Neural Network Approaches to Representation Learning for NLP

Navid Rekabsaz
Idiap Research Institute

navid.rekabsaz@idiap.ch @navidrekabsaz

Agenda

- Brief Intro to Deep Learning
  - Neural Networks
- Word Representation Learning
  - Neural word representation
  - Word2vec with Negative Sampling
  - Bias in word representation learning

---Break---

- Recurrent Neural Networks
- Attention Networks
- Document Classification with DL

Agenda

- **Brief Intro to Deep Learning**
  - **Neural Networks**
- Word Representation Learning
  - Neural word representation
  - word2vec with Negative Sampling
  - Bias in word representation learning

---Break---

- Recurrent Neural Networks
- Attention Networks
- Document Classification with DL

Recap on Linear Algebra

- Scalar a
- Vector $\vec{b}$
- Matrix W
- Tensor: generalization to higher dimensions
- Dot product
  - $\vec{a} \cdot \vec{b}^T = c$

    dimensions: 1×d · d×1 =1
  - $\vec{a} \cdot W = \vec{c}$

    dimensions: 1×d · d×e =1×e
  - $A \cdot B = C$

    dimensions: l×m · m×n =l×n
- Element-wise Multiplication
  - $\vec{a} \odot \vec{b} = \vec{c}$

Neural Networks

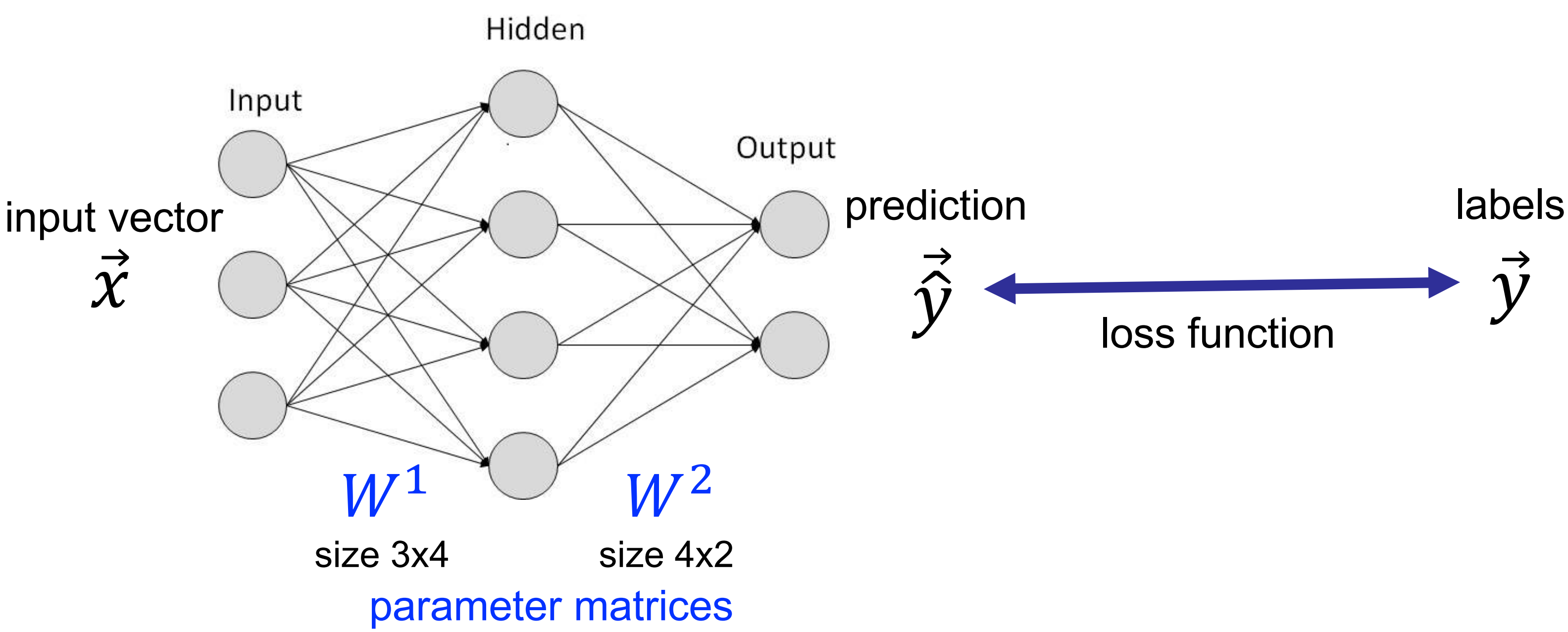
- Neural Networks are non-linear functions with many parameters

$$\vec{\hat{y}} = f(\vec{x})$$

- They consist of several simple non-linear operations
- Normally, the objective is to maximize likelihood, namely

$$p(y|x,\theta)$$

- Generally optimized using Stochastic Gradient Descent (SGD)

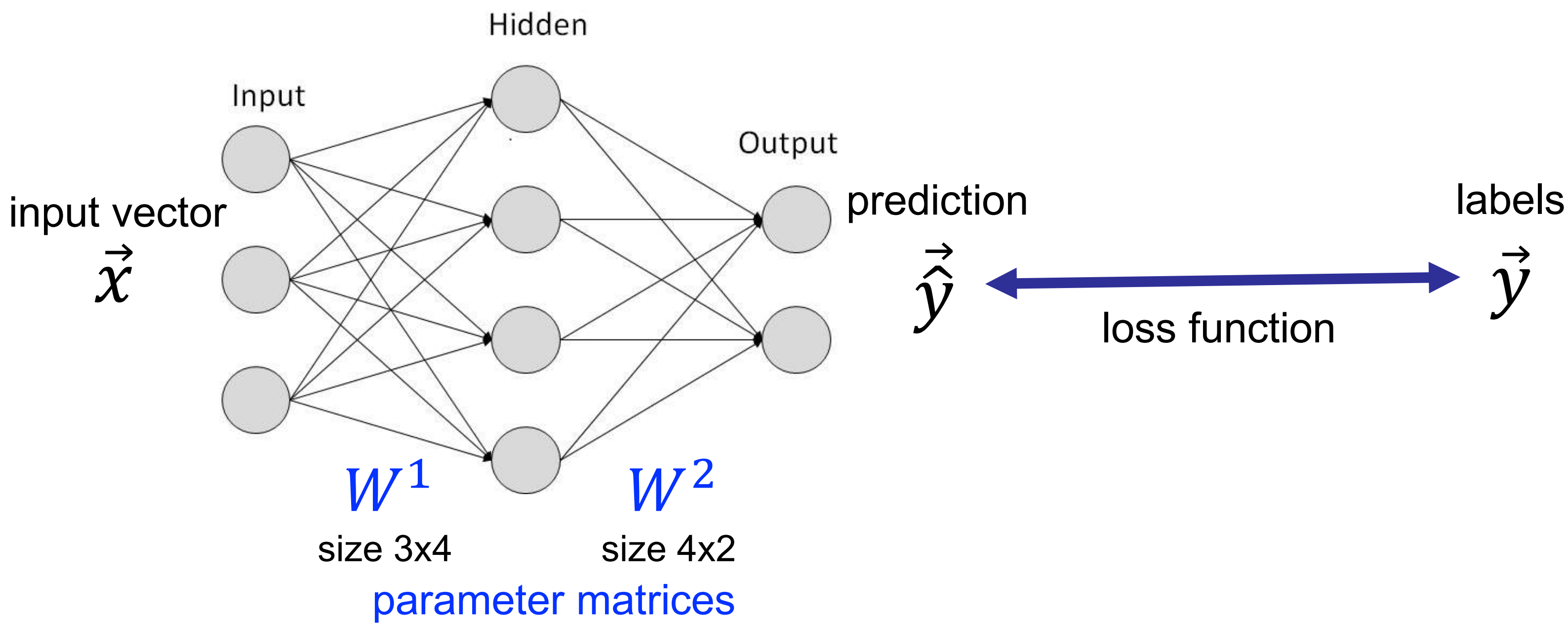


Neural Networks – Training with SGD (simplified)

Initialize parameters

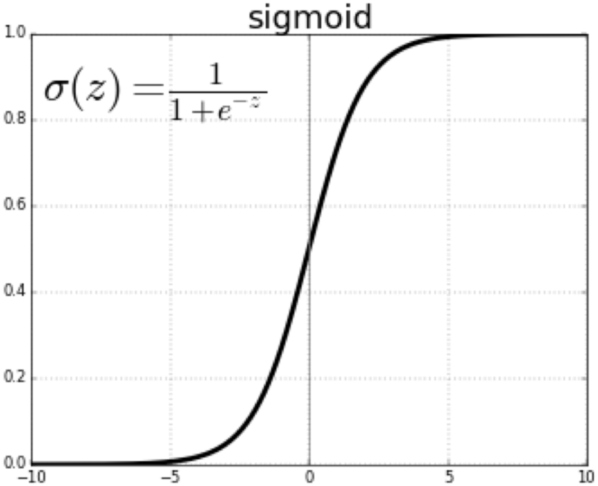
Loop over training data (or minibatches)

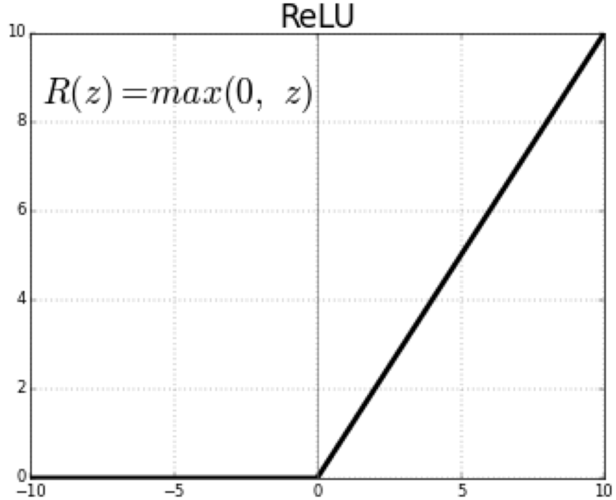
1. Do forward pass: given input $\vec{x}$ predict output $\hat{y}$
2. Calculate loss function by comparing $\hat{y}$ with labels y
3. Do backpropagation: calculate the gradient of each parameter in regard to the loss function
4. Update parameters in the direction of gradient
5. Exit if some stopping criteria are met

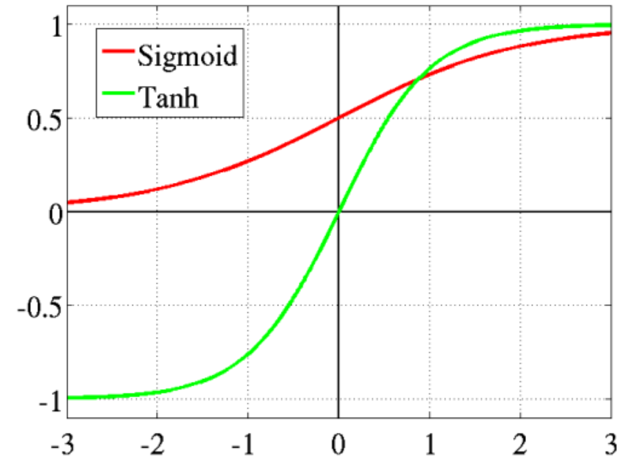


Neural Networks – Non-linearities

- Sigmoid
  - Projects input to value between 0 to 1 → becomes like a probability value
- ReLU (Rectified Linear Units)
  - Suggested for deep architectures to prevent vanishing gradient
- Tanh







Fetched from https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Neural Networks - Softmax

- Softmax turns a vector to a probability distribution
  - The vector values become in the range of 0 to 1 and sum of all the values is equal 1

$$softmax(\vec{v})_i = \frac{e^{v_i}}{\sum_{k=1}^{d} e^{v_k}}$$

- Normally applied to the output layer and provide a probability distribution over output classes

- For example, given four classes:

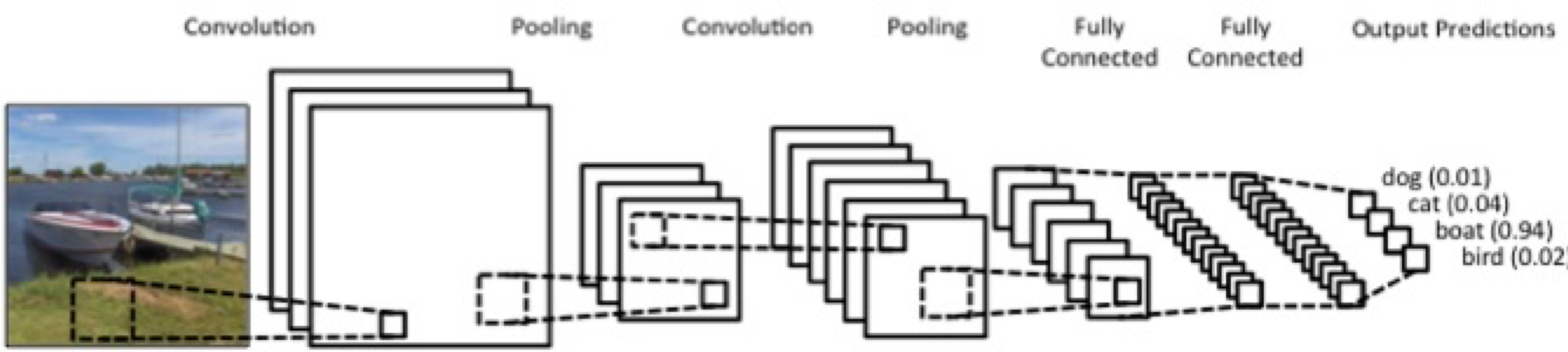
$$\vec{\hat{y}} = [2, 3, 5, 6] \qquad softmax(\hat{y}) = [0.01, 0.03, 0.26, 0.70]$$

Deep Learning

- Deep Learning models the overall function as a composition of functions (layers)

- With several algorithmic and architectural innovations
  - dropout, LSTM, Convolutional Networks, Attention, GANs, etc.

- Backed by large datasets, large-scale computational resources, and enthusiasm from academia and industry!



Adopted from http://mlss.tuebingen.mpg.de/2017/speaker_slides/Zoubin1.pdf

Agenda

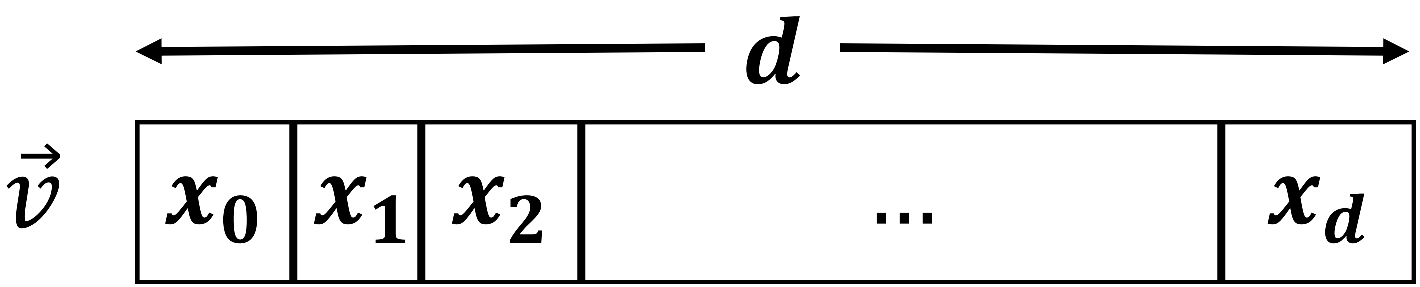
- Brief Intro to Deep Learning
  - Neural Networks
- **Word Representation Learning**
  - **Neural word representation**
  - **word2vec with Negative Sampling**
  - **Bias in word representation learning**

---Break---

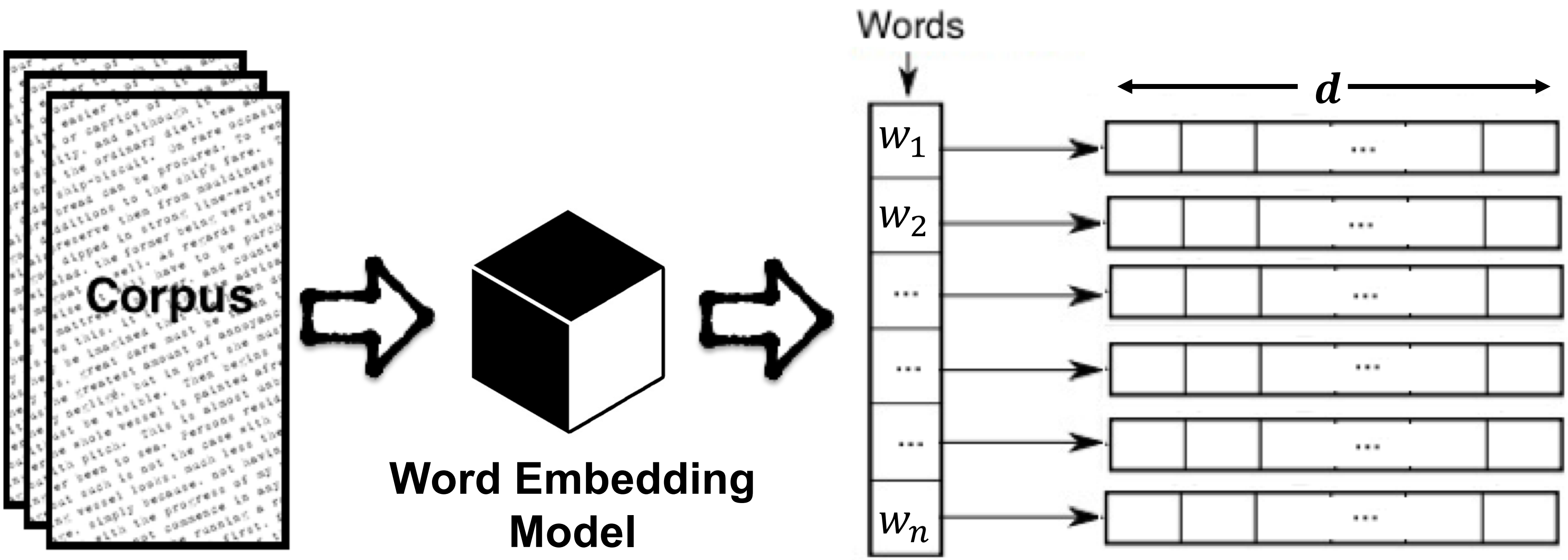
- Recurrent Neural Networks
- Attention Networks
- Document Classification with DL

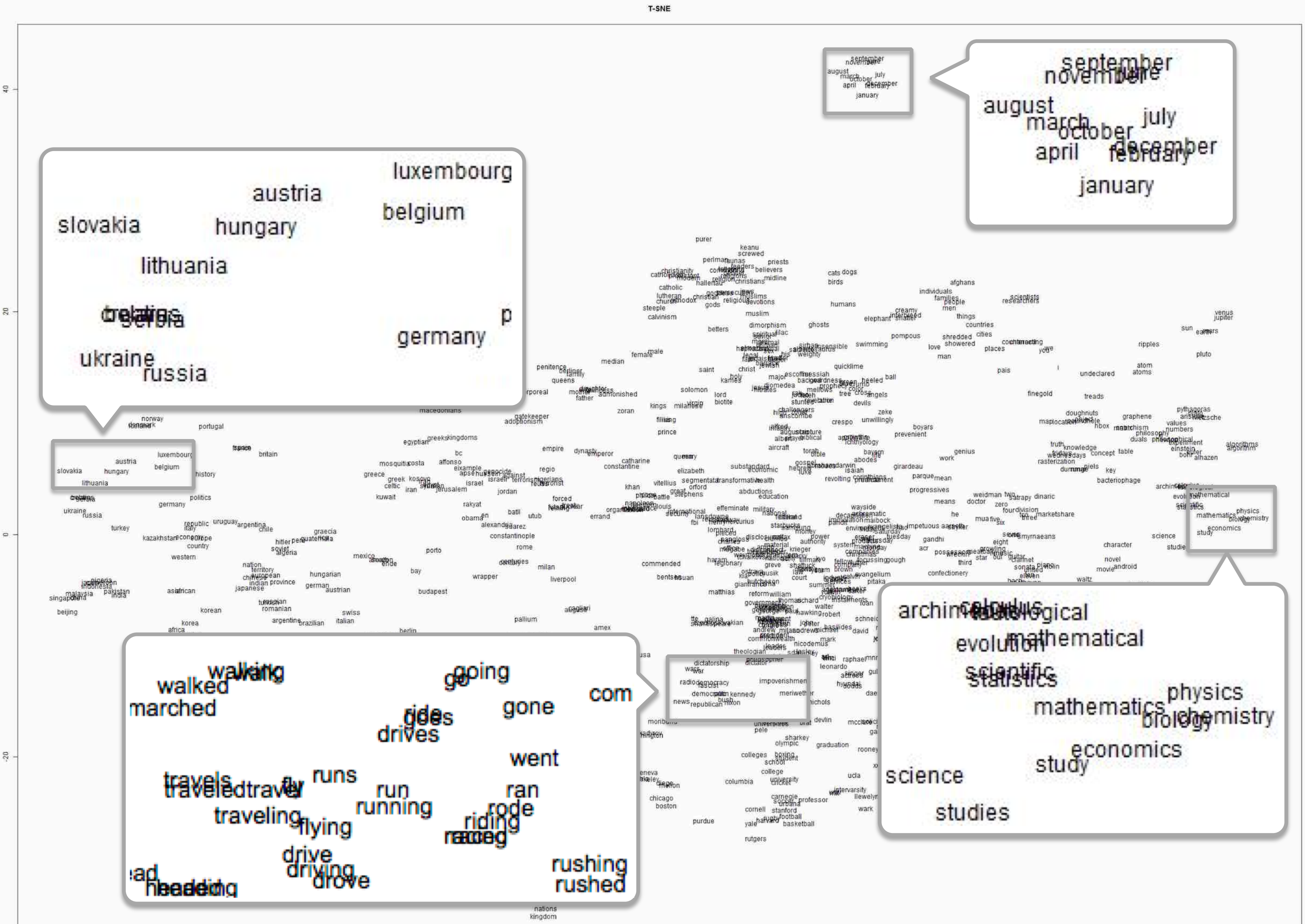
Vector Representation (Recall)

- Computation starts with representation of entities
- An entity is represented with a vector of *d* dimensions
- The dimensions usually reflects features, related to an entity
- When vector representations are dense, they are often referred to as embedding e.g. word embedding



Word Representation Learning





Vector representations of words projected in two-dimensional space

Intuition for Computational Semantics

"You shall know a word by the company it keeps!"

J. R. Firth, A synopsis of linguistic theory 1930–1955 (1957) drink

sacred

alcoholic

Tesgüino

beverage

out of corn

fermented

bottle

Mexico

Nida[1975]

fermentation

bottle

grain

medieval

Ale

brew

pale

bar

drink

alcoholic

Tesgüino ⟷ Ale

Algorithmic intuition:

Two words are related when they share many context words

Word-Context Matrix (Recall)

- Number of times a word *c* appears in the context of the word *w* in a corpus

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	c_1 Aardvark	c_2 computer	c_3 data	c_4 pinch	c_5 result	c_6 sugar
w_1 apricot	0	0	0	1	0	1
w_2 pineapple	0	0	0	1	0	1
w_3 digital	0	2	1	0	1	0
w_4 information	0	1	6	0	4	0

- Our first word vector representation!!

[1]

Words Semantic Relations (Recall)

	c_1 Aardvark	c_2 computer	c_3 data	c_4 pinch	c_5 result	c_6 sugar
w_1 apricot	0	0	0	1	0	1
w_2 pineapple	0	0	0	1	0	1
w_3 digital	0	2	1	0	1	0
w_4 information	0	1	6	0	4	0

- **Co-occurrence relation**
  - Words that appear near each other in the language
  - Like (*drink* and *beer*) or (*drink* and *wine*)
  - Measured by counting the co-occurrences
- **Similarity relation**
  - Words that appear in similar contexts
  - Like (*beer* and *wine*) or (*knowledge* and *wisdom*)
  - Measured by similarity metrics between the vectors

$$similarty(\text{digital}, \text{information}) = \text{cosine}\left(\vec{v}_{\text{digital}}, \vec{v}_{\text{information}}\right)$$

Sparse vs. Dense Vectors (Recall)

- Such word representations are highly sparse
  - Number of dimensions is the same as the number of words in the corpus $n \sim$ [10000–500000]
  - Many zeros in the matrix as many words don't co-occur
    - Normally ~98% sparsity
- Dense representations → Embeddings
  - Number of dimensions usually between $d \sim$ [10–1000]
- Why dense vectors?
  - More efficient for storing and load
  - More suitable for machine learning algorithms as features
  - Generalize better by removing noise for unseen data

Word Embedding with Neural Networks

Recipe for creating (dense) word embedding with neural networks

1. Design a neural network architecture!
2. Loop over training data (w, c)
   a. Set word w as input and context word c as output
   b. Calculate the output of network, namely
      The probability of observing context word c given word w
      $$P(c|w)$$
   c. Optimize the network to maximize the likelihood probability
3. Repeat

Details come next!

Prepare Training Samples

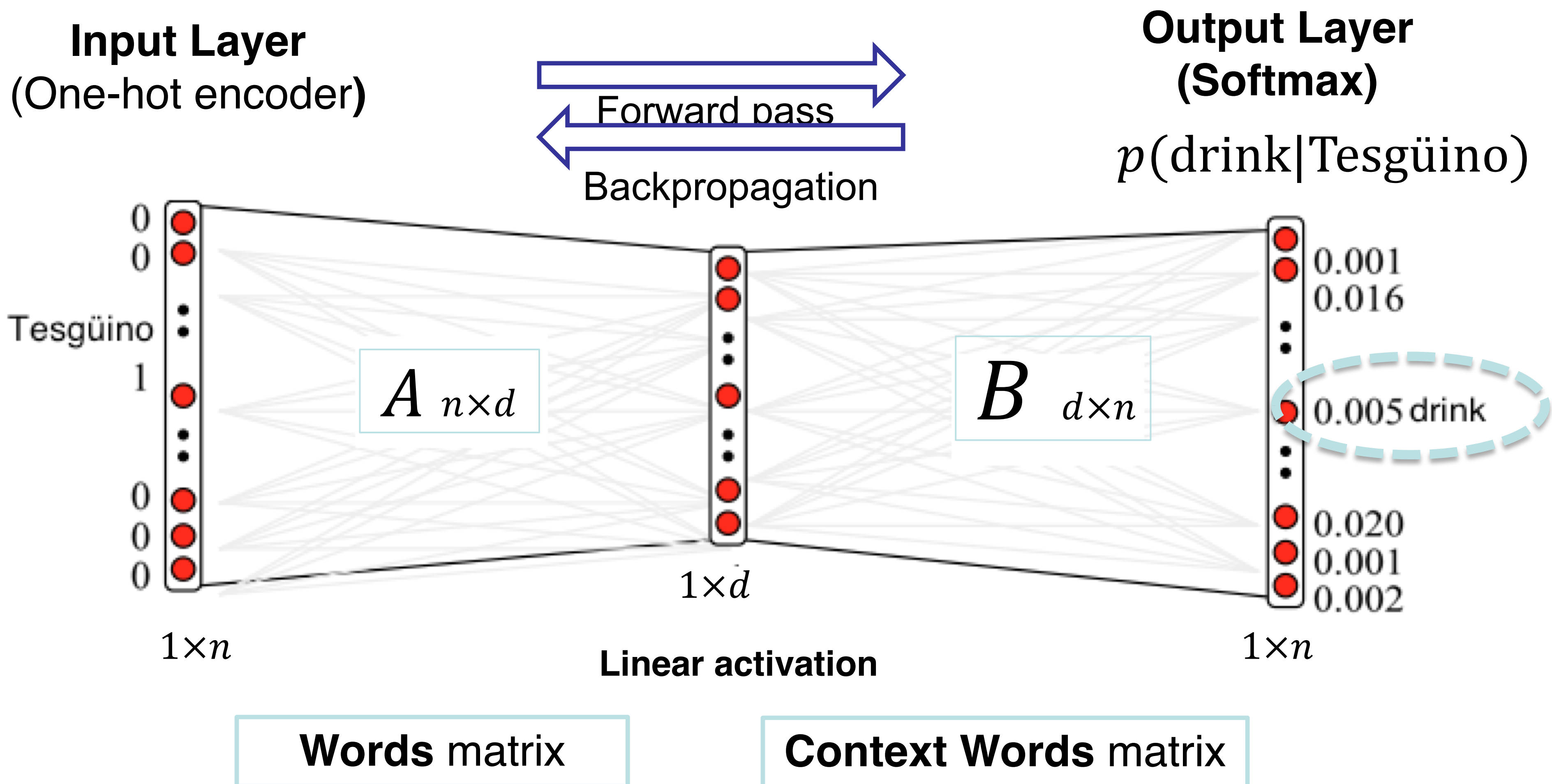
Window size of 2

Source Text	Training Samples
The quick brown fox jumps over the lazy dog. →	(the, quick) (the, brown)
The **quick** brown fox jumps over the lazy dog. →	(quick, the) (quick, brown) (quick, fox)
The quick **brown** fox jumps over the lazy dog. →	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown **fox** jumps over the lazy dog. →	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

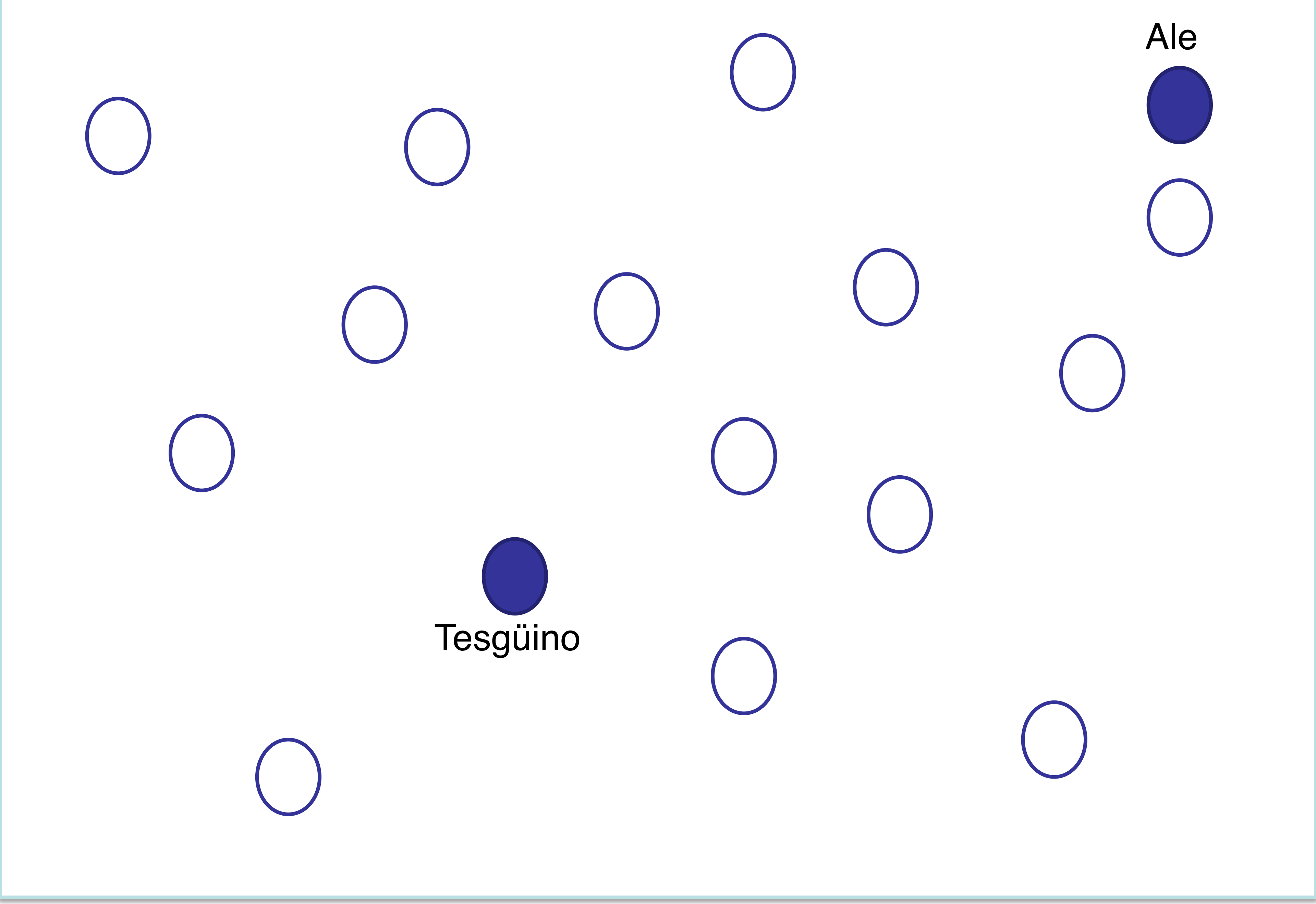
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Neural Word Embedding Architecture

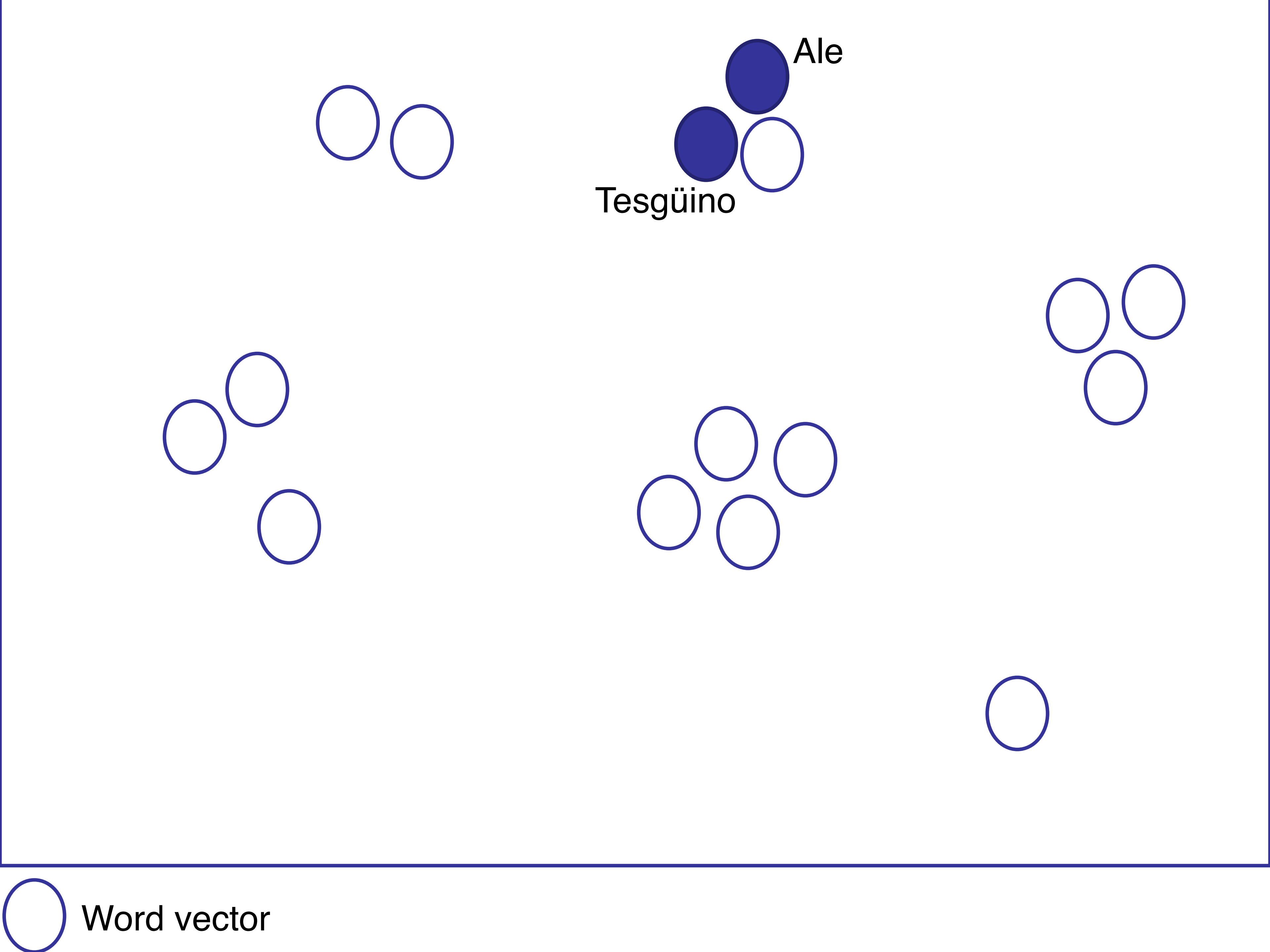
Train sample: (*Tesgüino*, *drink*)

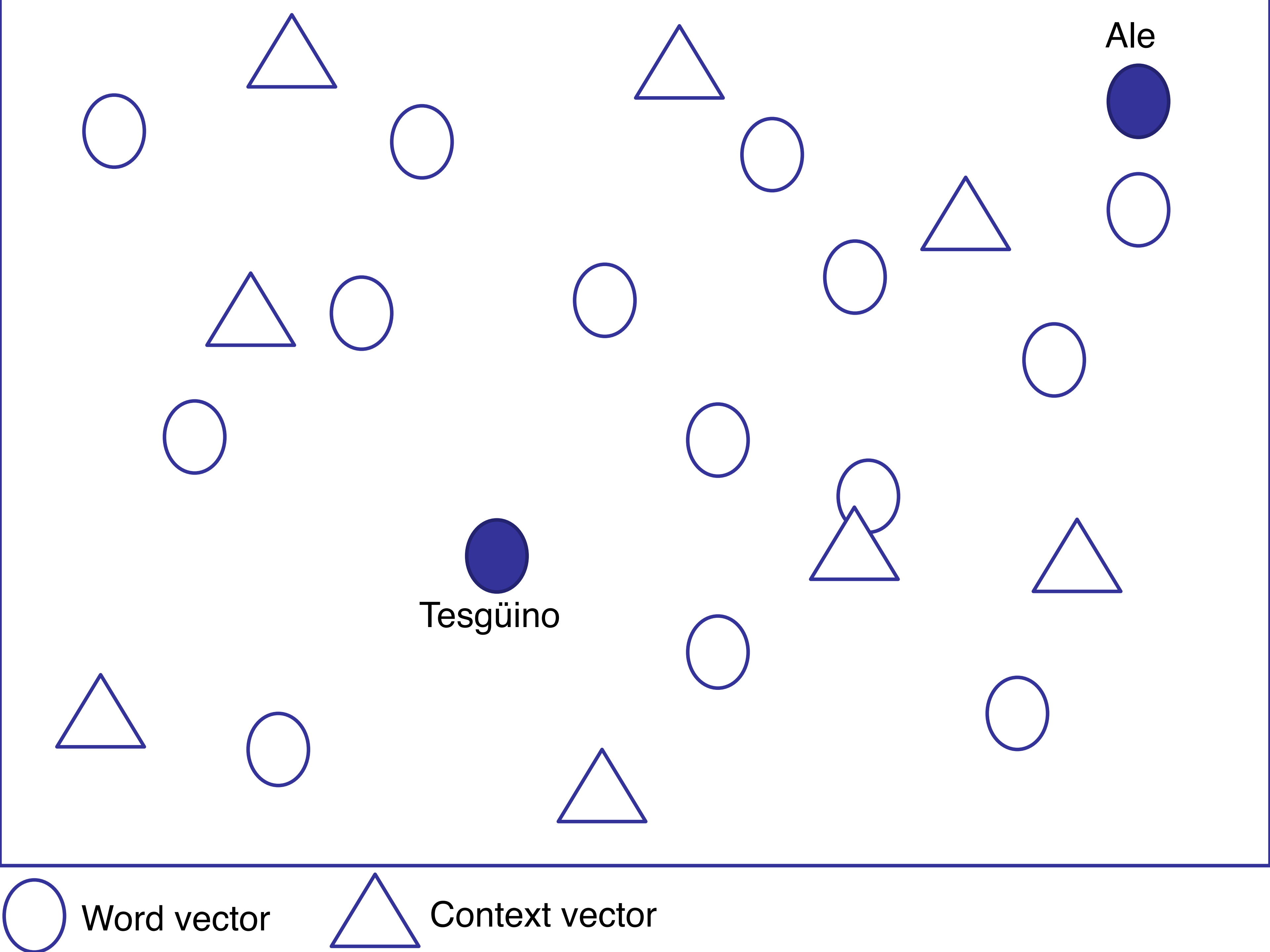


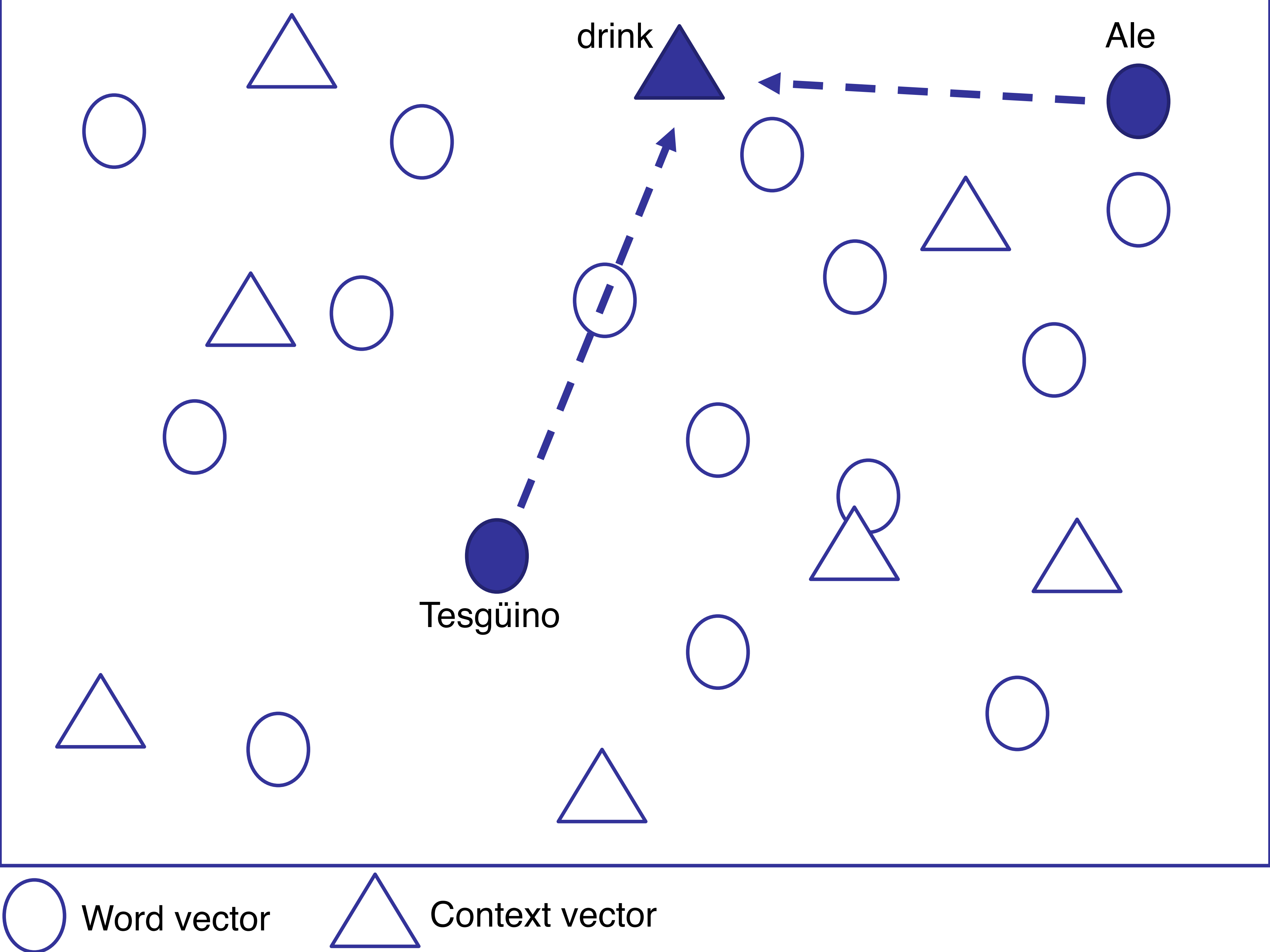
https://web.stanford.edu/~jurafsky/slp3/

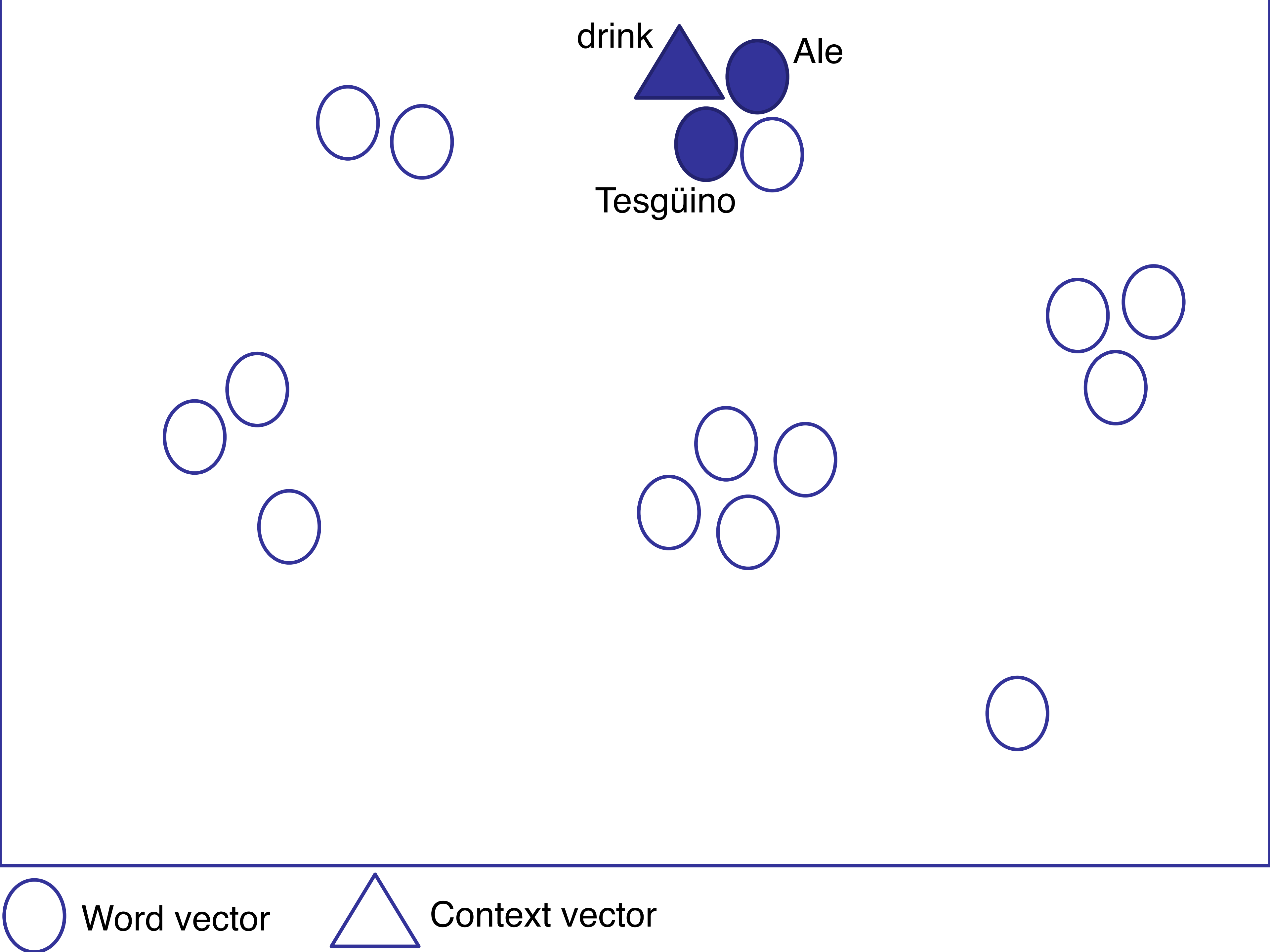


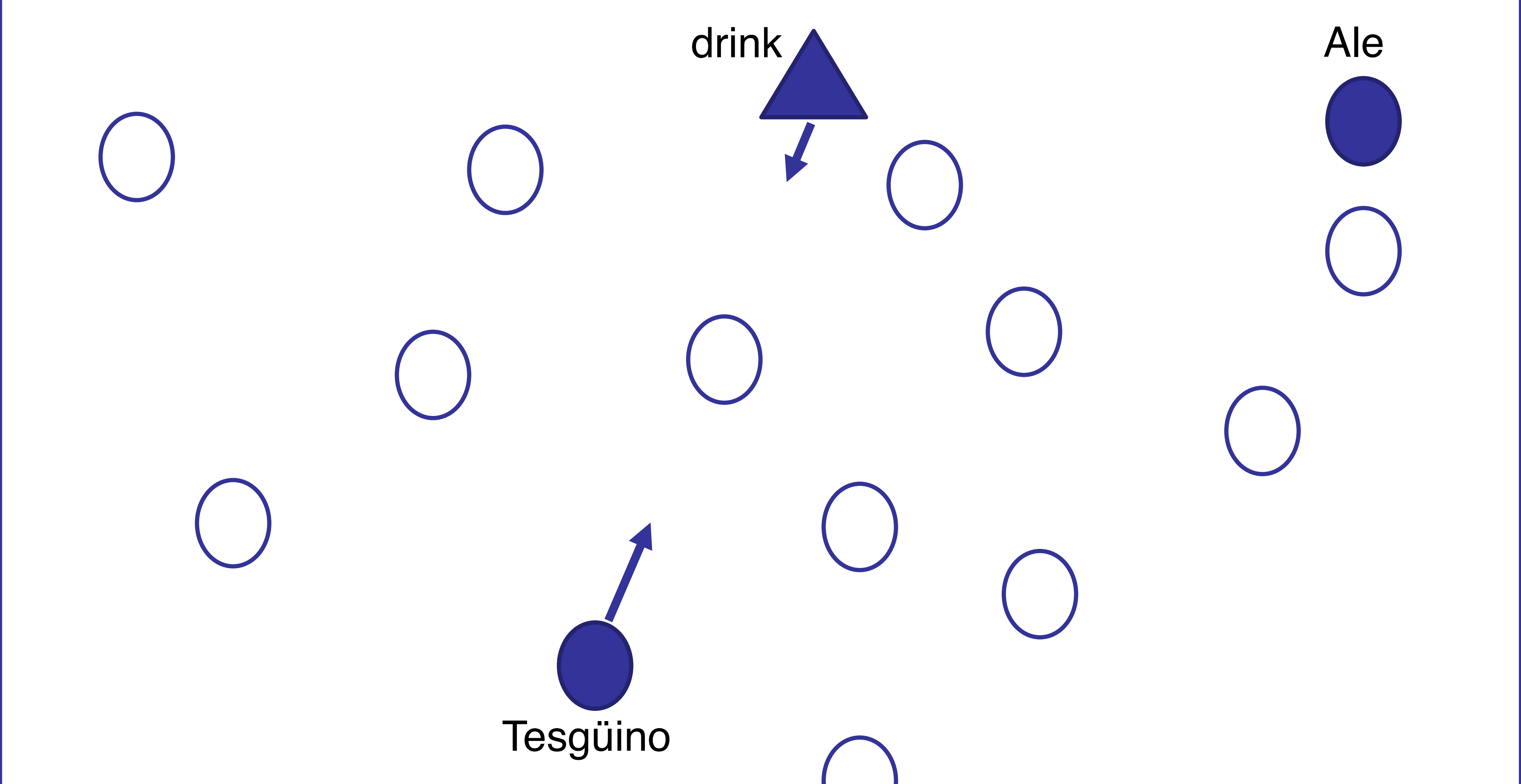
Word vector


Ale
Tesgüino
Word vector

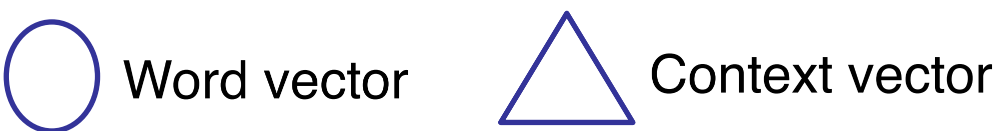

Ale
Tesgüino
Word vector
Context vector


drink
Ale
Tesgüino
Word vector
Context vector


drink
Ale
Tesgüino
Word vector
Context vector



- Train sample: *(Tesgüino, drink)*
- Update vectors to maximize $p(\mathrm{drink}|\mathrm{Tesgüino})$



Neural Word Embedding - Summary

- Output value is equal to: $\vec{a}_{\text{Tesgüino}} \cdot \vec{b}_{\text{drink}}$

- Output layer is normalized with Softmax

$$p(\text{drink}|\text{Tesgüino}) = \frac{\exp(\vec{a}_{\text{Tesgüino}} \cdot \vec{b}_{\text{drink}})}{\sum_{v \in \mathbb{V}} \exp(\vec{a}_{\text{Tesgüino}} \cdot \vec{b}_{v})}$$

$\mathbb{V}$ is the set of vocabularies

Sorry! Denominator is too expensive!

- Loss function is the Negative Log Likelihood (NLL) over all training samples T

$$L = -\frac{1}{T}\sum_{1}^{T} \log p(c|w)$$

word2vec (SkipGram) with Negative Sampling

- word2vec an efficient and effective algorithm

- Instead of $p(c|w)$, word2vec measures $p(y=1|w,c)$: the probability of genuine co-occurrence of (w,c)

$$p(y=1|w,c) = \underbrace{\sigma}_{\text{sigmoid}}(\vec{a}_w \cdot \vec{b}_c)$$

- When two words (w,c) appear in the training data, it is counted as a positive sample

- word2vec algorithm tries to distinguish between the co-occurrence probability of a positive sample from any negative sample

- To do it, word2vec draws *k* negative samples $\check{c}$ by randomly sampling from the words distribution → why randomly?

word2vec with Negative Sampling – Objective Function

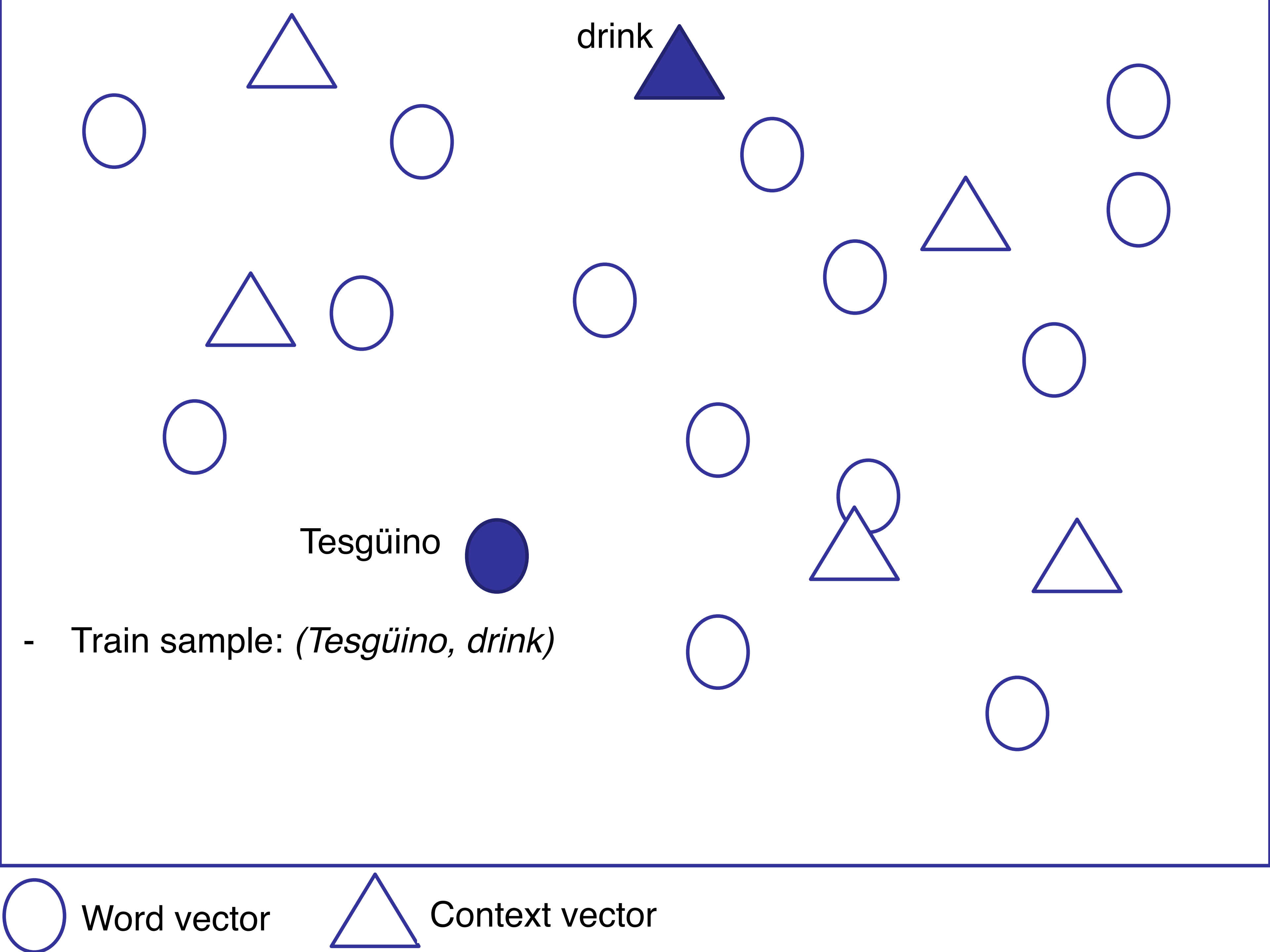
- The objective function
  - increases the probability for the positive sample (w, c)
  - decreases the probability for the k negative samples $(w, \check{c})$
- Loss function:

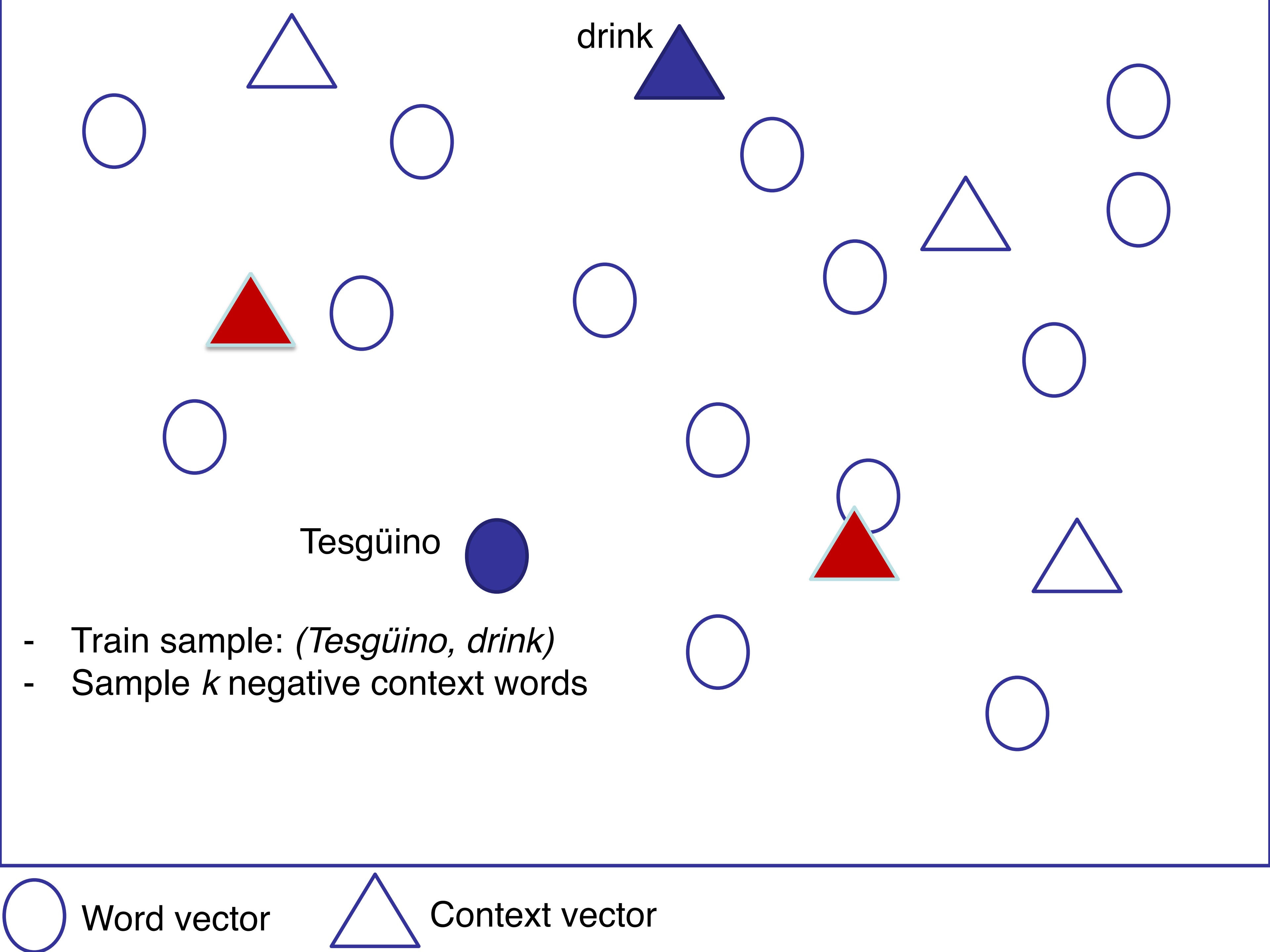
$$L = -\frac{1}{T}\sum_{1}^{T}\left[\log p(y=1|w,c) - \sum_{i=1}^{k}\log p(y=1|w,\check{c})\right]$$

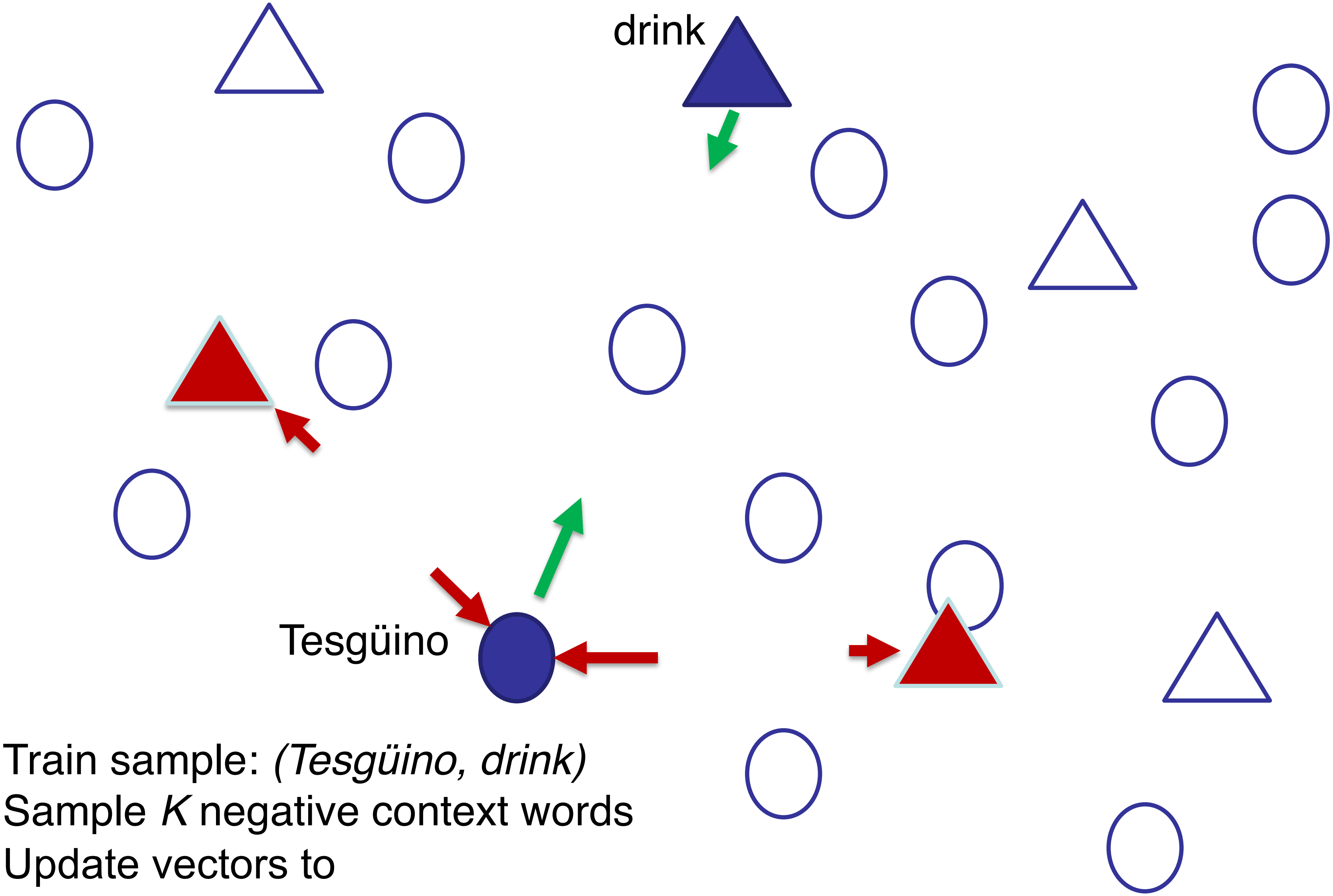
Training Samples

Negative Samples

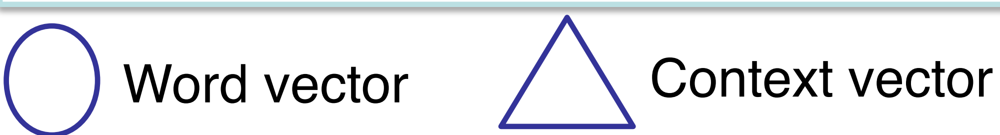
$k \sim$ 2-10


drink
Tesgüino
- Train sample: *(Tesgüino, drink)*
Word vector
Context vector


drink
Tesgüino
- Train sample: *(Tesgüino, drink)*
- Sample *k* negative context words
Word vector
Context vector



- Train sample: *(Tesgüino, drink)*
- Sample *K* negative context words
- Update vectors to
  - Maximize $p(y = 1|\text{Tesgüino}, \text{drink})$
  - Minimize $p(y = 1|\text{Tesgüino}, \check{c})$

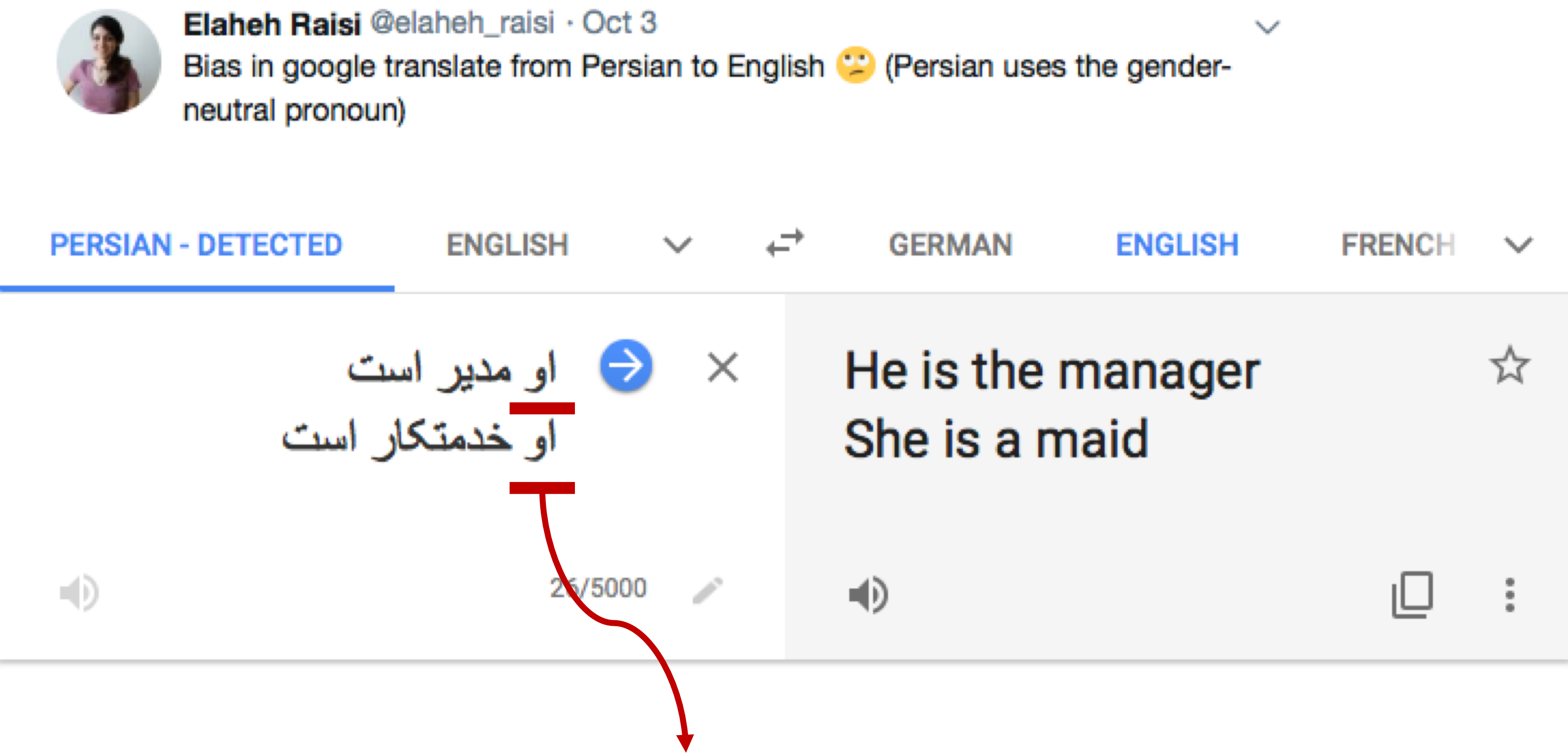


Discussion about Bias in Data

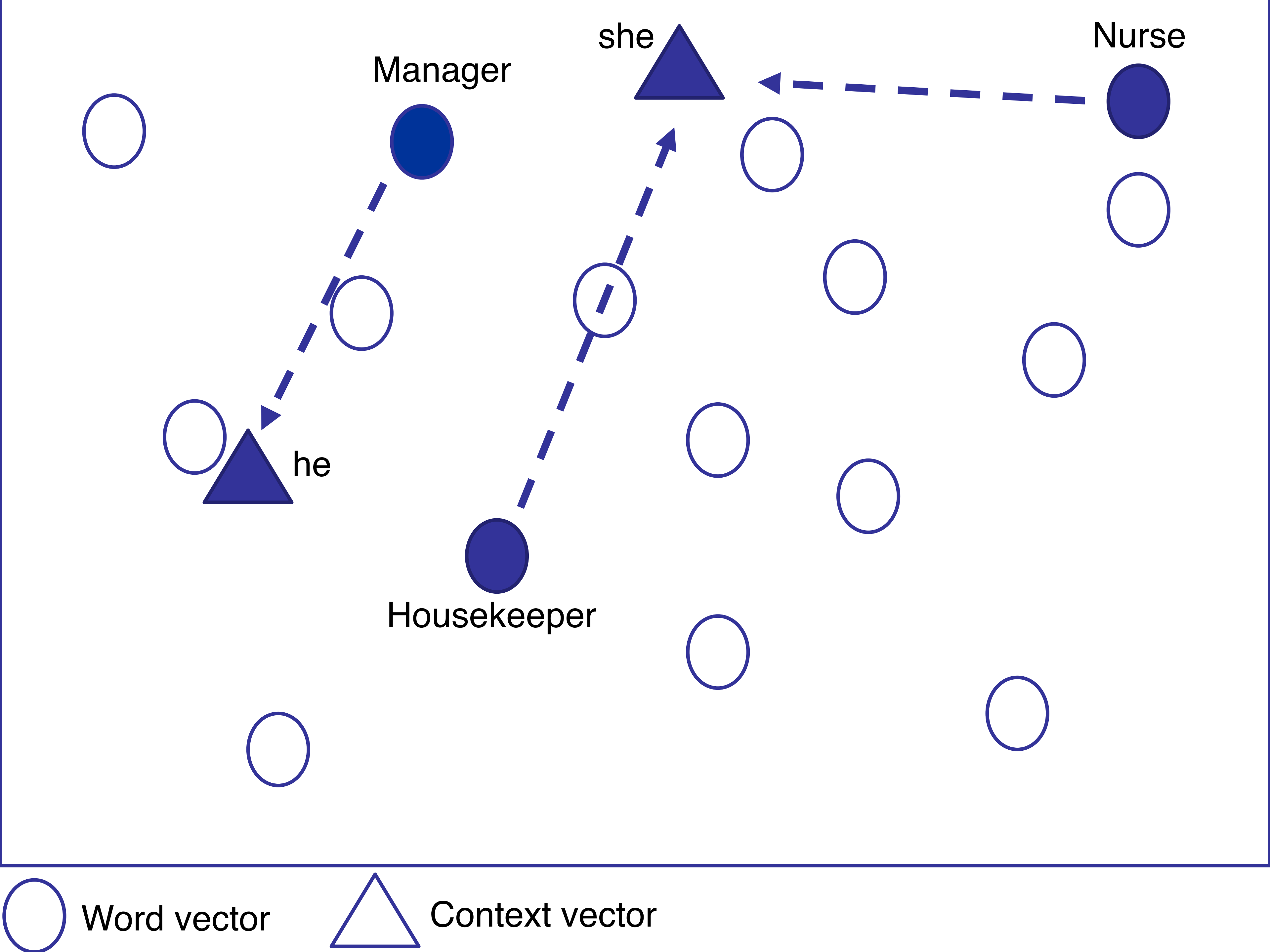
- A word embedding model captures intrinsic patterns of the given text corpus
- If the data contains (ethical) bias, the algorithm also encodes the bias in the embedding vectors
- Such bias can be propagated from word embedding to end-user NLP applications

"I think your test grading is biased in favor of students who answer the test questions correctly."

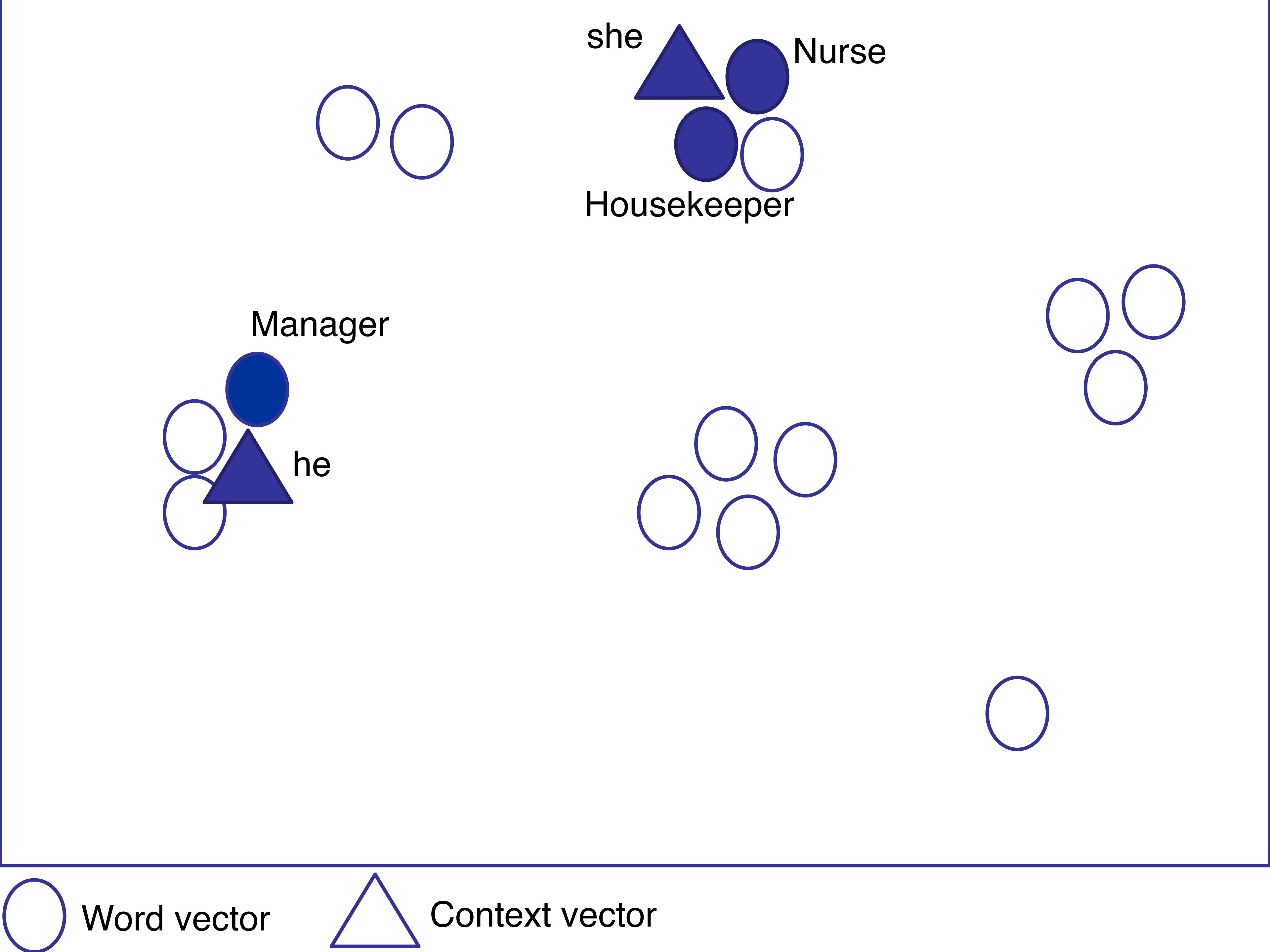
Bias in Machine Translation



same gender-neutral pronoun



she
Nurse
Manager
he
Housekeeper
Word vector
Context vector


she
Nurse
Housekeeper
Manager
he
Word vector
Context vector

Gender Bias in Wikipedia

- The bias of 350 occupations to female/male in the word2vec model, created on English Wikipedia