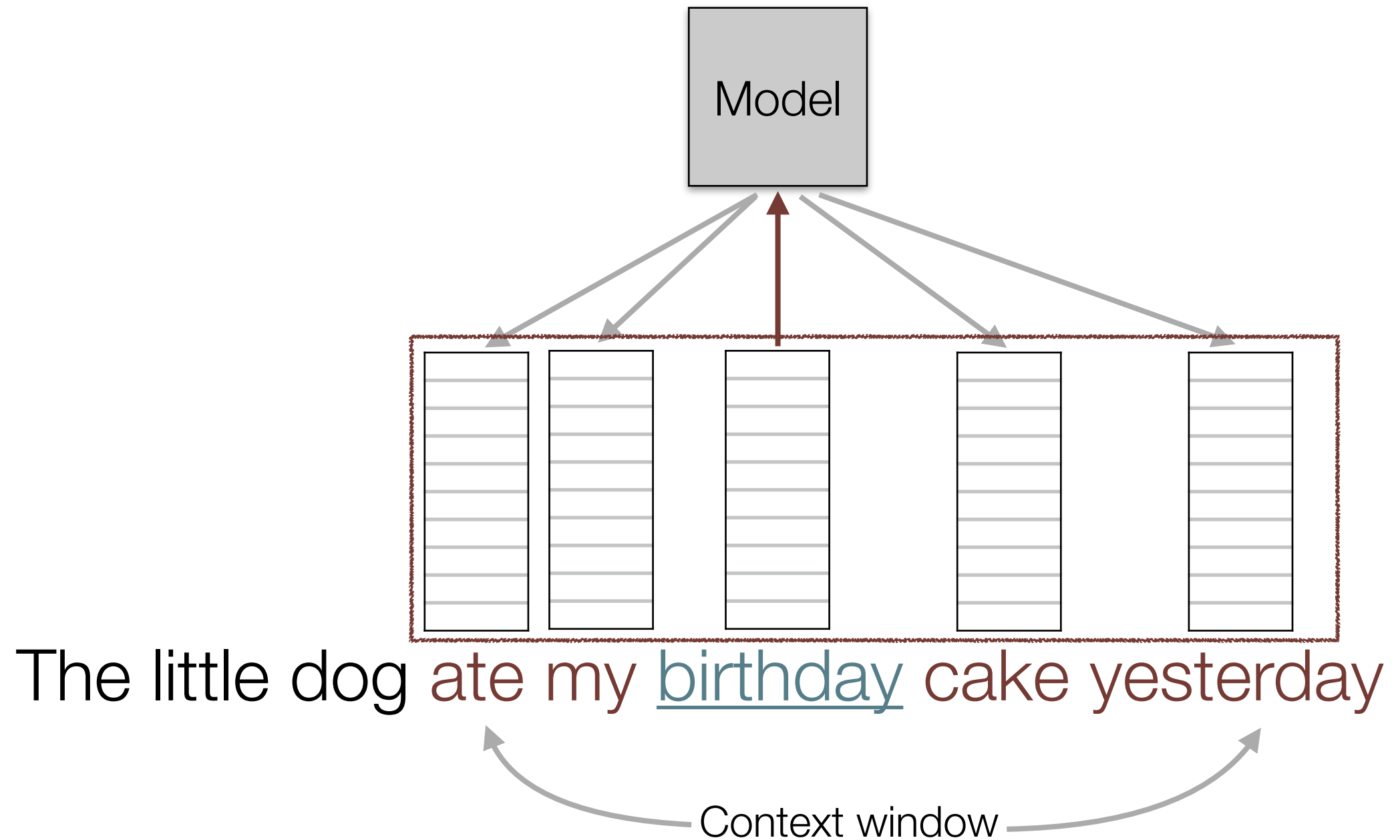
Skip-gram



Skip-gram



Skip-gram



Objective Function of Skip-gram

- Word embeddings (or distributed word representations) are trained to predict well words that appear in its context.
- Given a set of sentences w_1, \dots, w_T , the objective of the skip-gram model is to maximize the log-likelihood:

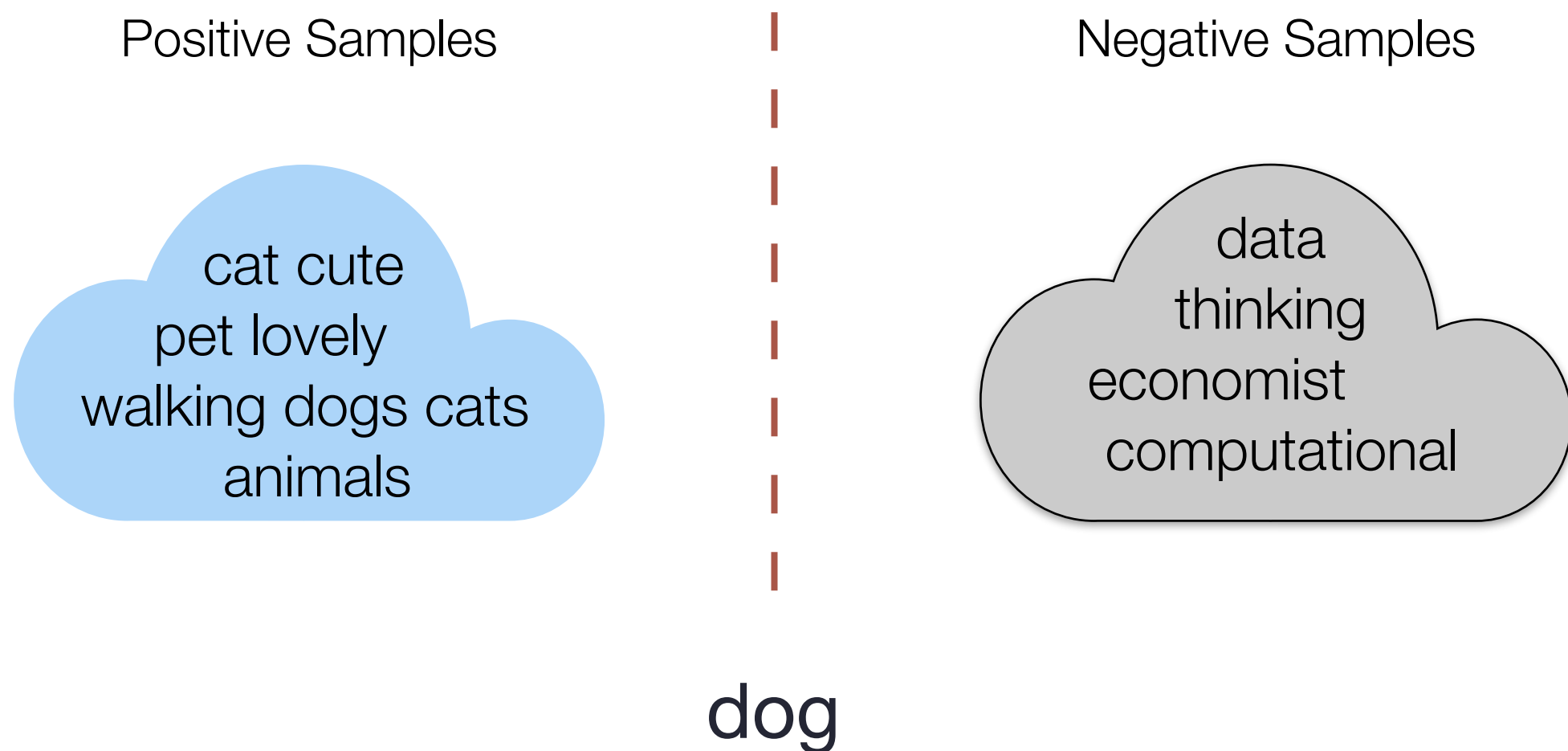
$$\sum_{t=1}^T \sum_{c \in \mathcal{C}_t} \log p(w_c \mid w_t)$$

- With a scoring function s maps pairs of a target word and a contextual word to a real number.

$$p(w_c \mid w_t) = \frac{e^{s(w_t, w_c)}}{\sum_{j=1}^W e^{s(w_t, j)}}$$

Negative Sampling

- The model learns from not only related words (positive samples), but also unrelated words (negative samples) for knowing the boundary better.



Negative Sampling

- For the target word at position t we consider all contextual words as positive samples and randomly select negative samples from the vocabulary.
- The objective with negative sampling can be re-written as:

The diagram illustrates the negative sampling objective function. It features a mathematical expression with two main components: a sum over positive samples and a sum over negative samples. Annotations include arrows pointing from descriptive text to each part of the equation and a box highlighting the negative sampling term.

$$\sum_{t=1}^T \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right]$$

logistic loss function

Original loss for positive samples

Loss for negative samples

Property of Word Embeddings

- Each word is represented in a vector with a dimension in between 50 and 1000 in a dense space.
- Similarity
 - Similar or related words are close in the vector space.
- Regularity
 - Rome : Italy = Paris : ?
 - $\text{vec}(\text{Rome}) - \text{vec}(\text{Italy}) + \text{vec}(\text{France}) \sim \text{vec}(\text{Paris})$

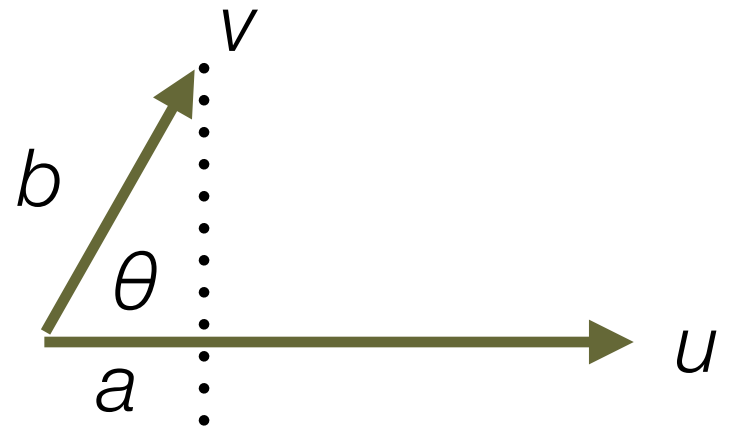
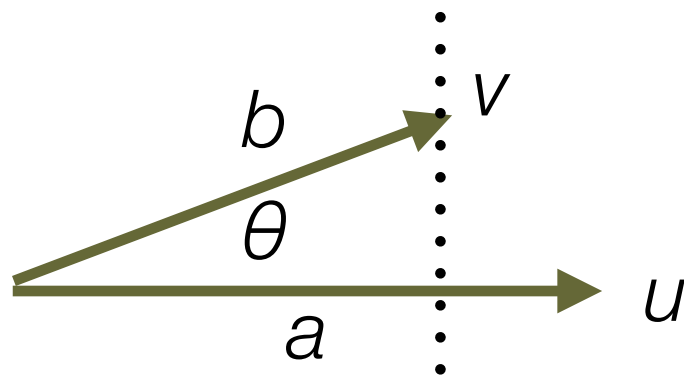
Semantic Similarity of Word Embeddings

- Similar words in the dense space share similar word embeddings
 - $\text{Similarity}(\text{dog}, \text{cat}) > \text{Similarity}(\text{dog}, \text{thinking})$
 - $\text{Distance}(\text{dog}, \text{cat}) < \text{Distance}(\text{dog}, \text{thinking})$
- How to calculate the similarity?
 - Cosine similarity

Cosine Similarity (Recap)

- The cosine of the angle θ between two vectors.
 - $\cos(0^\circ) = 1$
 - $\cos(90^\circ) = 0$
 - $\cos(180^\circ) = -1$
- Similar vectors have a smaller θ and a greater $\cos(\theta)$

$$\cos(\theta) = \frac{a}{b}$$



Cosine Similarity Calculation (Recap)

- In a vector space with m dimensions, the cosine similarity of two vectors v and u can be calculated as follows.

$$\text{similarity} = \cos(\theta) = \frac{\vec{v} \cdot \vec{u}}{|\vec{v}| |\vec{u}|}$$

← Overlapped items in two vectors
← Normalized by their lengths

$$\vec{v} \cdot \vec{u} = \sum_{i=1}^m v_i u_i \quad |\vec{v}| = \sqrt{\sum_{i=1}^m v_i^2}$$

Dot product (内積)
of the two vectors

Length of a vector

Cosine Similarity in Python

```
def cosine_similarity(v, u):  
    sum = 0.0  
    v_len = 0.0  
    u_len = 0.0  
    for v_, u_ in zip(v, u):  
        sum += v_ * u_  
        v_len += v_ ** 2  
        u_len += u_ ** 2  
    return sum / ((v_len ** 0.5) * (u_len ** 0.5))
```

$$\text{similarity} = \cos(\theta) = \frac{\vec{v} \cdot \vec{u}}{|\vec{v}| |\vec{u}|}$$
$$\vec{v} \cdot \vec{u} = \sum_{i=1}^m v_i u_i \quad |\vec{v}| = \sqrt{\sum_{i=1}^m v_i^2}$$

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
-0.5 = cosine_similarity([[1, 0, -1]], [[-1, -1, 0]])[0][0]
```

The WordSimilarity-353 Test Collection

- One of the most popular benchmark for testing word embeddings.
- Publicly available: <http://alfonseca.org/eng/research/wordsim353.html>
- Similarity \neq Relatedness
 - Pencil and pen is similar
 - Cloth and closet is related but not similar

Similar Word Pair in WS-353

Word 1	Word 2	Score
tiger	tiger	10
fuck	sex	9.44
midday	noon	9.29
journey	voyage	9.29
dollar	buck	9.22
money	cash	9.15
coast	shore	9.1
money	currency	9.04
football	soccer	9.03
magician	wizard	9.02

Word 1	Word 2	Score
king	cabbage	0.23
professor	cucumber	0.31
noon	string	0.54
chord	smile	0.54
rooster	voyage	0.62
sugar	approach	0.88
stock	life	0.92
stock	jaguar	0.92
monk	slave	0.92
lad	wizard	0.92

Related Word Pair in WS-353

Word 1	Word 2	Score
environment	ecology	8.81
Maradona	football	8.62
OPEC	oil	8.59
computer	software	8.5
money	bank	8.5
Jerusalem	Israel	8.46
law	lawyer	8.38
weather	forecast	8.34
FBI	investigation	8.31
nature	environment	8.31

Word 1	Word 2	Score
king	cabbage	0.23
professor	cucumber	0.31
noon	string	0.54
chord	smile	0.54
rooster	voyage	0.62
sugar	approach	0.88
stock	life	0.92
stock	jaguar	0.92
monk	slave	0.92
lad	wizard	0.92

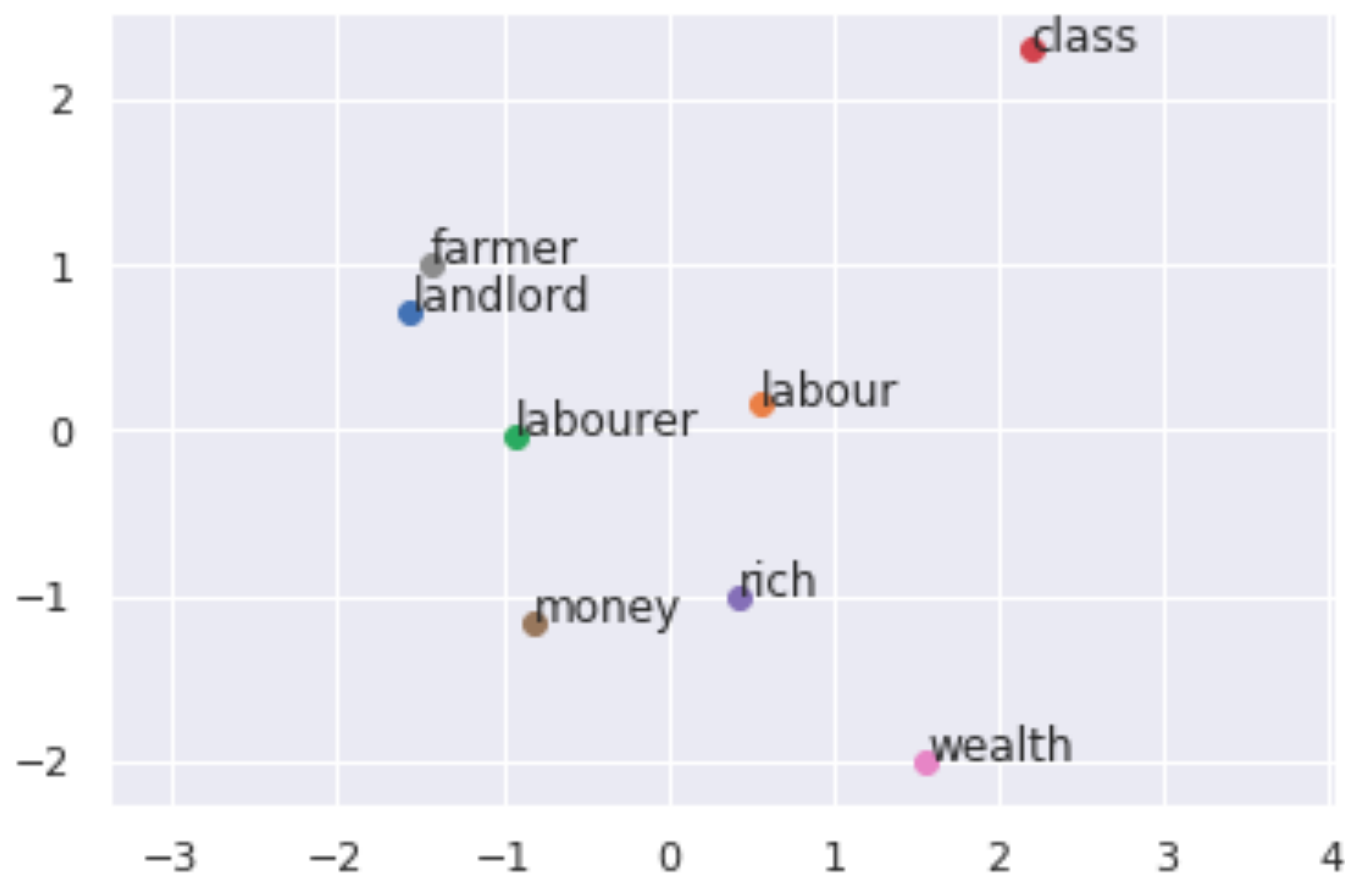
Simlex-999

- Another popular benchmark that focuses on similarity only.
- Antonyms are considered related in WS-353, but antonym pairs are scored the lowest in Simlex-999.

Pair	Simlex-999 rating	WordSim-353 rating
(coast, shore)	9	9.1
(clothes, closet)	1.96	8

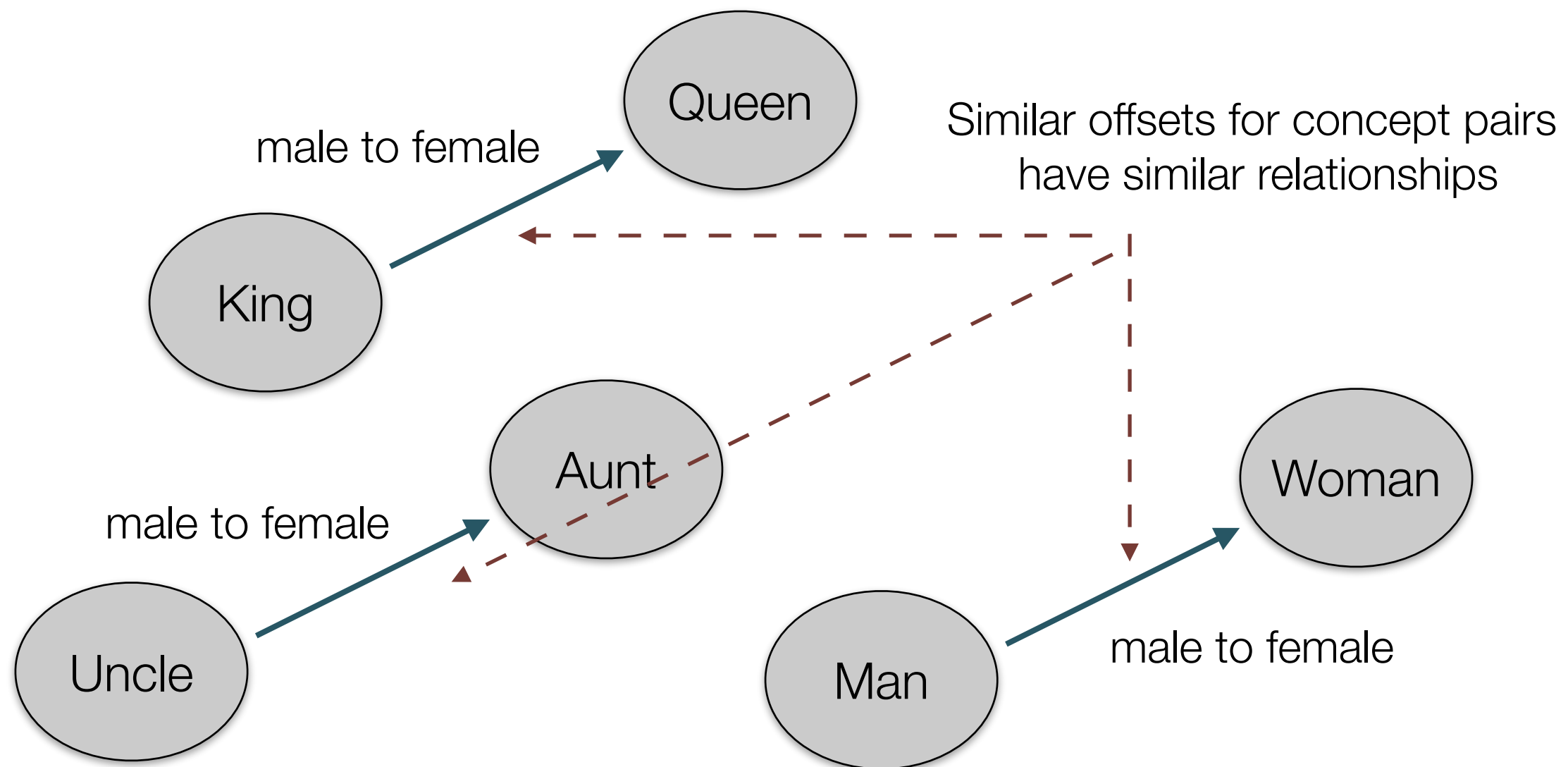
Visualization of Word Embeddings

- Project each word in the word embedding space to a low (2 or 3, typically) dimensional space.
 - Principal component analysis, PCA
 - Scatter plot shows the relations of the words.



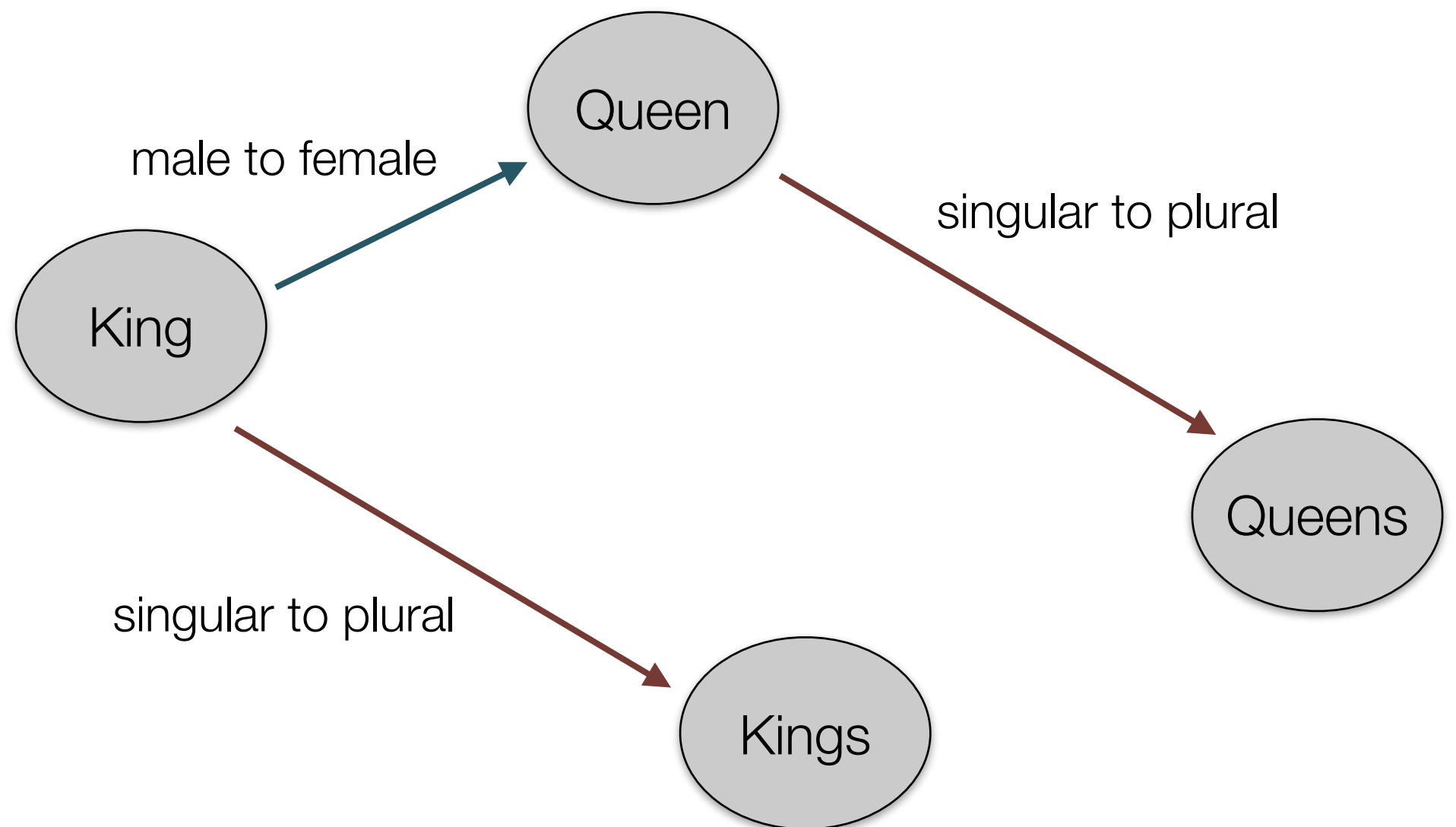
Linguistic Regularities

- Linguistic regularities in word embeddings are able to capture the relationship between concepts.



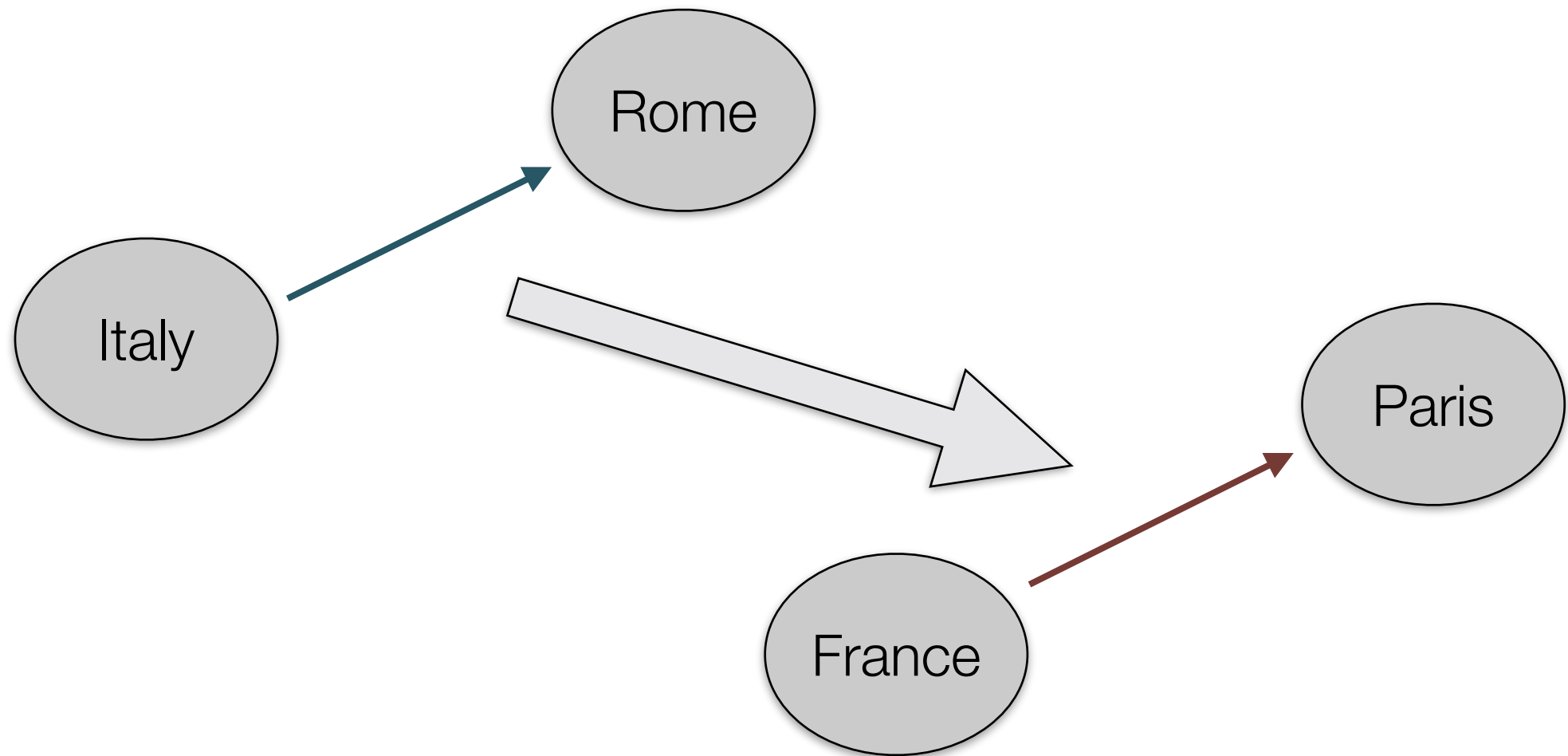
Linguistic Regularities in Cascade

- Multiple relations are embedded in a single word



Analogueical Reasoning

- The properties of linguistic regularities can be applied in analogueical reasoning
- Italy : Rome = France : ? Rome - Italy + France = Paris

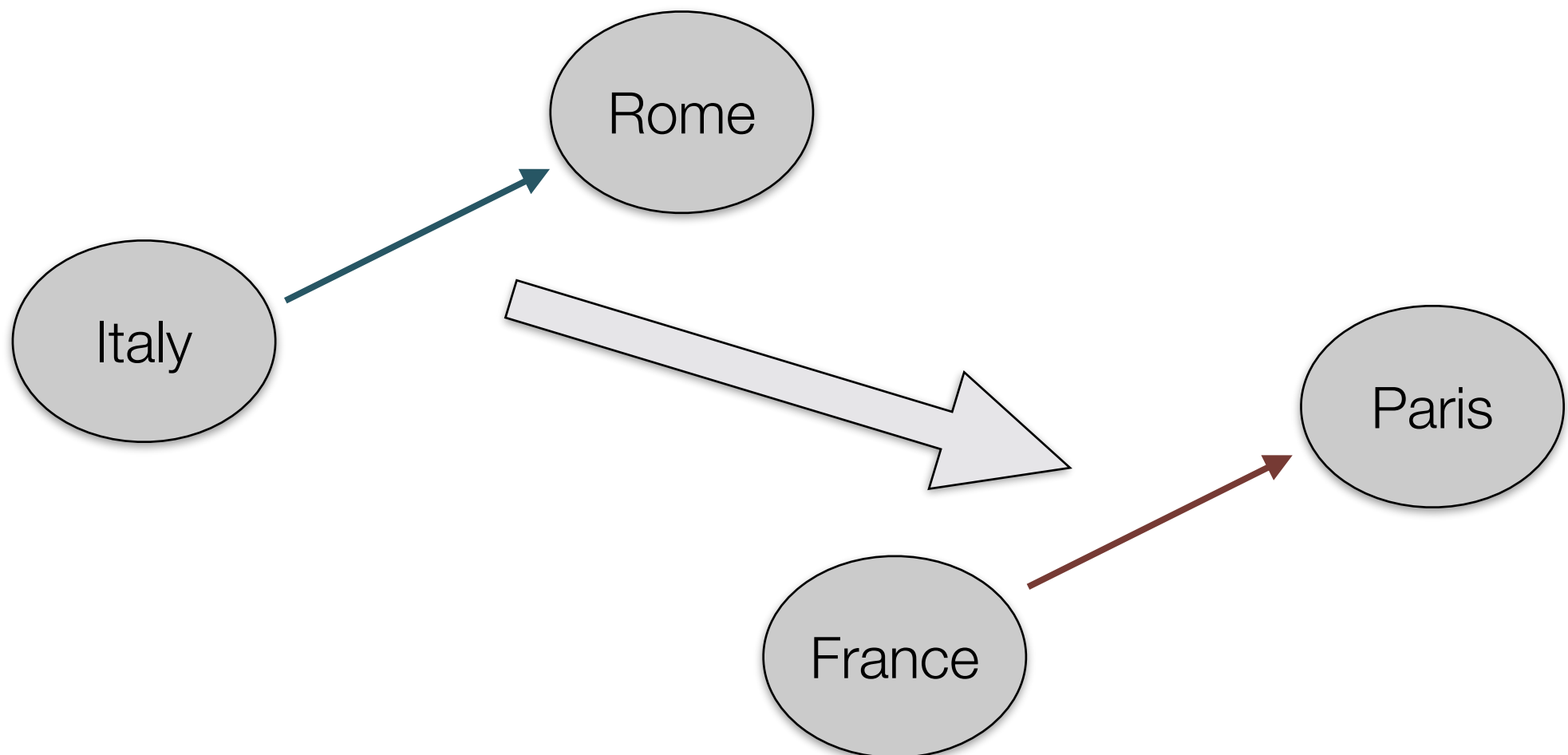


Vector Arithmetic for Analogical Reasoning

- Now, concepts (words) are something computable.

- $\text{Rome} - \text{Italy} + \text{France} = \text{Paris}$

Not really arithmetic addition, subtraction, and equations



3CosAdd (Mikolov et al., 2013)

- Find the b^* with the highest similarity in the cosine similarity of b^* and $b - a + a^*$
- $a:a^* = b:b^*$ Rome : Italy = Paris : ?

$$\text{3CosADD} \quad \operatorname{argmax}_{b^* \in V} \cos(b^*, b - a + a^*)$$

3CosMul (Levy et al, 2014)

- Find the b^* with the highest similarity in 3 cosine multiplication
- $a:a^* = b:b^*$ Rome : Italy = Paris : ?

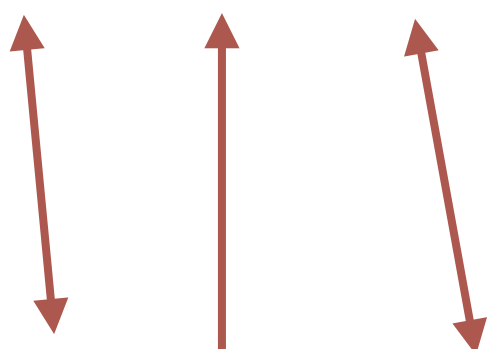
$$\arg \max_{b^* \in V} \frac{\cos(b^*, b) \cos(b^*, a^*)}{\cos(b^*, a) + \varepsilon}$$

($\varepsilon = 0.001$ is used to prevent division by zero)

3CosAdd vs 3CosMul

- 3COSADD allows one sufficiently large term to dominate the expression
- 3COSMUL achieves a better balance amplifying the small differences between terms and reducing the larger ones

$$\text{3CosADD } \operatorname{argmax}_{b^* \in V} \cos(b^*, b - a + a^*)$$

$$\text{3CosMUL } \operatorname{argmax}_{b^* \in V} \frac{\cos(b^*, b) \cos(b^*, a^*)}{\cos(b^*, a) + \epsilon}$$


Stereotype Inherent in Word Embeddings

- Some bias (偏見) ideas and stereotypes (刻板印象) are also learned in word embeddings, according to the training corpus.
- Man : Woman = King : Queen
- Man : Computer Programmer = Woman : Homemaker

Stereotype in Gender

most similar occupations of “her”

**homemaker
nurse receptionist
librarian socialite
hairstresser nanny bookkeeper
stylist housekeep interior designer
guidance counselor**

most similar occupations of “he”

**maestro
skipper protege
philosopher captain
architect financier warrior
broadcaster magician fighter pilot
boss**

Gender Stereotype from she : he Analogies

sewing : carpentry	register nurse : physician
nurse : surgeon	interior designer : architect
blond : burly	feminism : conservatism
giggle : chuckle	vocalist : guitarist
sassy : snappy	diva : superstar
volleyball : football	cupcakes : pizzas
housewife : shopkeeper	softball : baseball
cosmetics : pharmaceuticals	petite : lanky
charming : affable	hairstylist : barber

FastText

- Also consider the subword information.
- Taking the n-grams character sequences of a word as additional contextual information.
 - The word “algorithm” with $n=3$
 - al, alg, lgo, gor, ori, rit, ith, thm, hm
- Especially useful for Chinese
 - Each hanzi (漢字) is a subword

FastText

- Pre-trained word embeddings are available for 157 languages.
- <https://fasttext.cc/docs/en/crawl-vectors.html>
- A general text classifier based on the FastText embeddings is also attached.

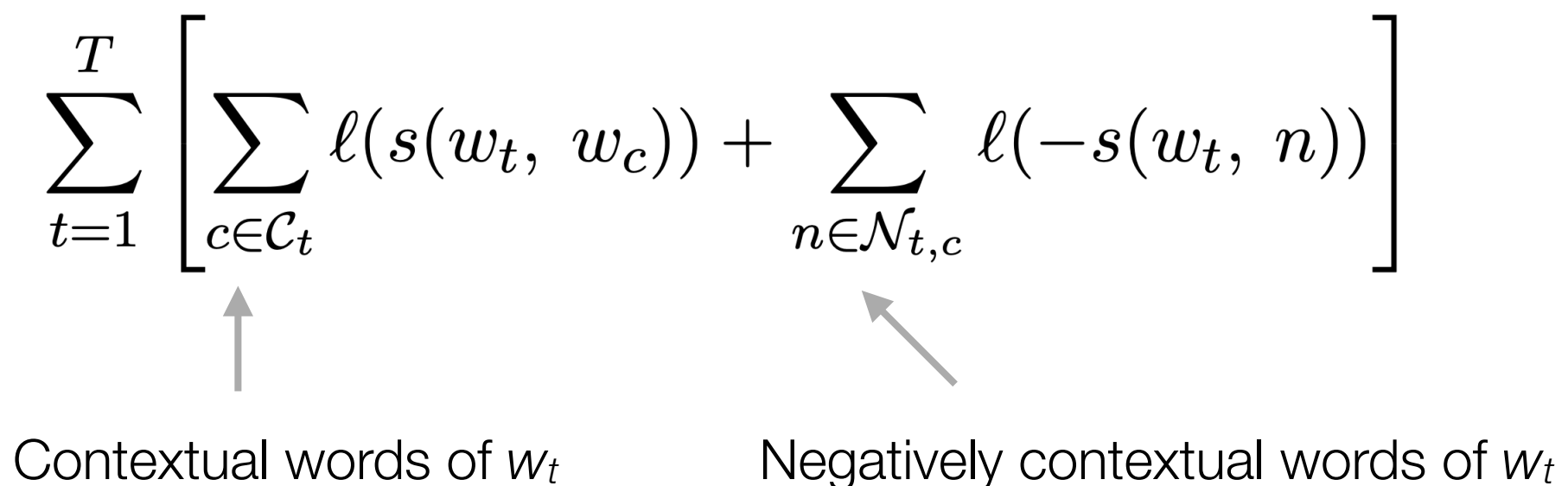
Bilingual/Multilingual Word Representations

- Aimed at projecting words from multiple languages into a universal, single vector space.
- Words, from different languages, sharing a similar meaning will be close in the vector space.
 - $\text{cosine}(\text{狗}, \text{dog}) > \text{cosine}(\text{dog}, \text{cat})$
- Very useful for cross-lingual tasks
- Acting like a translator who masters multiple languages.

Bilingual Word Embeddings with Shuffling

- Training Skip-gram with a parallel corpus (Vulic & Moens, 2015).
- Redefine the contextual words of Skip-gram by adding words from the translated counterpart.

$$\sum_{t=1}^T \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right]$$



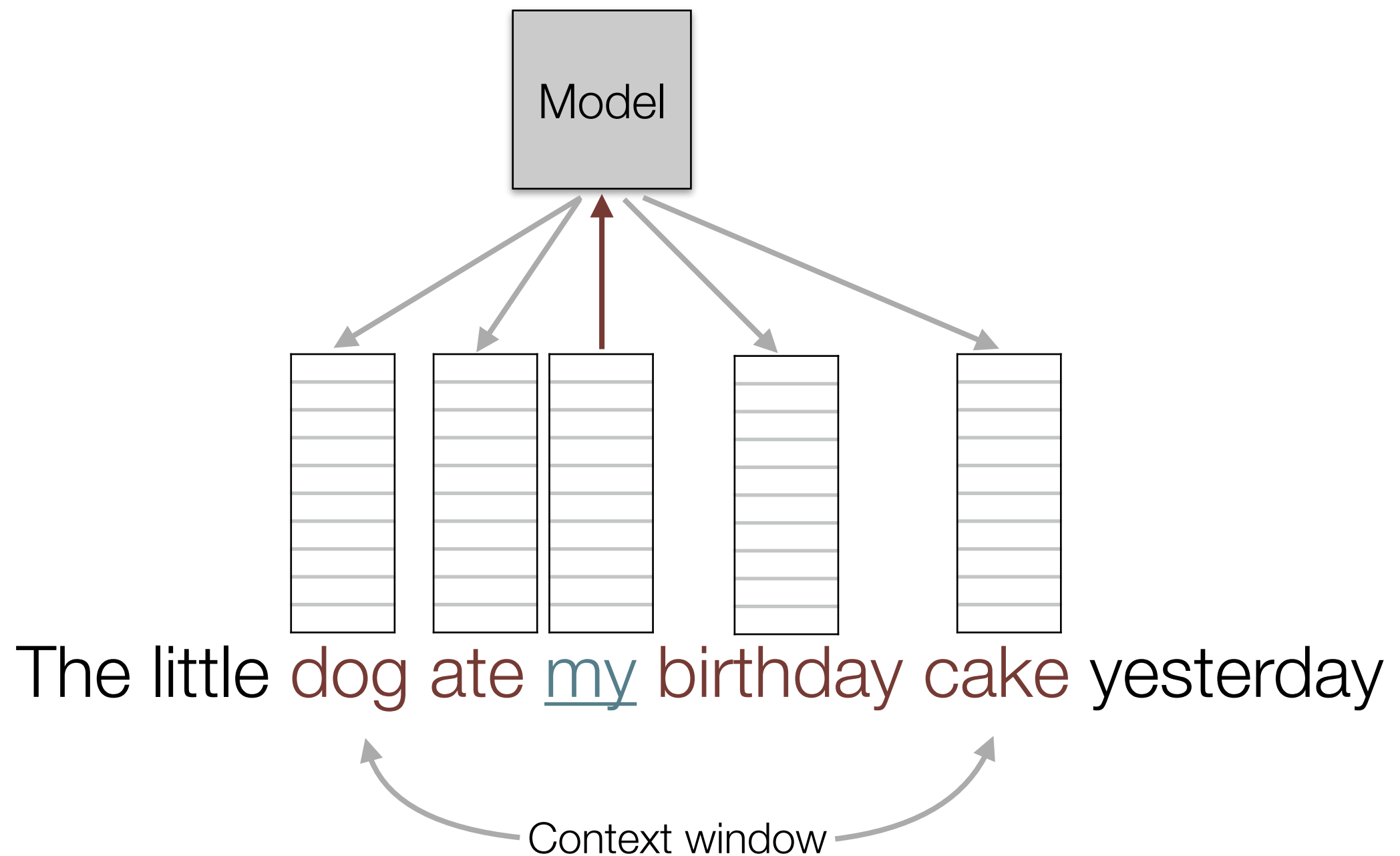
Contextual words of w_t Negatively contextual words of w_t

Parallel Corpus

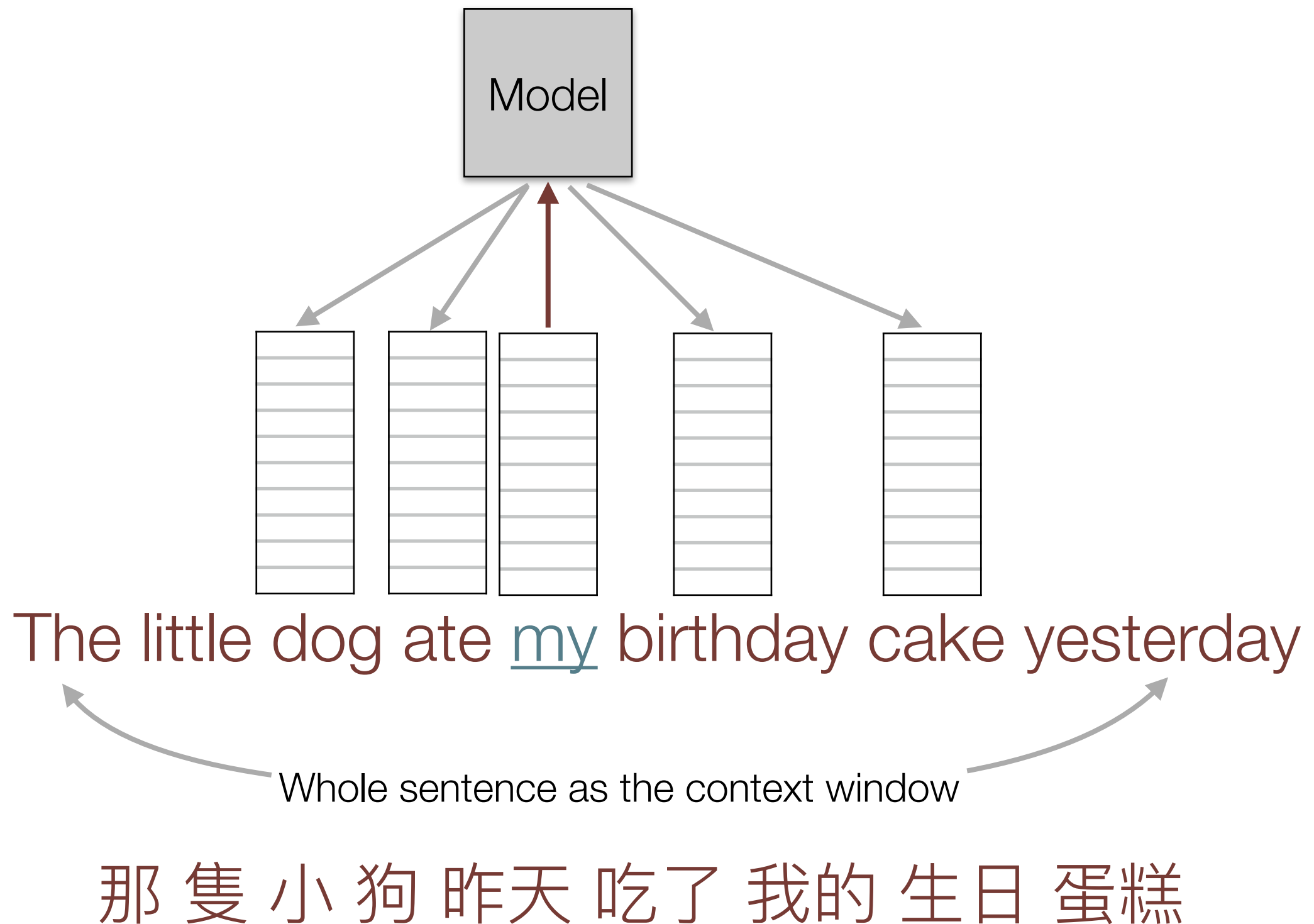
- A corpus consists of translation pairs from two languages.
- Usually aligned at the sentence level.
- Mandatory resource for studying machine translation.

Chinese	English
今天下大雨	It is heavy rain today.
昨天發布新的產品	New products were released yesterday.
有人對命案指出新的證據	Someone pointed out new evidence to the murder case.
昨天遊行超過一萬人	More than 10,000 people marched yesterday.
...	...

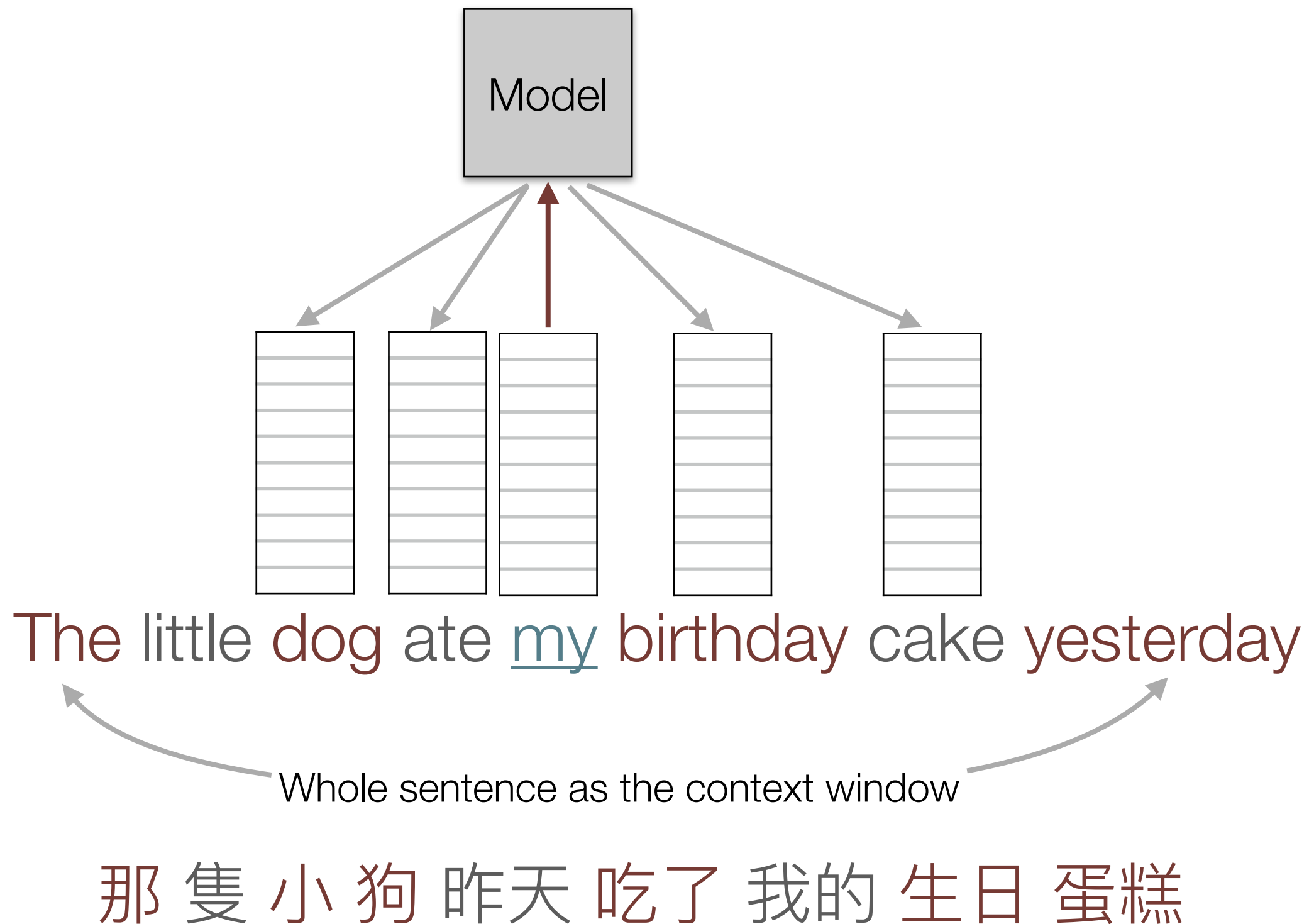
Original Skip-gram



Considering Bilingual Contextual Words




Shuffle to Limit Contextual Words



Bilingual Word Embeddings with Shuffling

- Based on the assumption of Big Data, highly related words (both monolingual and cross-lingual) will have a higher chance to be selected in the contextual window.

$$\sum_{t=1}^T \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right]$$

$$\sum_{t=1}^T \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) \right]$$


Randomly select words from the sentence w_t appears and from the counterpart sentence, in the other language.

Bilingual Word Embeddings with Shuffling

- Based on the assumption of Big Data, highly related words (both monolingual and cross-lingual) will have a higher chance to be selected in the contextual window.

$$\sum_{t=1}^T \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right]$$

$$\sum_{t=1}^T \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) \right]$$

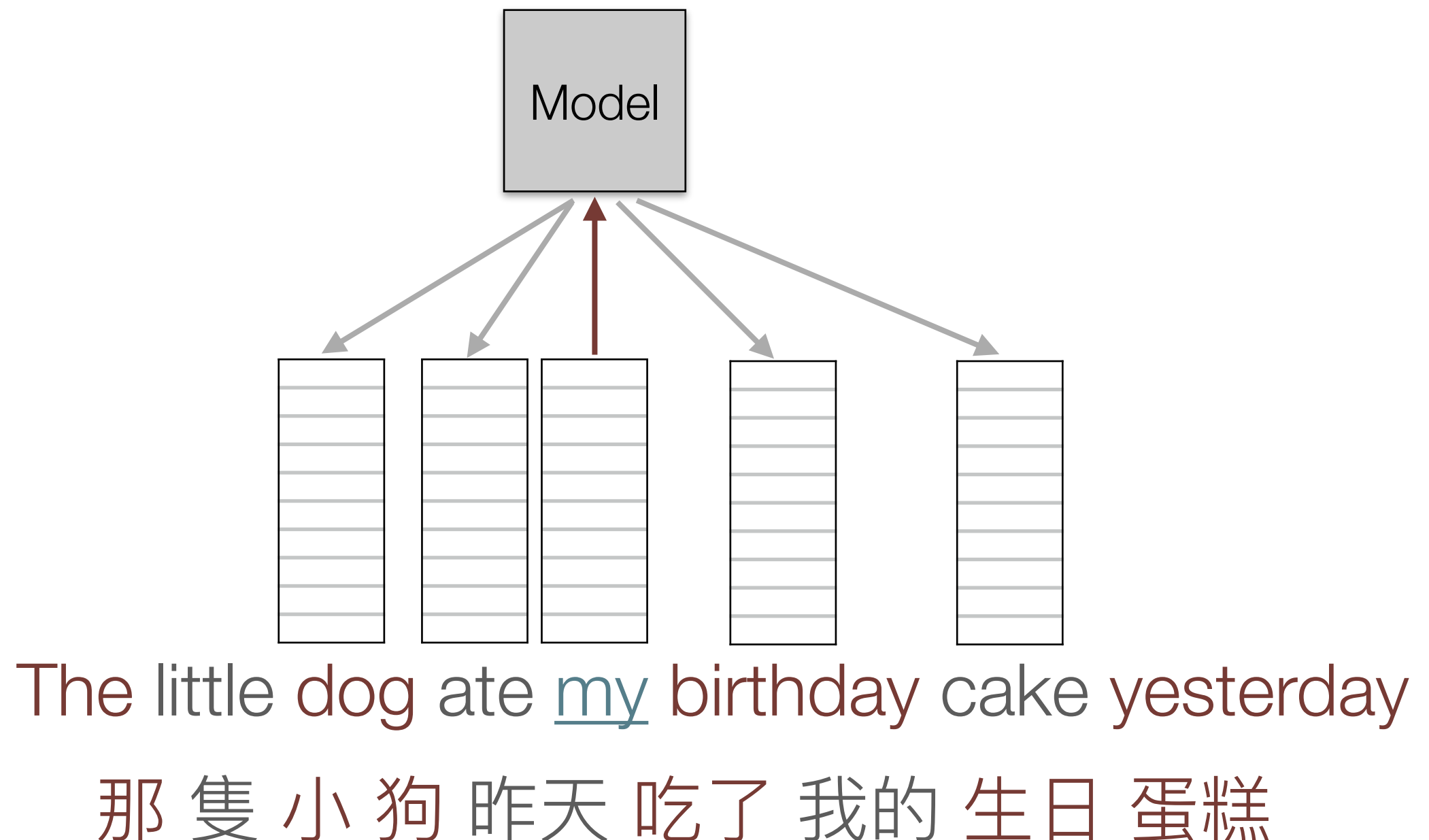
$$\mathcal{C}_t = \{\mathcal{W} \subseteq \mathcal{S}_t\} \cup \{\mathcal{W}' \subseteq \mathcal{S}'_t\}$$

\mathcal{S}_t : The sentence w_t appears

\mathcal{S}'_t : The counterpart sentence of \mathcal{S}_t in the parallel corpus

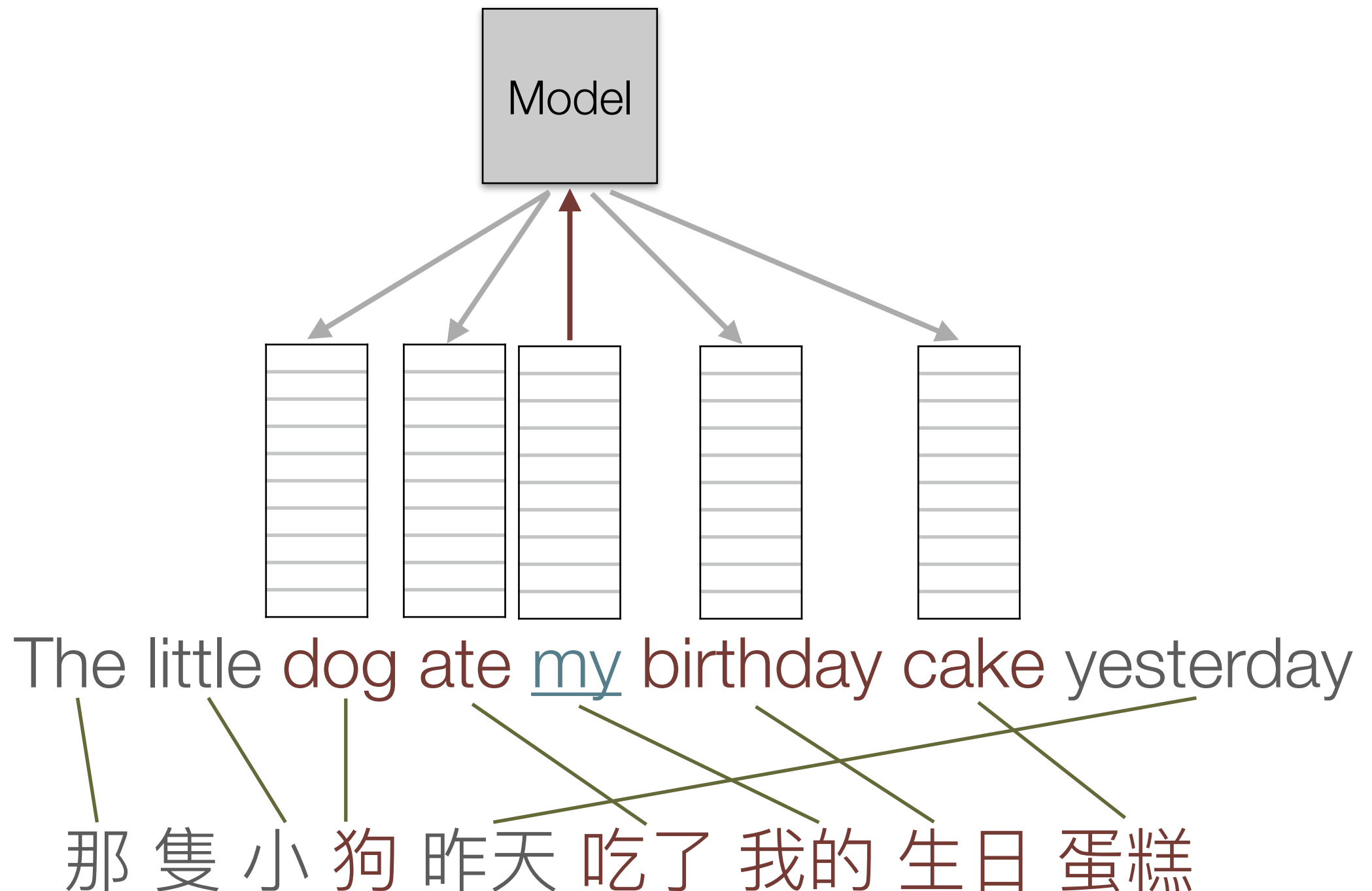
Drawback of the Shuffle Method

- However, randomly selecting words within a large window introduces many noises.



Bilingual Word Embeddings with Word Alignment

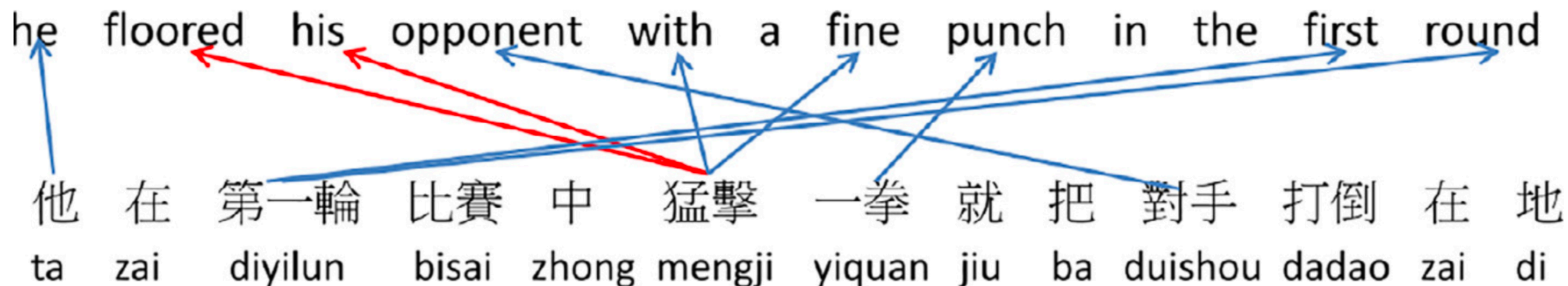
- It will be better to align words between S_t and S'_t



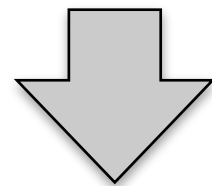
Automatic Word Alignment

- Parallel corpora at the word level are limited and relatively small-scale.
- So, automatic word alignment algorithm is applied to construct the large-scale word-level parallel corpus.
 - GIZA++ (En/Zh)
 - Tsinghua Alignment (En/Zh)

Bilingual Word Embeddings with Word Alignment



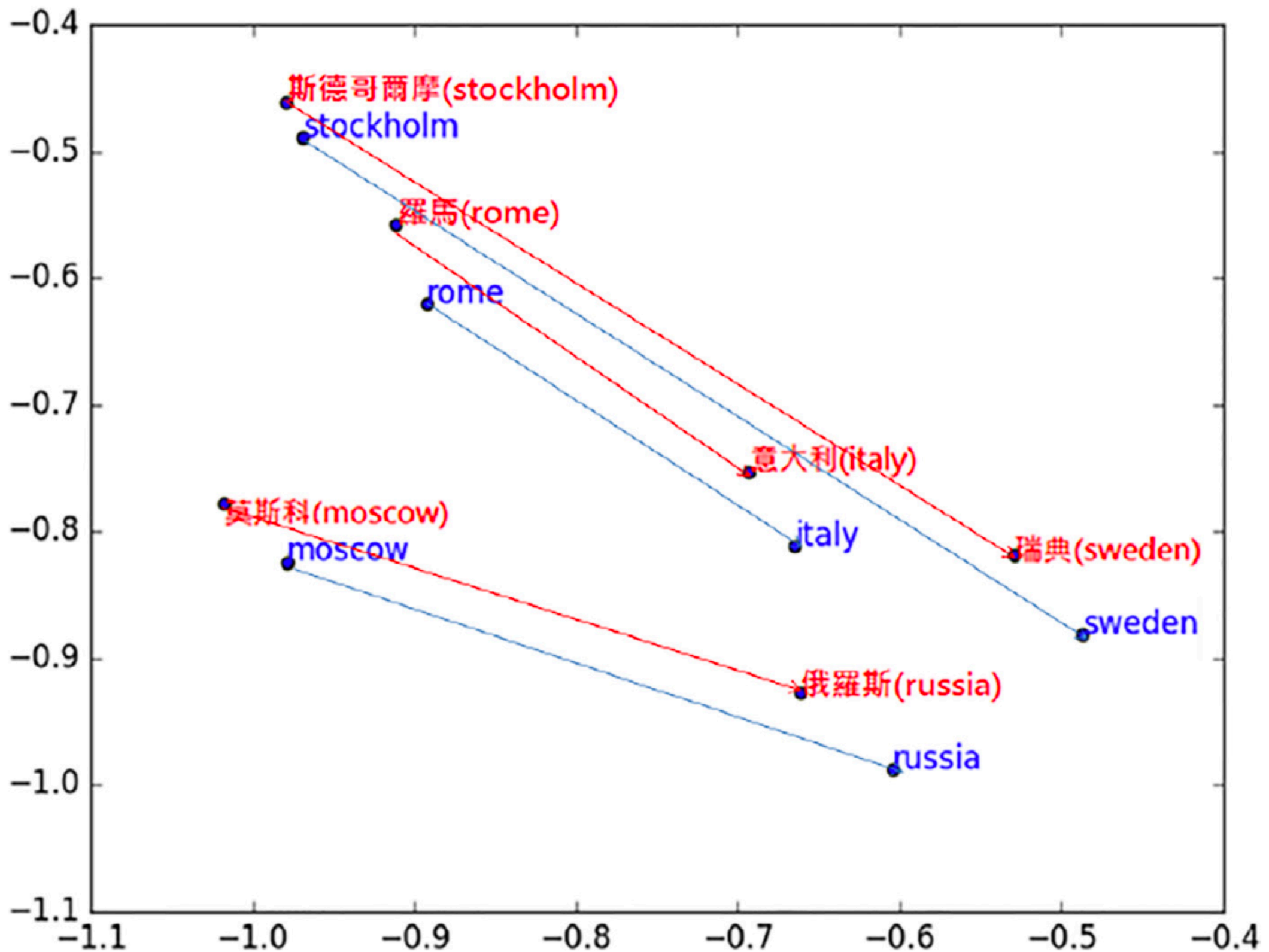
$$\sum_{t=1}^T \sum_{c \in \mathcal{C}_t} \log p(w_c | w_t)$$



$$\Omega_e = \frac{1}{M} \sum_{(S_e, S_c \in T_{e,c})} \sum_{w_e \in S_e} \left[\sum_{cw_e \in S_e[i_e-l:i_e+l]} \log p(cw_e | w_e) + \sum_{cw_c \in S_c[\frac{i_e-l}{k}:\frac{i_e+l}{k}]} \log p(cw_c | w_e) \right]$$

Consider contextual words from the other language

Cross-lingual Analogical Reasoning



Variants of Word Embedding Models

- Objective function

- Skip-gram vs CBOW

$$\sum_{t=1}^T \sum_{c \in \mathcal{C}_t} \log p(w_c | w_t)$$

$$\sum_{t=1}^T \sum_{c \in \mathcal{C}_t} \log p(w_t | w_c)$$

- Definition of the "context" information
 - word2vec: Surrounding words
 - fasttext: Surrounding words + common subwords
 - Bilingual word embeddings: Surrounding words + words in the translated counterpart.

Applications of Word Embeddings

Applications of Word Embeddings

- Historical linguistics
 - Studying the change of word usages over time
- Lexicon mining
 - Finding related words / collocations
- **Similarity measurement for information retrieval**
- **Classification**

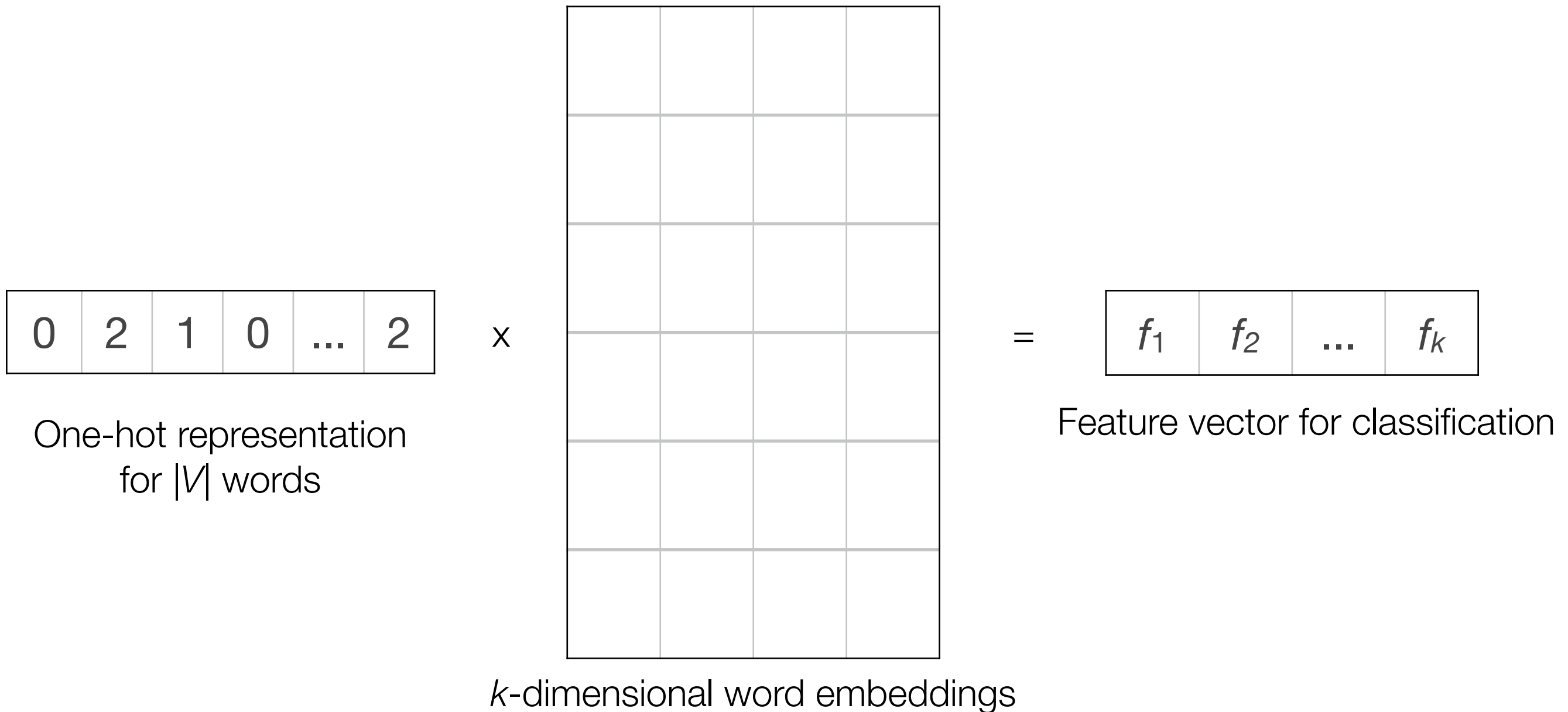
Sentence/Document Representation with Word Embeddings

- Word embeddings provide representation for each individual word.
- However, most IR/NLP applications deal with larger text units such as sentence, paragraph, and document.
- How to represent a sentence/paragraph/document with word embeddings?

Sentence/Document Representation with Word Embeddings

- Sum
- Weighted sum
- Contextual embeddings
 - Generating variant word vector for a word according to the word's context.

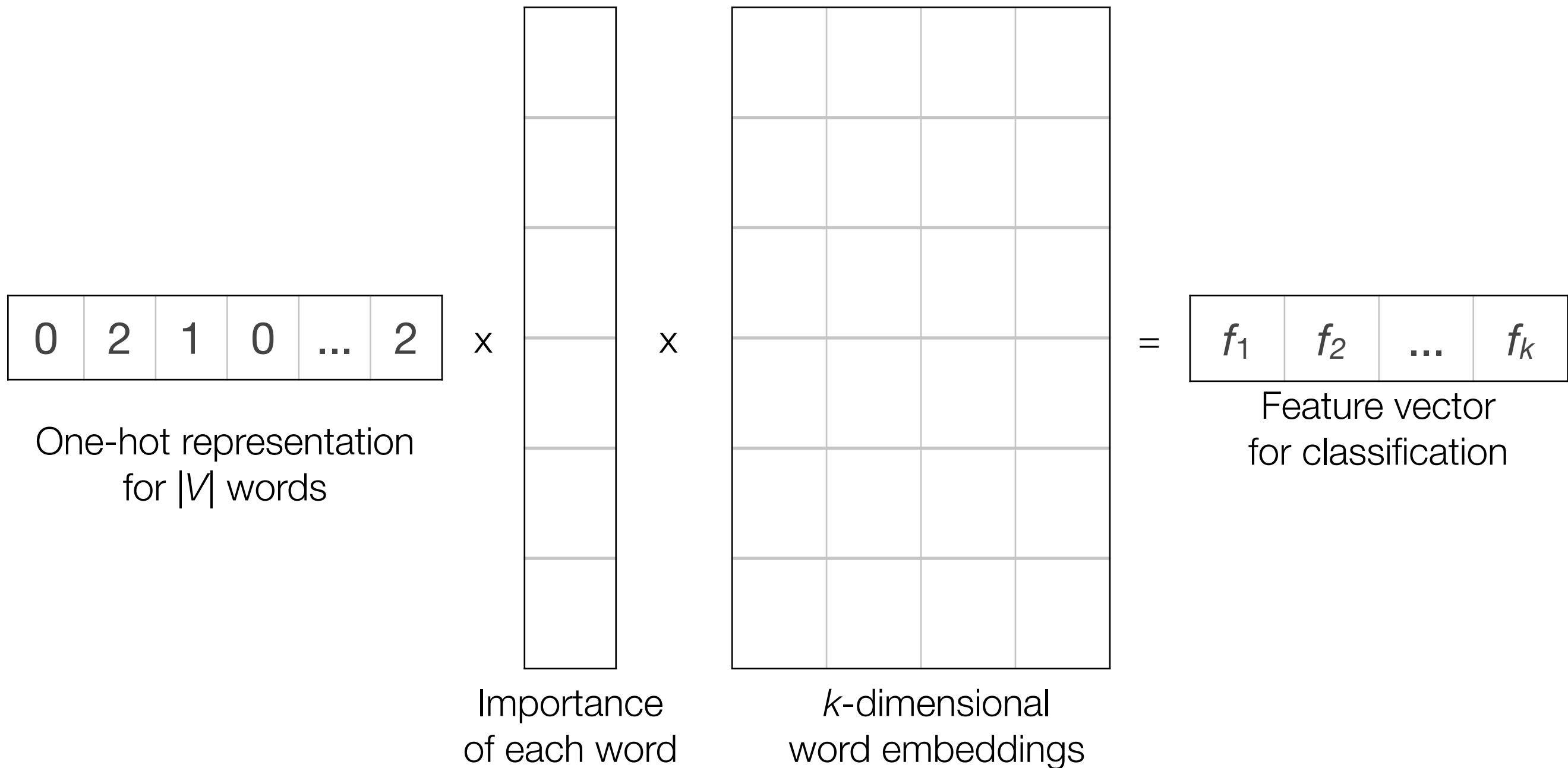
Sum of Word Embeddings



$$E_{\mathbf{x}} = E_{x_1} + E_{x_2} + E_{x_3} + \dots + E_{x_n}$$

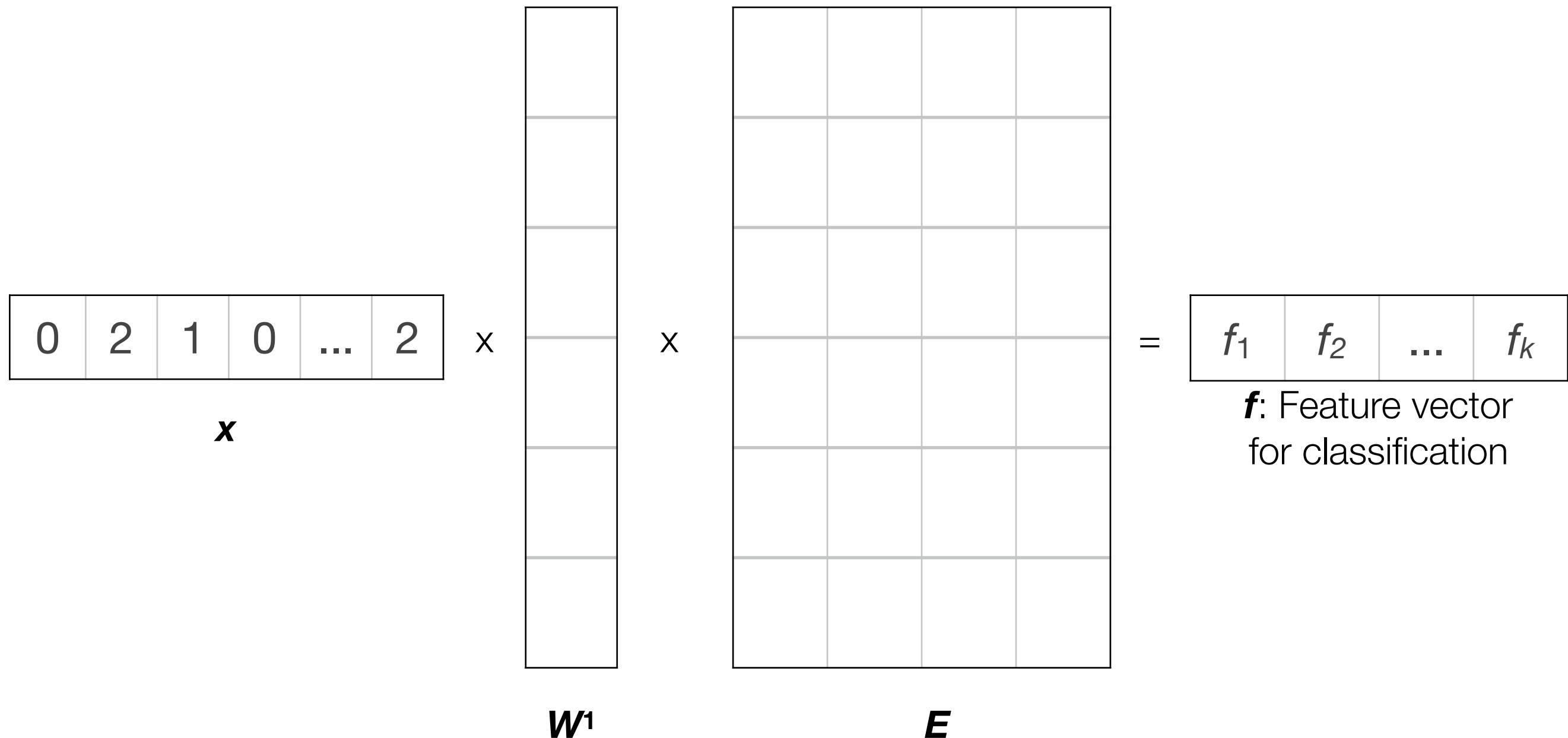
$$E_{\mathbf{x}} = \frac{E_{x_1} + E_{x_2} + E_{x_3} + \dots + E_{x_n}}{n}$$

Weighted Sum of Word Embeddings



$$E_{\mathbf{x}} = w_1 E_{x_1} + w_2 E_{x_2} + w_3 E_{x_3} + \dots + w_n E_{x_n}$$

Logistic Regression with Weighted Sum of Word Embeddings for Classification



$$\bar{y} = \text{sigmoid}(fW^2) = \text{sigmoid}(xW^1EW^2)$$

Parameters to learn

Sentence Embeddings

- A set of approaches that encode a sentence/paragraph/document into a vector/matrix.
 - Skip-thought
 - ELMO
 - BERT
 - XLNet
- To be continued next week.