CSCI 544: Applied Natural Language Processing

# Word Embeddings

Xuezhe Ma (Max)

USC University of Southern California

# Logistical Notes

- **Project Group Formation Deadline: 02/04**
  - https://docs.google.com/spreadsheets/d/1b62x9-Qzf5NI0l9KWX1VRPCm4K7Q6fV-v6BZgF5ZGS0/edit?usp=sharing
  - A group of exact 5 students

# Course Organization

**NLP Tasks**

| Models | Text Classification | Sequence Labeling | Syntactic Parsing | Seq2seq & MT | Language Modeling & Generation |
|---|---|---|---|---|---|
| | Naive Bayes<br>Logistic Regression<br>SVM … | HMMs<br>MEMMs<br>CRFs | Tree-structured Models/Algorithms | …… | …… |
| | $\boxed{\textbf{Word Embedding} \text{ \& Neural Networks}}$ | RNN<br>LSTM | Neural Structured Networks | Attention & Transformers | Generative Models & LLMs |

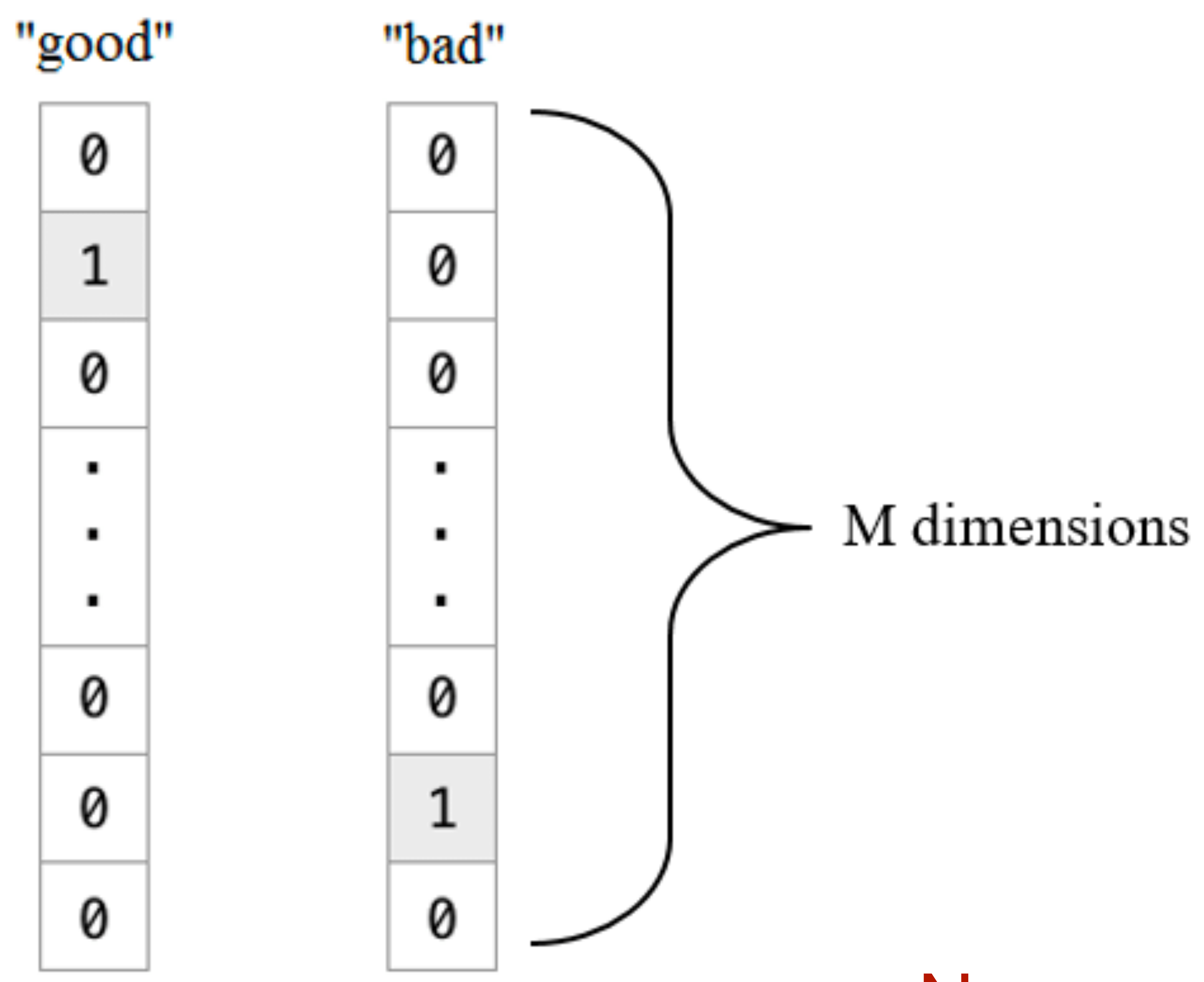
# Recap: Problems of Traditional Text Classification

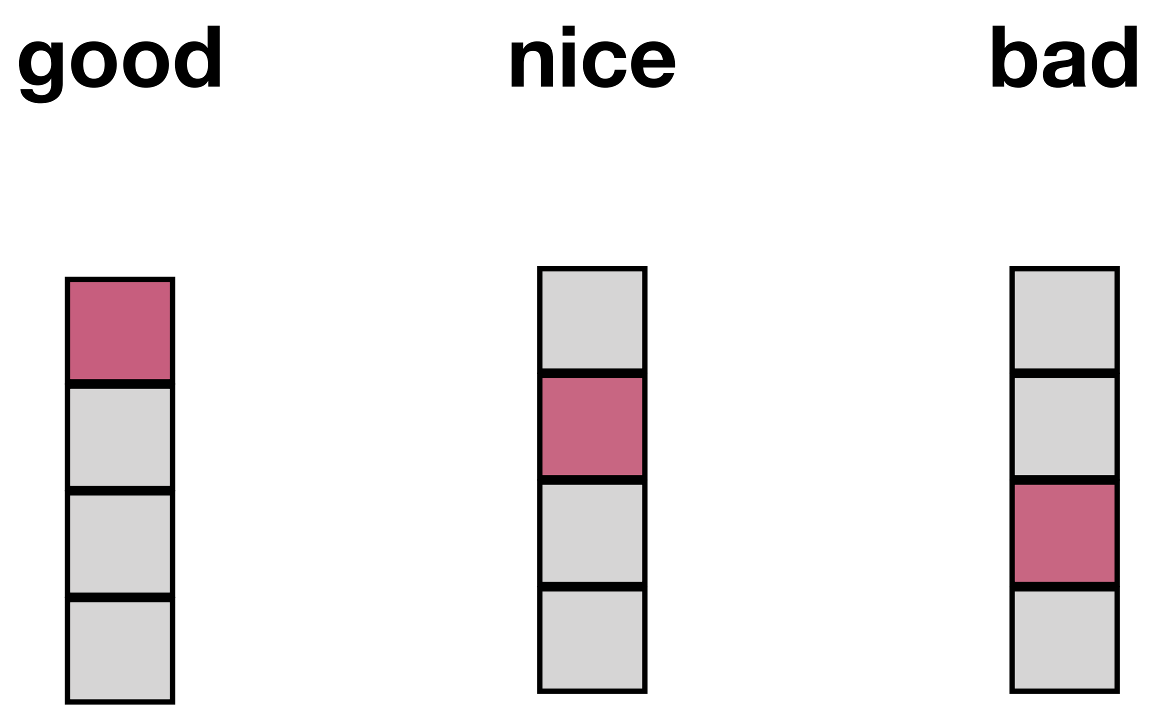
- **Insufficient attention on feature representations**
  - Bag of words & TF-IDF
  - Only frequency information, not semantic meaning of each word
  - No contextual information

# Problem: Frequency-based Features

## One-hot binary vectors

A vocabulary of $M$ words




$$\mathrm{dist}(\textbf{good}, \textbf{nice}) = \mathrm{dist}(\textbf{good}, \textbf{bad})$$

No semantic meaning for words!

# Problem: Contextual Information

**Bat**

**hit with bat**

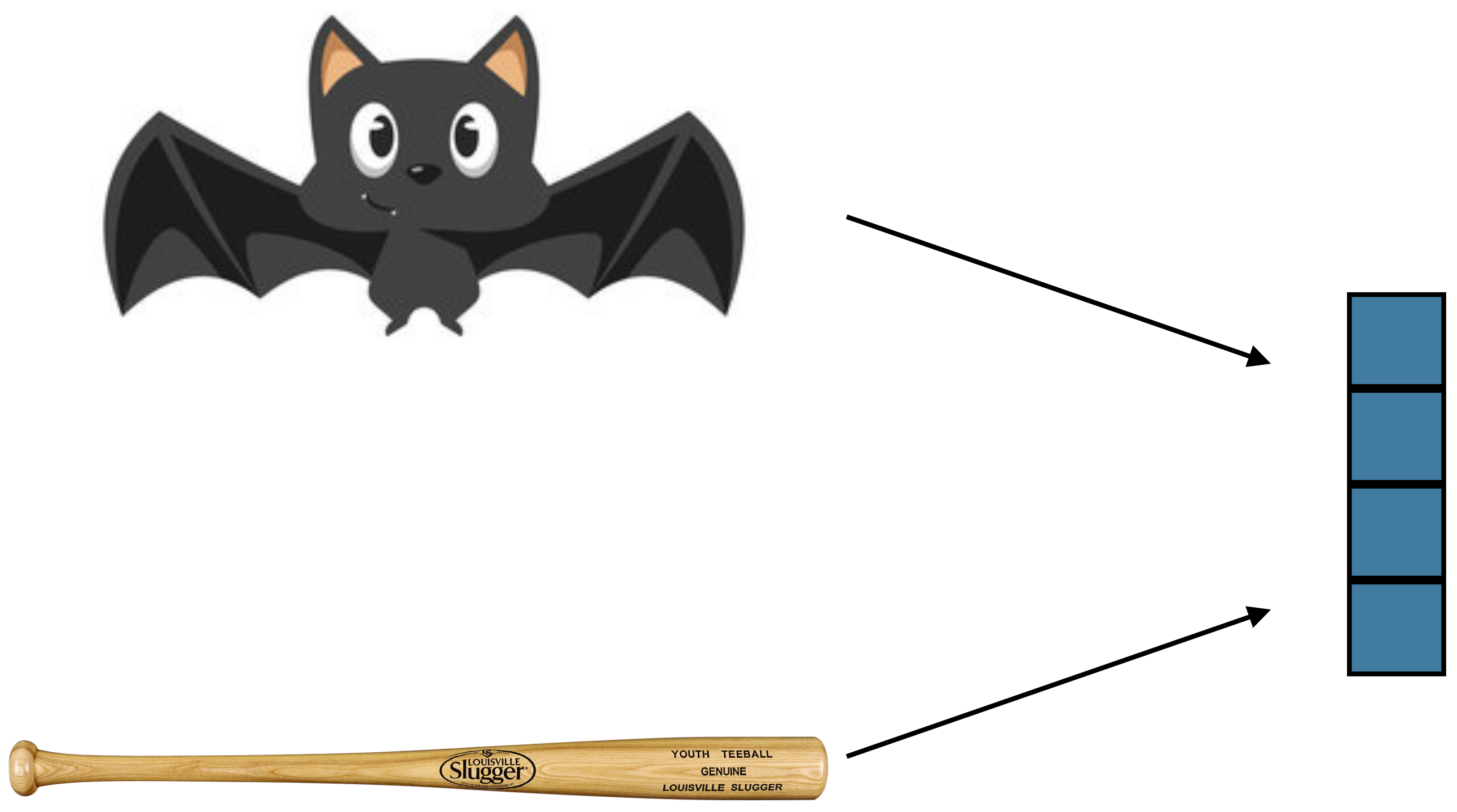
# Recap: Problems of Traditional Text Classification

- **Insufficient attention on feature representations**
  - Bag of words & TF-IDF
  - Only frequency information, not semantic meaning of each word
  - No contextual information (next lecture)

# Word Embeddings

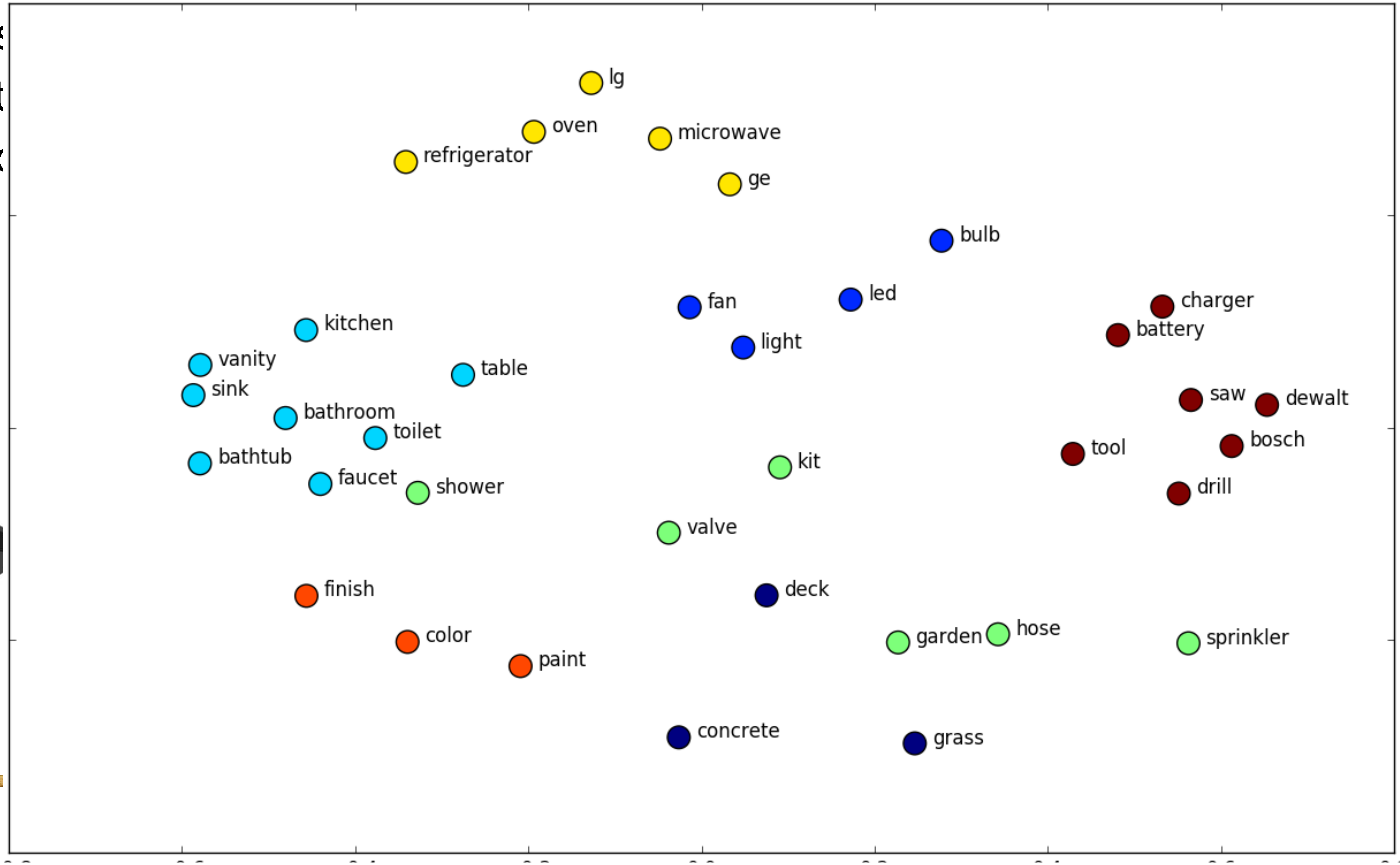
- **One dense vector per word**
  - The same size is used for all words
  - Relatively low dimensional (e.g. d < 300)
  - Vectors for similar words are similar (w.r.t distance measures)

**Bat**



# Word Embeddings

- One dense vector per word
  - The s
  - Relat
  - Vect




**Bat**

# Learning Word Embeddings

- **Training Procedure**
  - **Init:** Randomly initialize embedding vectors, one vector per word (e.g sampling from Normal distribution)
  - **Training Loop:**
    - For each positive word pair $(w_1, w_2)$:
      - Decreasing their distance: $\min \text{dist}(w_1, w_2)$
    - For each negative word pair $(w_1, w_2)$:
      - Increasing their distance: $\max \text{dist}(w_1, w_2)$

# Learning Word Embeddings

- **What is supervision?**
  - Manually defined similarity scores between words?
  - good vs. nice vs. great vs. bright vs. fine vs. … …

**The Distributional Hypothesis in Computational Linguistics:**
"Similar words occur in similar contexts" (Firth, '57)

# Learning Word Embeddings

- How to define contexts?
- How to calculate similarities between words?

**The Distributional Hypothesis in Computational Linguistics:**
"Similar words occur in similar contexts" (Firth, '57)

| Prediction-based | Factorization-based |
| --- | --- |
| Word2Vec | GloVe |

# Word2Vec

- **Test-of-time Award @ NeurIPS 2023**

## Distributed Representations of Words and Phrases and their Compositionality


**Tomas Mikolov**
Google Inc.
Mountain View
mikolov@google.com

**Ilya Sutskever**
Google Inc.
Mountain View
ilyasu@google.com

**Kai Chen**
Google Inc.
Mountain View
kai@google.com

**Greg Corrado**
Google Inc.
Mountain View
gcorrado@google.com

**Jeffrey Dean**
Google Inc.
Mountain View
jeff@google.com

# Word2Vec

- **Contexts:** surrounding words of a fixed small window in a piece of texts

I am a student at USC

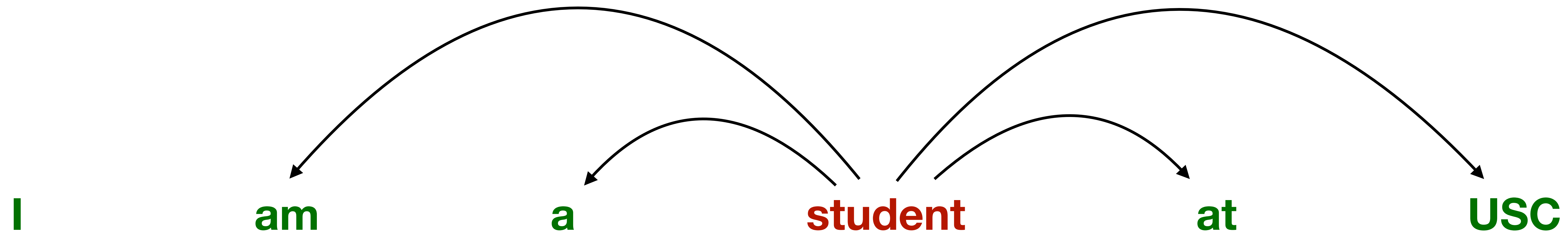
- **Similarity**: conditional probability to predict a word occurring in the same context

$$P(W_{out} | W_{in})$$

# Word2Vec

- **Core Idea:** learning embeddings using a prediction task involving neighboring words in a huge real-world corpus

- **Skip-Gram:** given a center word, we predict the context words




$P(\textbf{am} \mid \textbf{student})$ $\quad$ $P(\textbf{at} \mid \textbf{student})$

$P(\textbf{a} \mid \textbf{student})$ $\quad$ $P(\textbf{USC} \mid \textbf{student})$

# Word2Vec

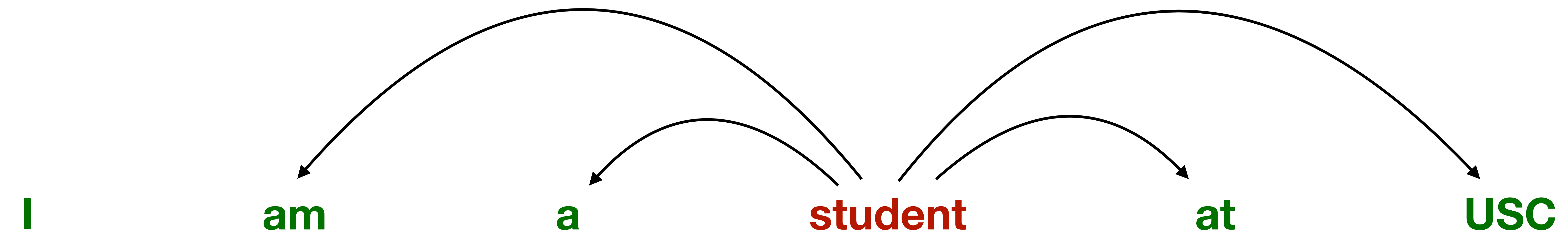
- **Vocabulary**: a dictionary of words $V$
- **Two sets of embedding vectors**
  - For each word $w \in V$
  - $u_w$ is the input vector of the word $w$
  - $v_w$ is the output vector of the word $w$
- **Prediction probability via softmax function**

$$P(w_{out} | w_{in}) = \frac{\exp(v_{w_{out}}^T \cdot u_{w_{in}})}{\sum_{w \in V} \exp(v_w^T \cdot u_{w_{in}})}$$

# Word2Vec

- **Step1**: Collect and pre-process a huge real-world corpus
- **Step2**: Create a vocabulary $V$
- **Step3**: Go through the full corpus
  - For each valid context, update embedding vectors to maximize

$$P(w_{out} | w_{in}) = \frac{\exp(v_{w_{out}}^T \cdot u_{w_{in}})}{\sum_{w \in V} \exp(v_w^T \cdot u_{w_{in}})}$$




$P(\textbf{am} | \textbf{student})$ $P(\textbf{at} | \textbf{student})$

$P(\textbf{a} | \textbf{student})$ $P(\textbf{USC} | \textbf{student})$

# Challenges in Word2Vec

- **Sparsity Problem**
  - Vectors of frequent words get more updates than rare words

- **Expensive Computation**
  - $$P(w_{out} | w_{in}) = \frac{\exp(v_{w_{out}}^T \cdot u_{w_{in}})}{\boxed{\sum_{w \in V} \exp(v_w^T \cdot u_{w_{in}})}}$$

# Sub-Sampling in Word2Vec

- **Discarding frequent words with some probability**
  - For each word $w \in V$ in the training data, we discard it with probability

$$P(w) = 1 - \sqrt{\frac{t}{f(w)}}$$

$f(w)$ is the word frequency, $t$ is a hyper-parameter (e.g. $t = 10^{-5}$)

# Challenges in Word2Vec

- **Sparsity Problem**
  - Vectors of frequent words get more updates than rare words

- **Expensive Computation**
  - $$P(w_{out} | w_{in}) = \frac{\exp(v_{w_{out}}^T \cdot u_{w_{in}})}{\boxed{\sum_{w \in V} \exp(v_w^T \cdot u_{w_{in}})}}$$

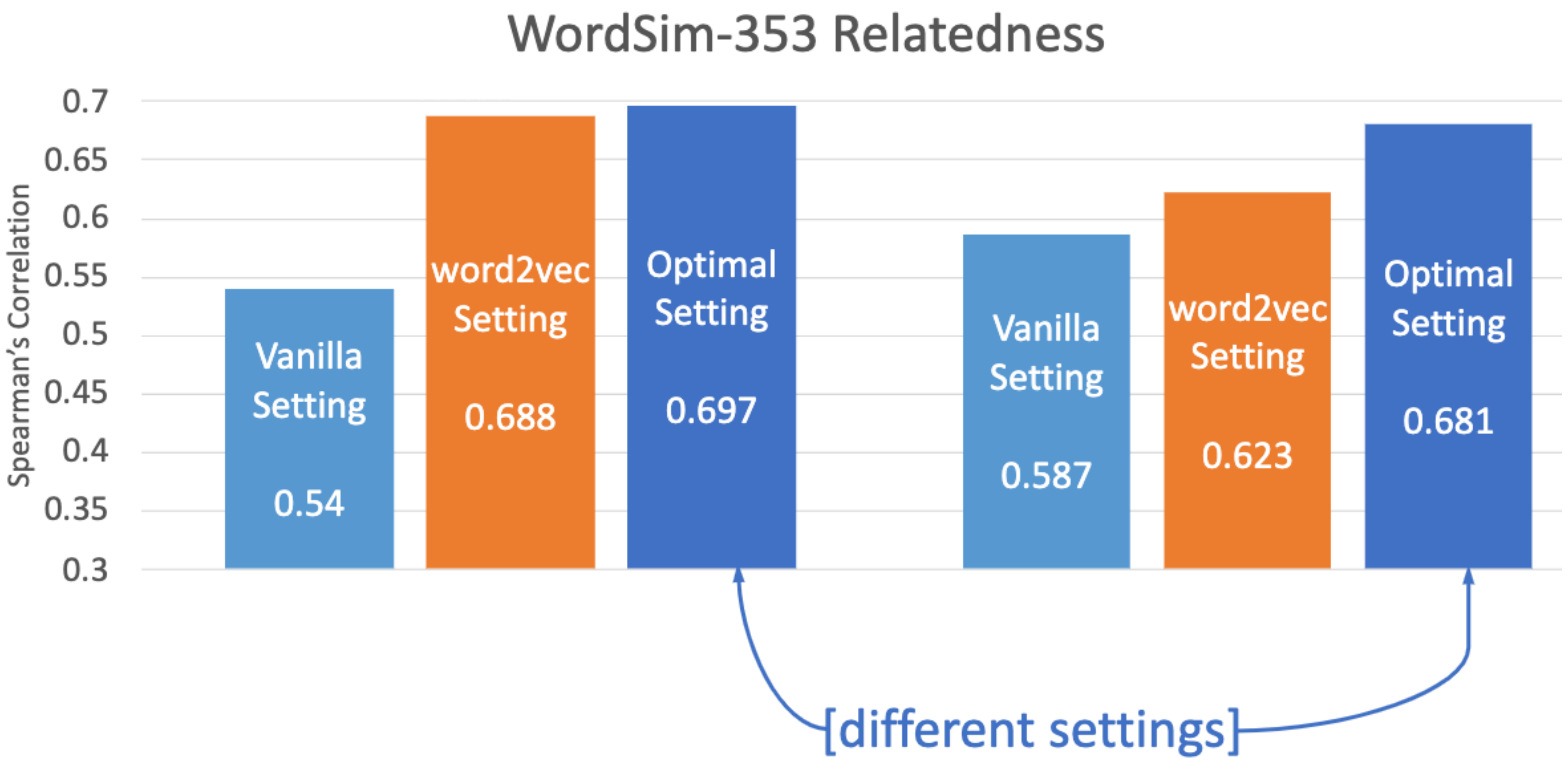
# Negative Sampling

- **Approximating the nominator with $K$ random samples**
  - Sampling $K$ words from a noise distribution $w_1, \ldots w_K \sim q(w)$
  - Approximate the loss

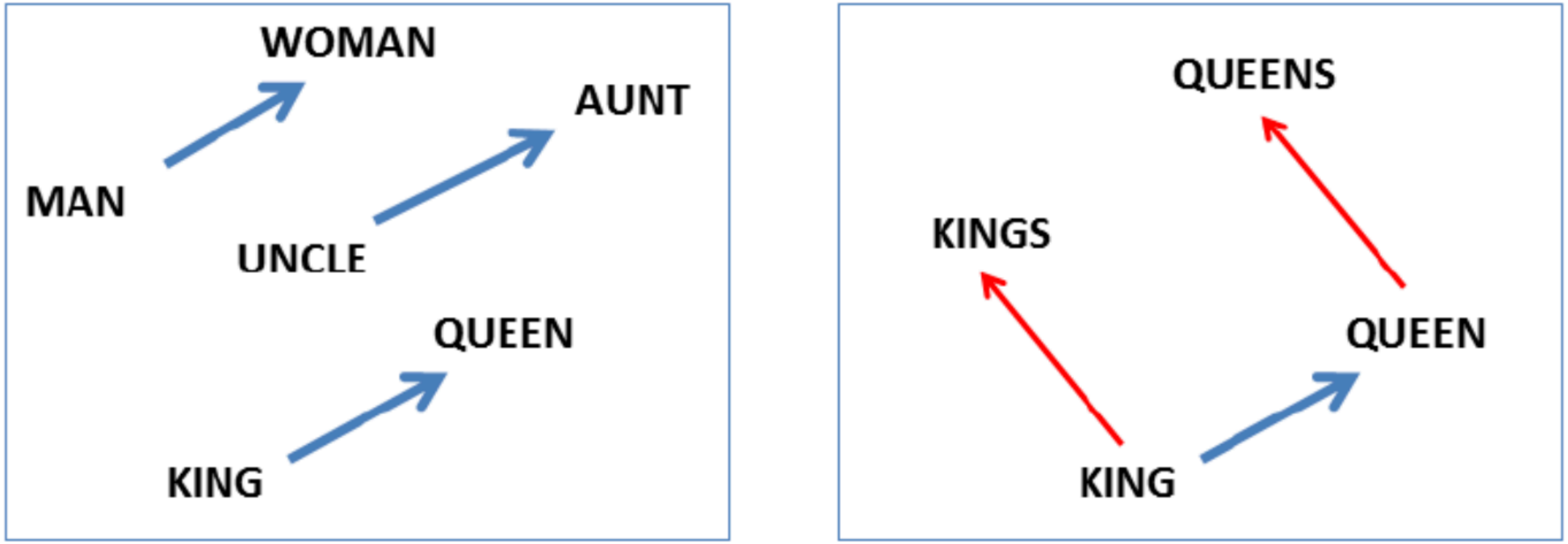
$$\log P(w_{out} | w_{in}) \approx \log \sigma(v_{w_{out}}^T \cdot u_{w_{in}}) - \sum_{k=1}^{k} \log \sigma(v_{w_k}^T \cdot u_{w_{in}})$$

  - $\sigma$ is the logistic function $\sigma(x) = \dfrac{1}{1 + e^{-x}}$
  - The choice of $q(w)$
    - Neither too far away nor too close to $p(w)$
    - In the Word2Vec paper, the author choose $q(w) \sim p(w)^{3/4}$

# Word2Vec


WordSim-353 Relatedness
Spearman's Correlation
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
Vanilla Setting 0.54
word2vec Setting 0.688
Optimal Setting 0.697
Vanilla Setting 0.587
word2vec Setting 0.623
Optimal Setting 0.681
[different settings]

# Word2Vec



# Factorization-based Methods

- **Word2Vec is hard to be interpreted**
  - Any theoretical insights?

**GloVe: Global Vectors for Word Representation**

**Jeffrey Pennington, Richard Socher, Christopher D. Manning**
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

# GloVe

- **Using word vectors to approximate pairwise mutual information (PMI) of two words**

$$\mathrm{PMI}(\mathrm{word}_1, \mathrm{word}_2) = \log_2 \frac{P(\mathrm{word}_1, \mathrm{word}_2)}{P(\mathrm{word}_1)P(\mathrm{word}_2)}$$

# GloVe

- Using word vectors to approximate pairwise mutual information (PMI) of two words

$$\mathrm{PMI}(\mathrm{word}_1, \mathrm{word}_2) = \log_2 \frac{P(\mathrm{word}_1, \mathrm{word}_2)}{P(\mathrm{word}_1)P(\mathrm{word}_2)}$$

- Similar to Word2Vec, there are two sets of word vectors in GloVe
  - $u_w$ is the input vector of the word $w$
  - $v_w$ is the output vector of the word $w$

$$\exp(v_{w_{out}}^T \cdot u_{w_{in}}) \approx \textbf{PMI}(w_{out}, w_{in})$$

# GloVe: Matrix-Factorization

- **Given all possible input and output words**

$$V^T \cdot U \approx \log \mathbf{M}$$

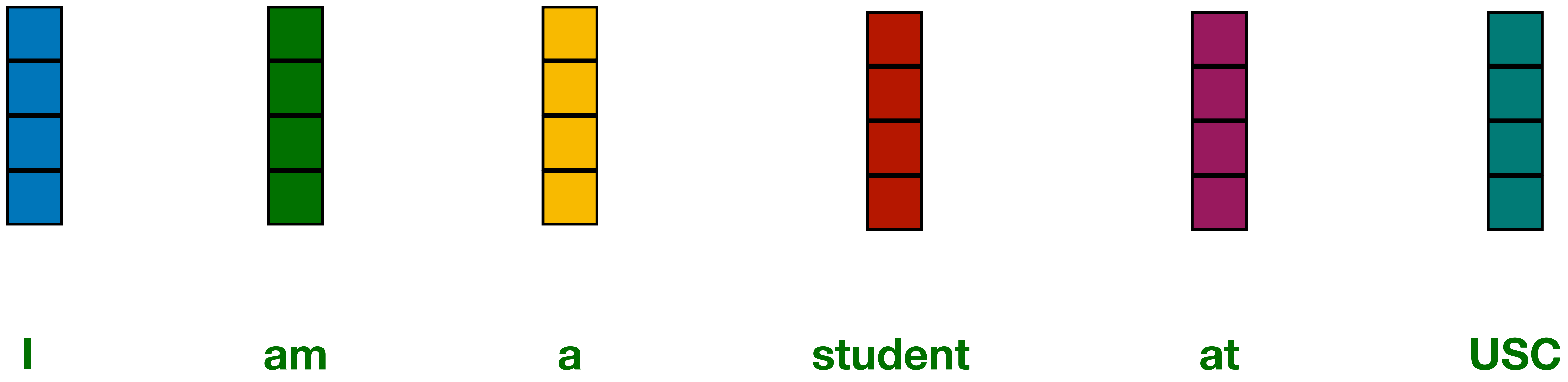
- **Solution: the factorization of M with rank $d$**

# Word2Vec vs GloVe

- **No particular word embedding approach is the SOTA for all applications**
- **Some key factors:**
  - Amount and quality of training data
  - Hyper-parameters
    - Vector dimension $d$
    - Subsampling $t$
    - Negative sampling $q(w)$ and $K$
    - Matrix factorization algorithms in GloVe
    - ……

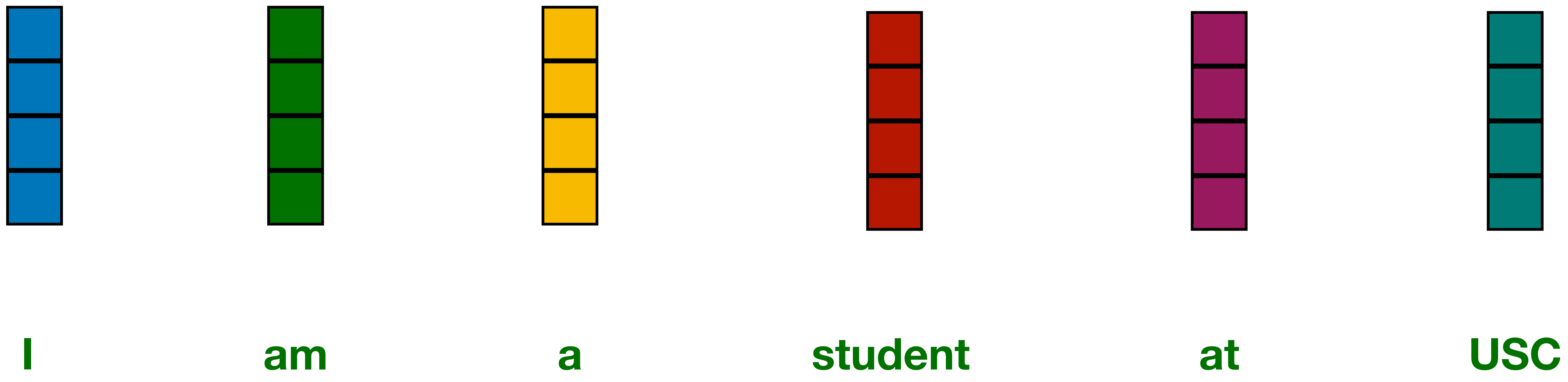
# Apply Word Embeddings to Tasks

- **Classification**
  - We need a single feature vector to feed into ML classifiers



# Apply Word Embeddings to Tasks

- **Classification**
  - We need a single feature vector to feed into ML classifiers

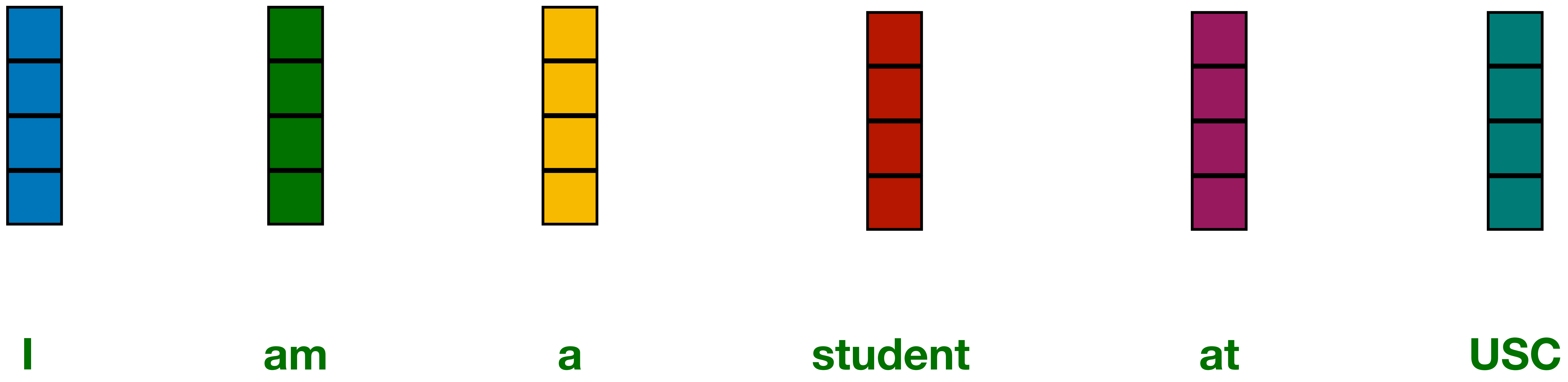



Element-wise pooling: $v = \textbf{pool}(v_1, \ldots, v_n)$

e.g. AVG, MAX

# Apply Word Embeddings to Tasks

- **Classification**
  - We need a single feature vector to feed into ML classifiers




Element-wise pooling: $v = \mathbf{pool}(v_1, \ldots, v_n)$

e.g. AVG, MAX

No word-order/context information!

# Q&A