ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

idiap
RESEARCH INSTITUTE

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: $1 \times d \cdot d \times 1 = 1$
  - $\vec{a} \cdot W = \vec{c}$

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

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

Neural Networks

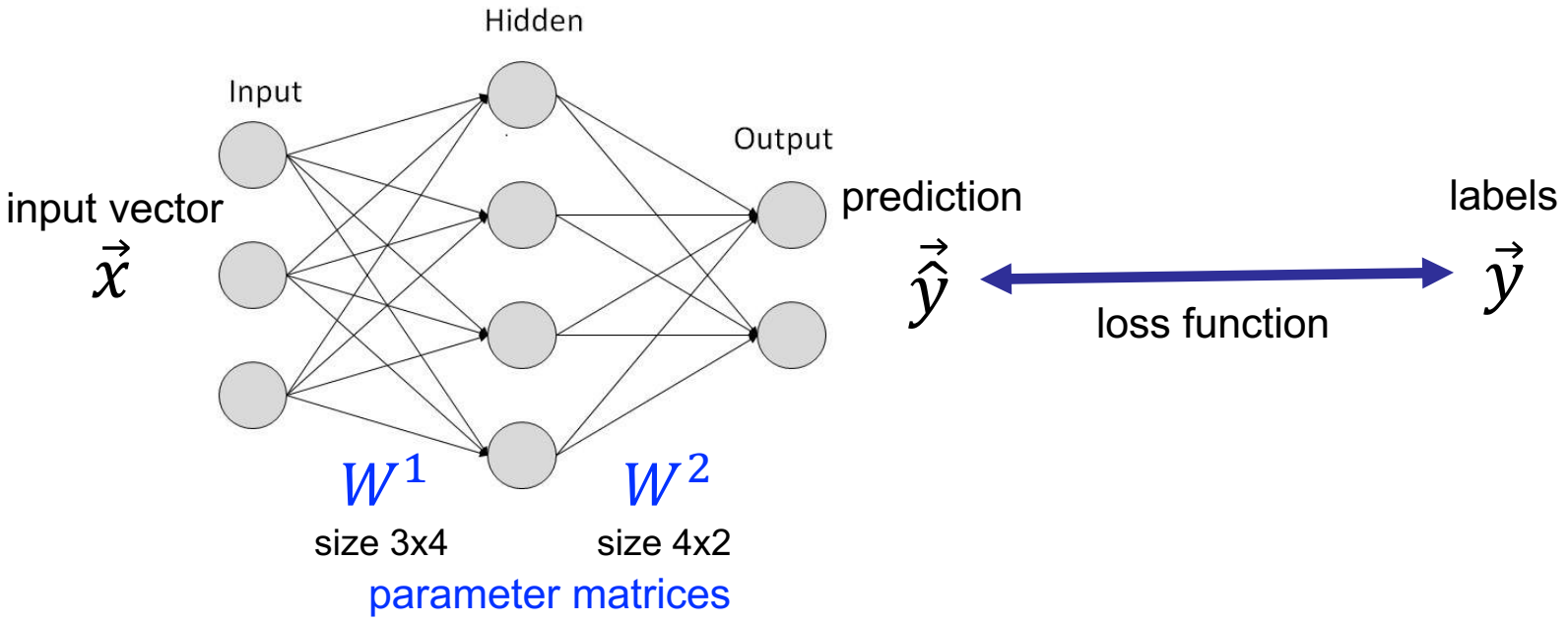
- Neural Networks are non-linear functions with many parameters

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

- 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)

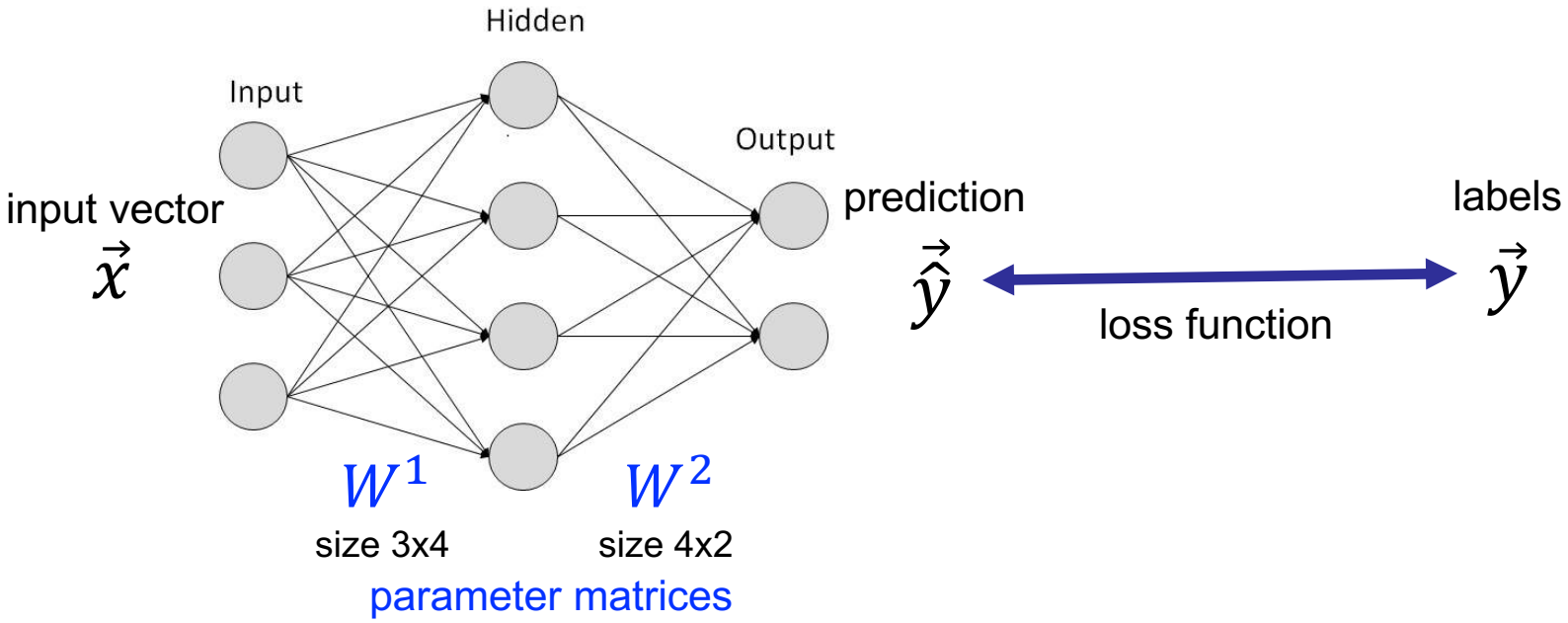


Neural Networks – Training with SGD (simplified)

Initialize parameters

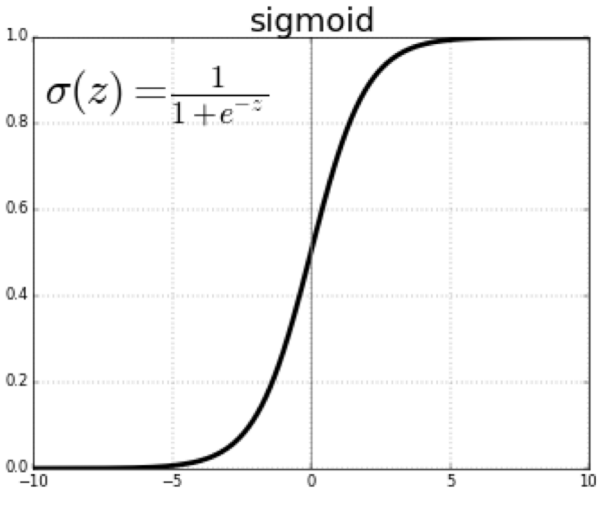
Loop over training data (or minibatches)

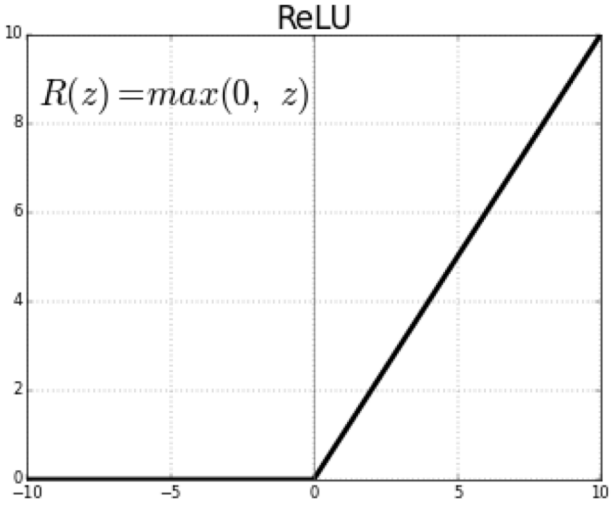
1. Do forward pass: given input $\vec{x}$ predict output $\hat{y}$
2. Calculate loss function by comparing $\hat{y}$ with labels y
3. Do backpropagation: calculate the gradient of each parameter in regard to the loss function
4. Update parameters in the direction of gradient
5. Exit if some stopping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

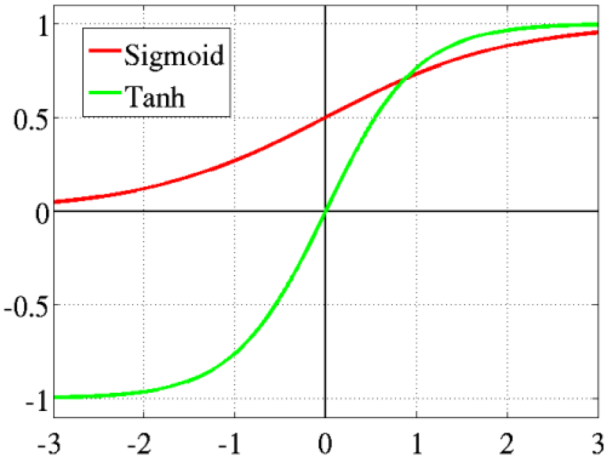


Neural Networks – Non-linearities

- 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

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

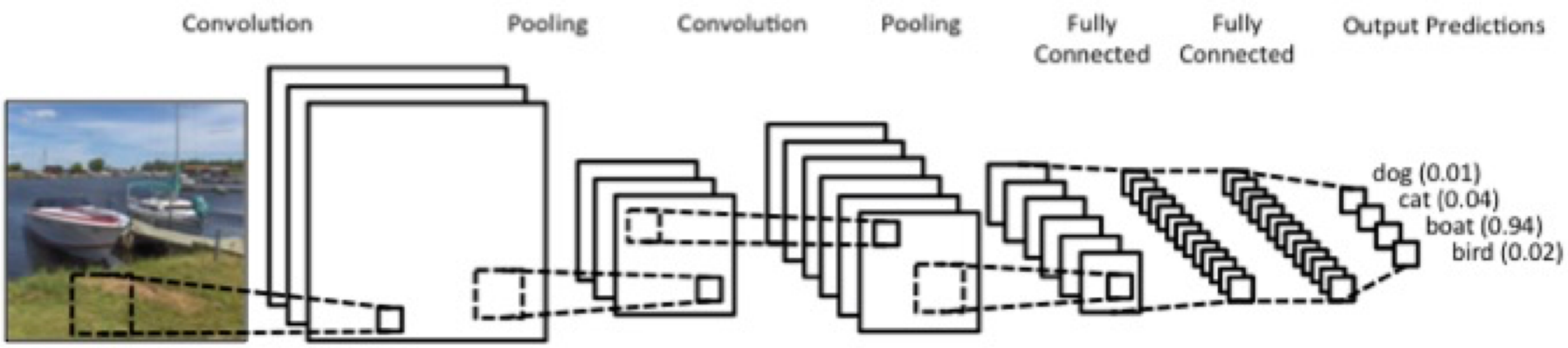
- Normally applied to the output layer and provide a probability distribution over output classes

- For example, given four classes:

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

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

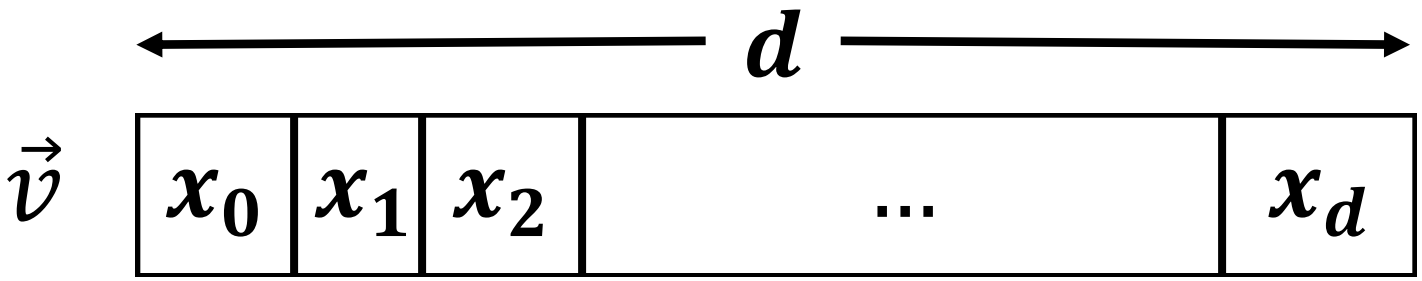
- Brief Intro to Deep Learning
  - Neural Networks
- **Word Representation Learning**
  - **Neural word representation**
  - **word2vec with Negative Sampling**
  - **Bias in word representation learning**

---Break---

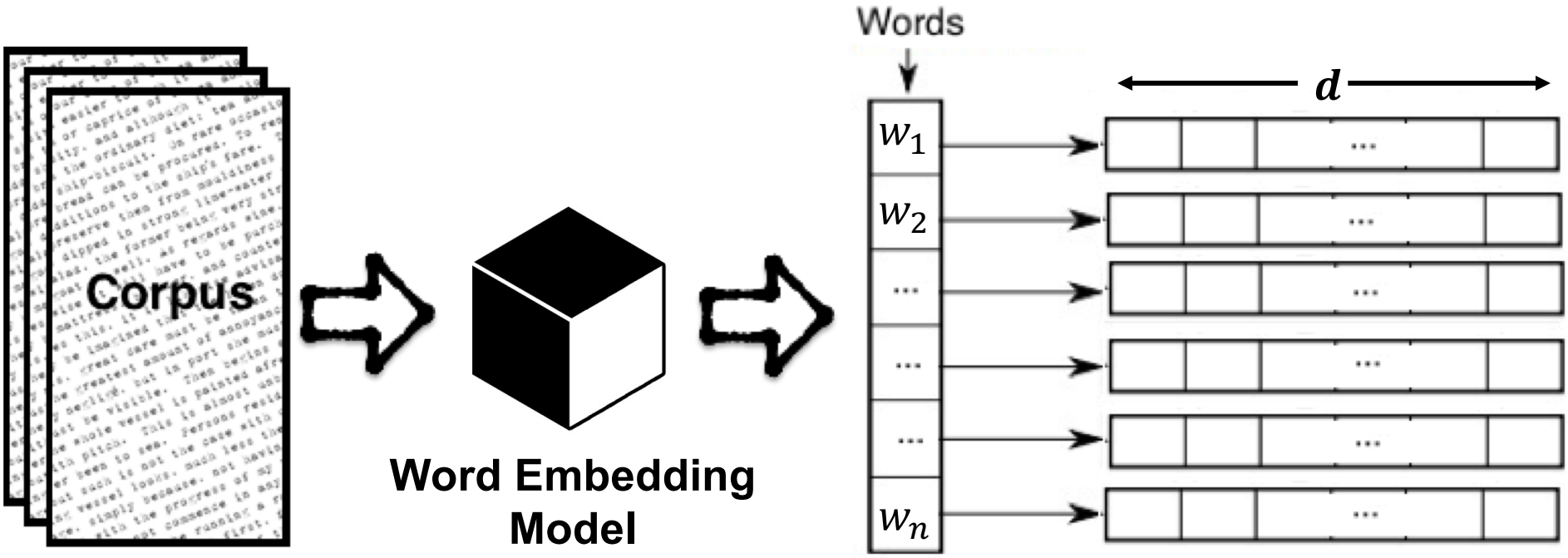
- Recurrent Neural Networks
- Attention Networks
- Document Classification with DL

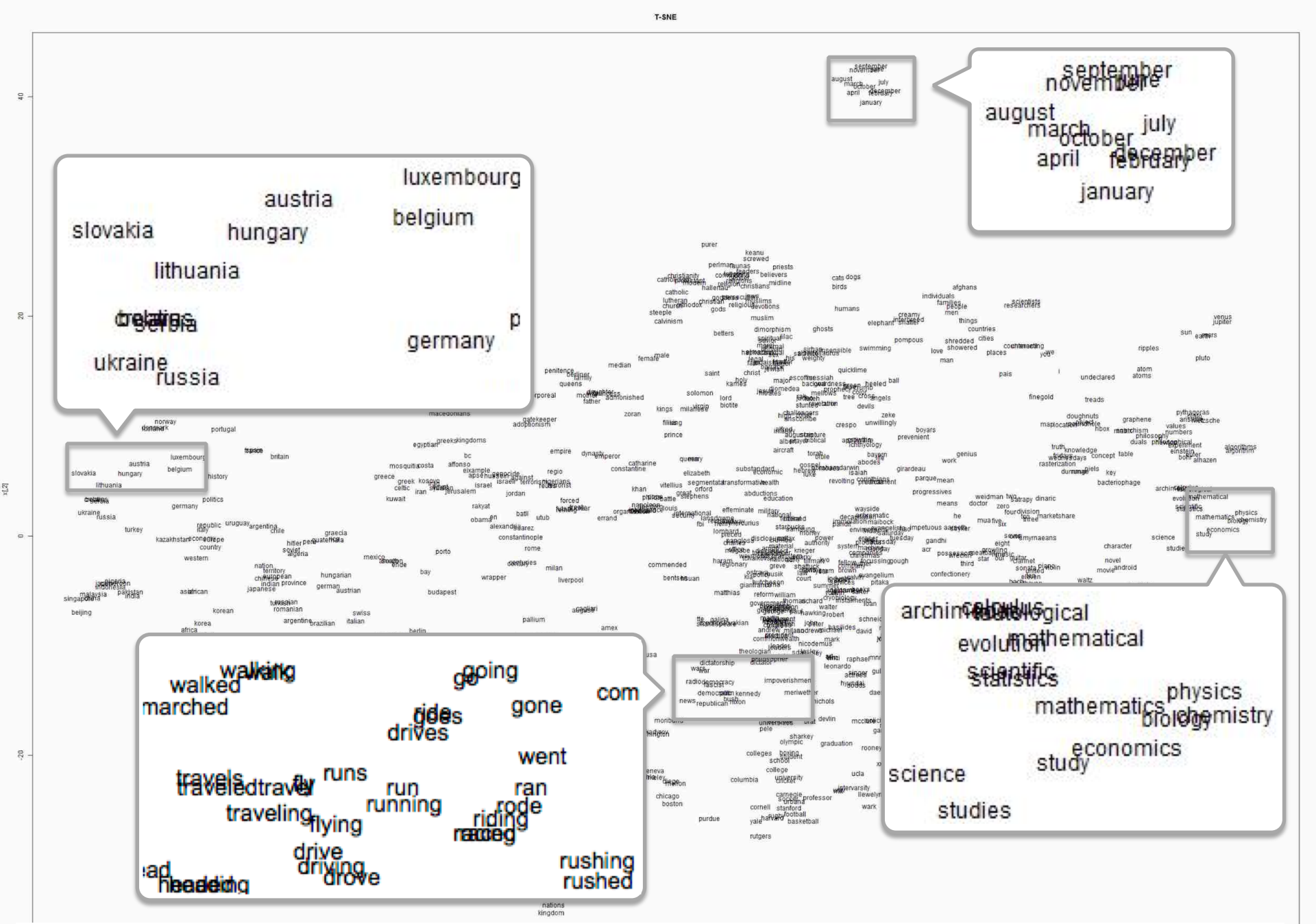
Vector Representation (Recall)

- 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

Mexico

bottle

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

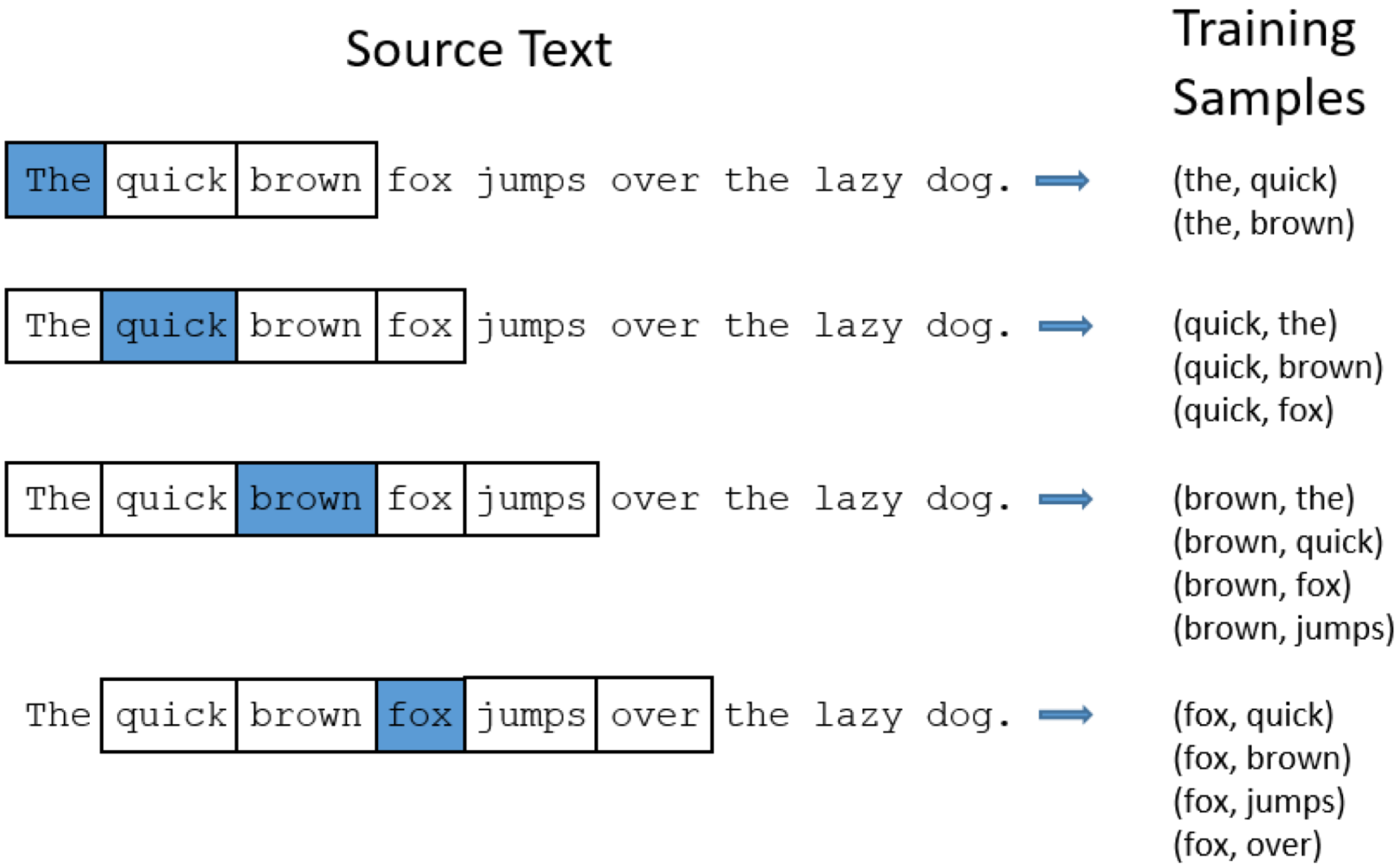
$$P(c|w)$$

   c. Optimize the network to maximize the likelihood probability
3. Repeat

Details come next!

Prepare Training Samples

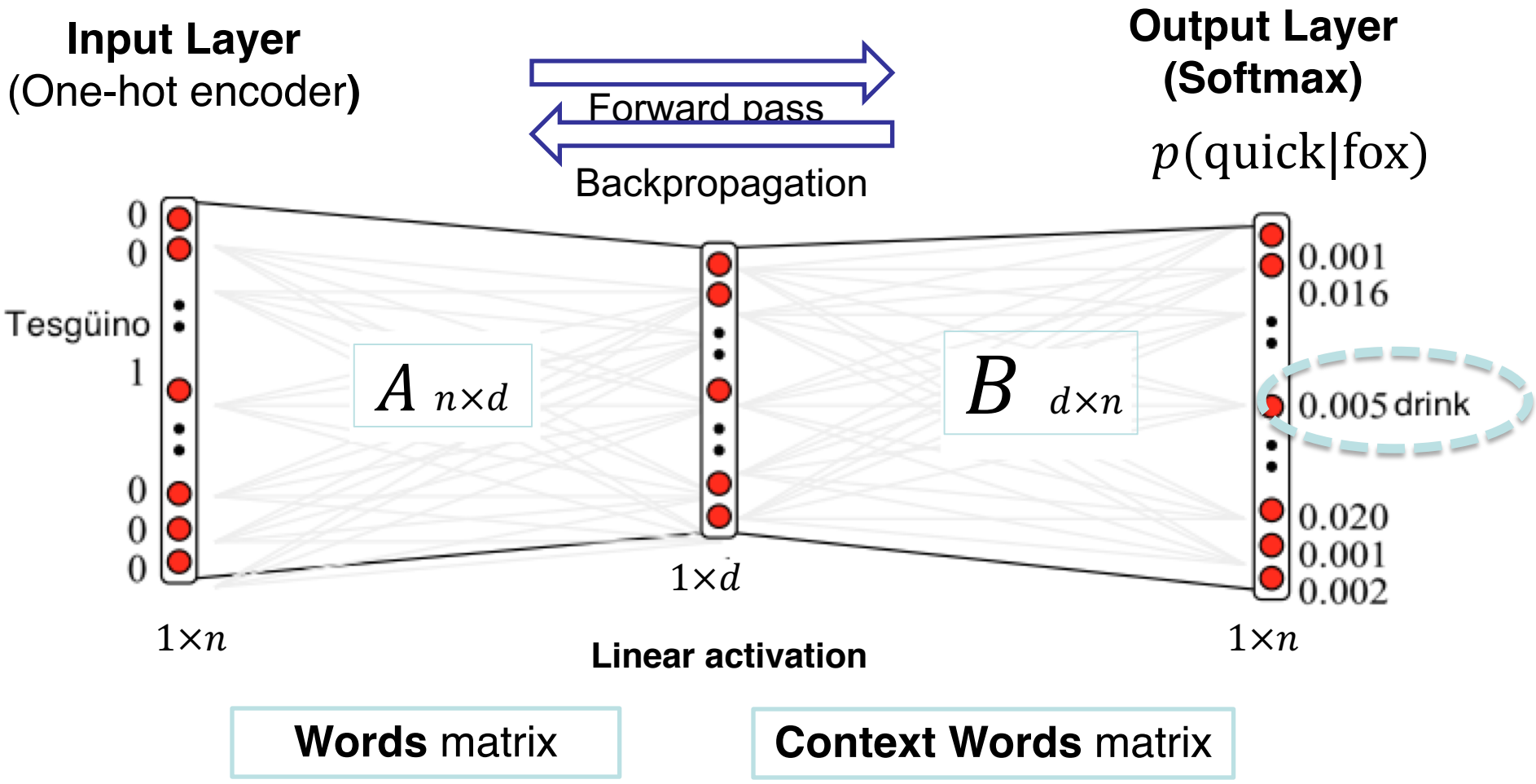
Window size of 2



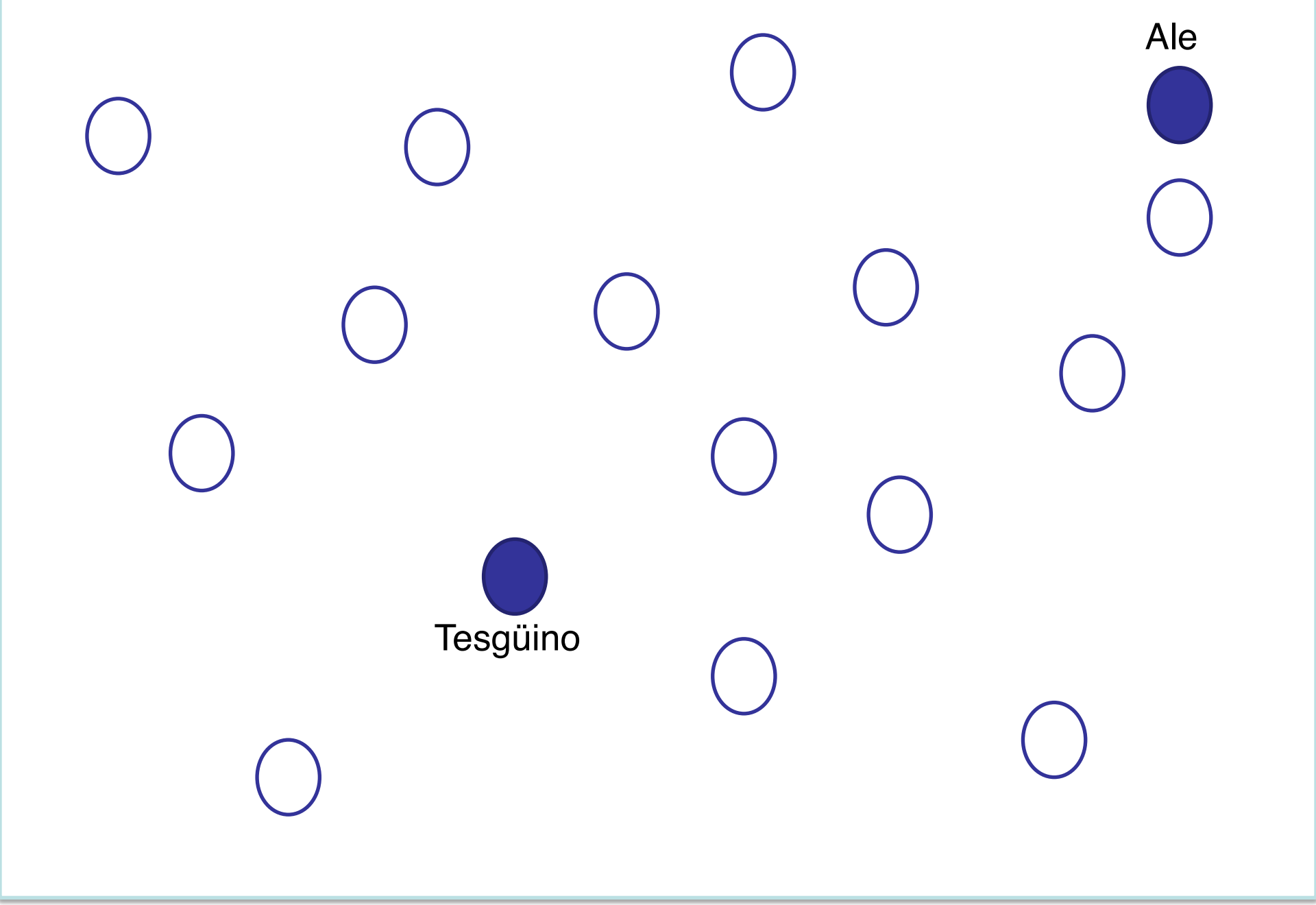
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Neural Word Embedding Architecture

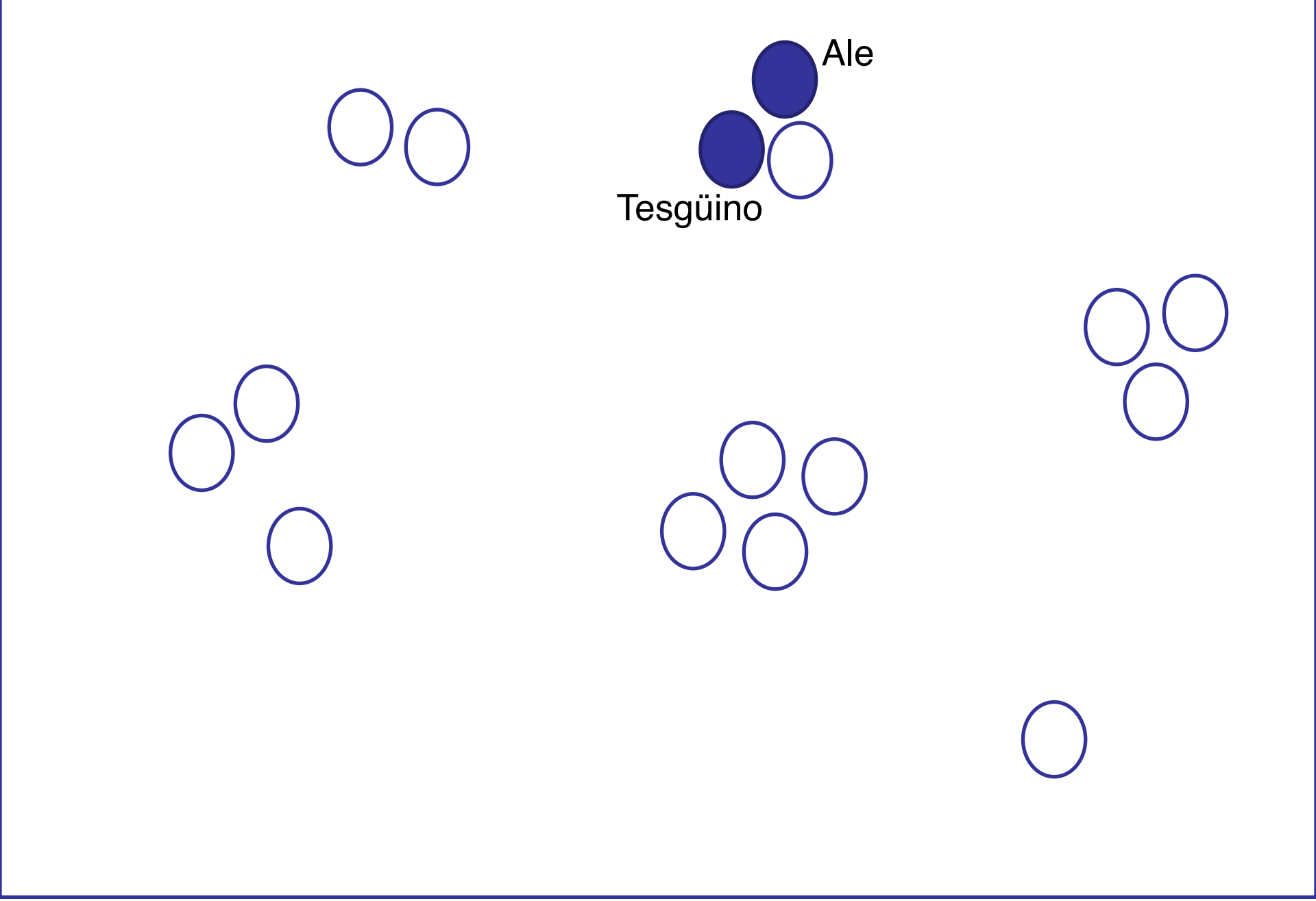
Train sample: (*Tesgüino*, *drink*)



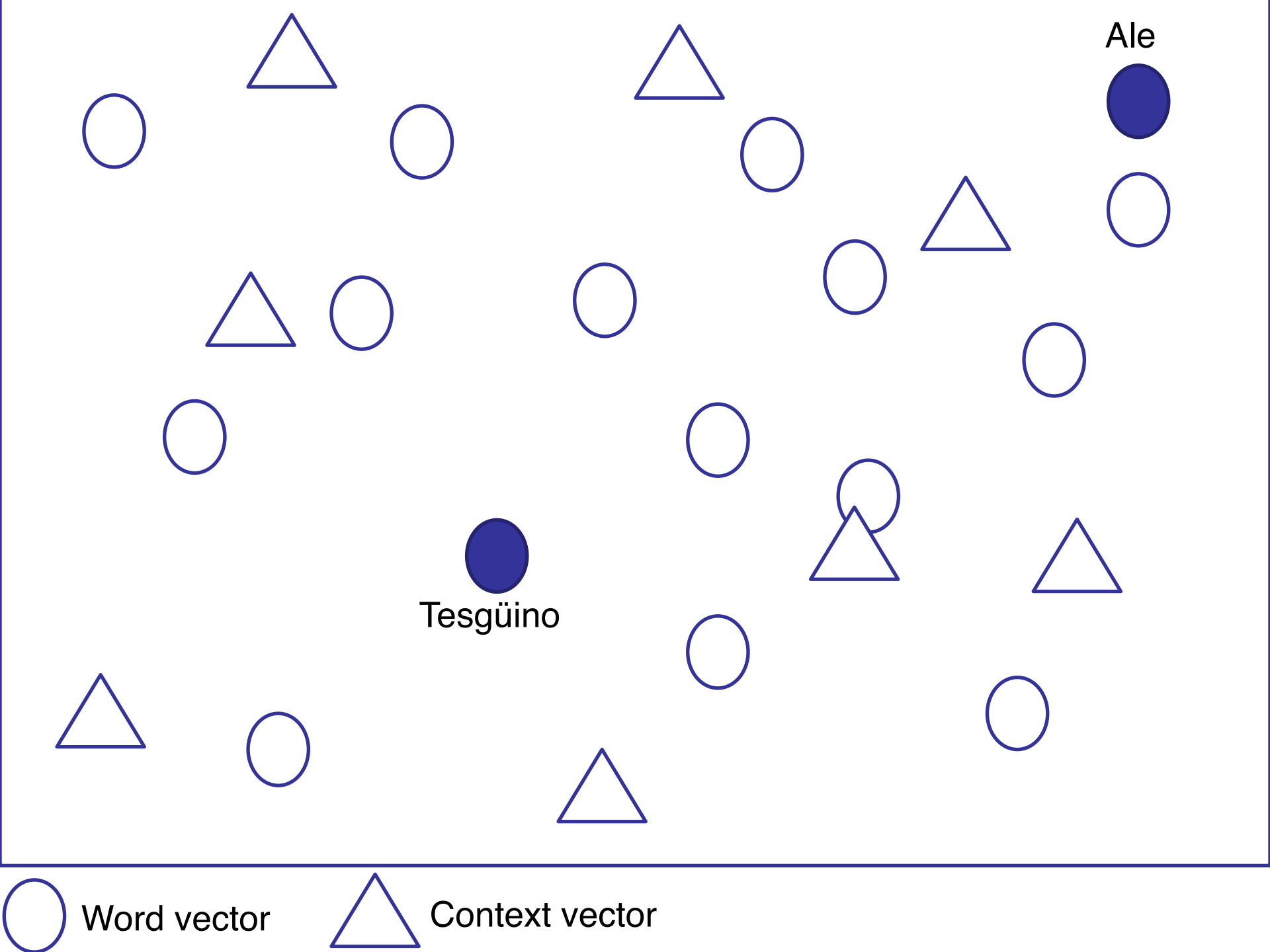
https://web.stanford.edu/~jurafsky/slp3/

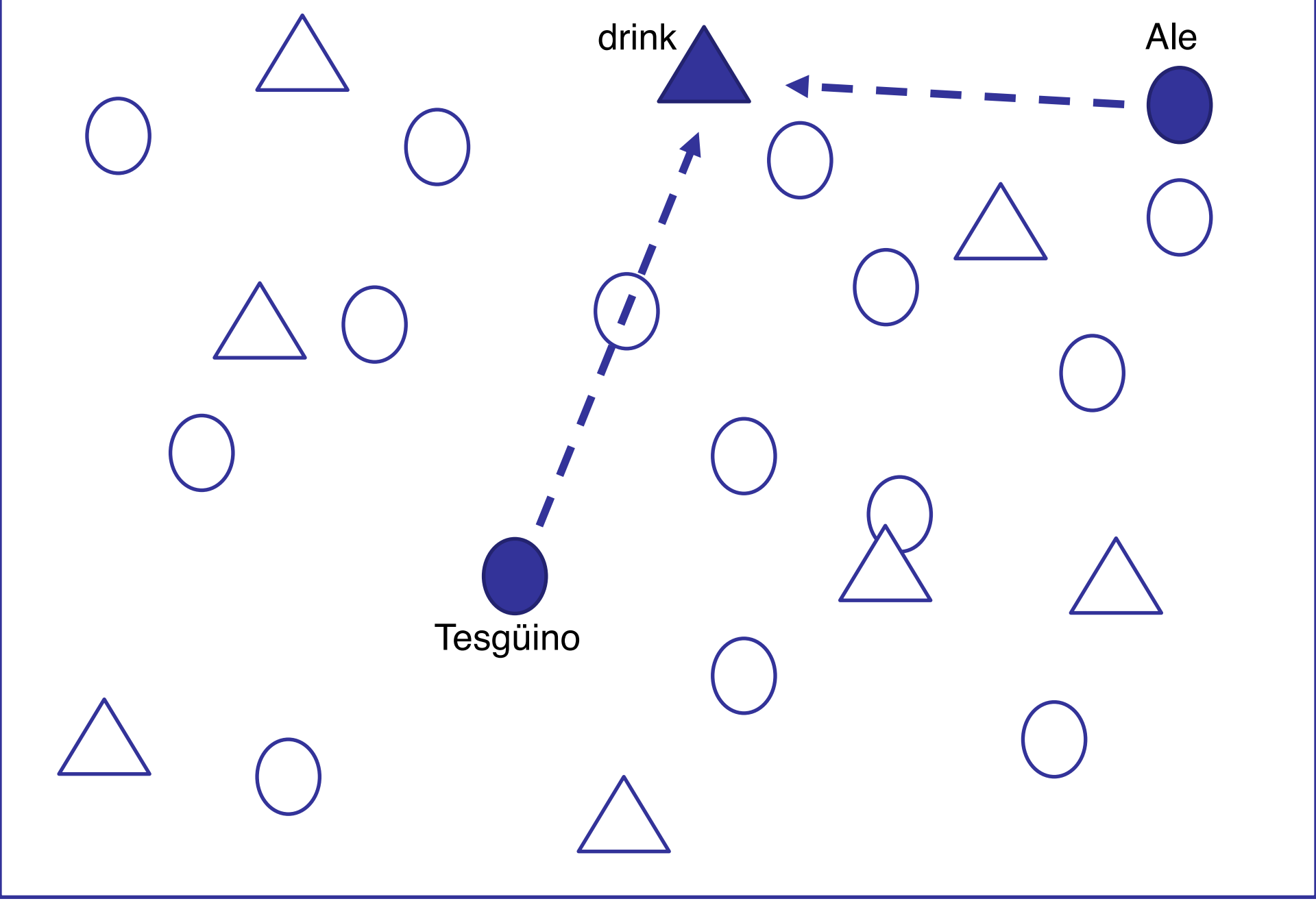


Word vector



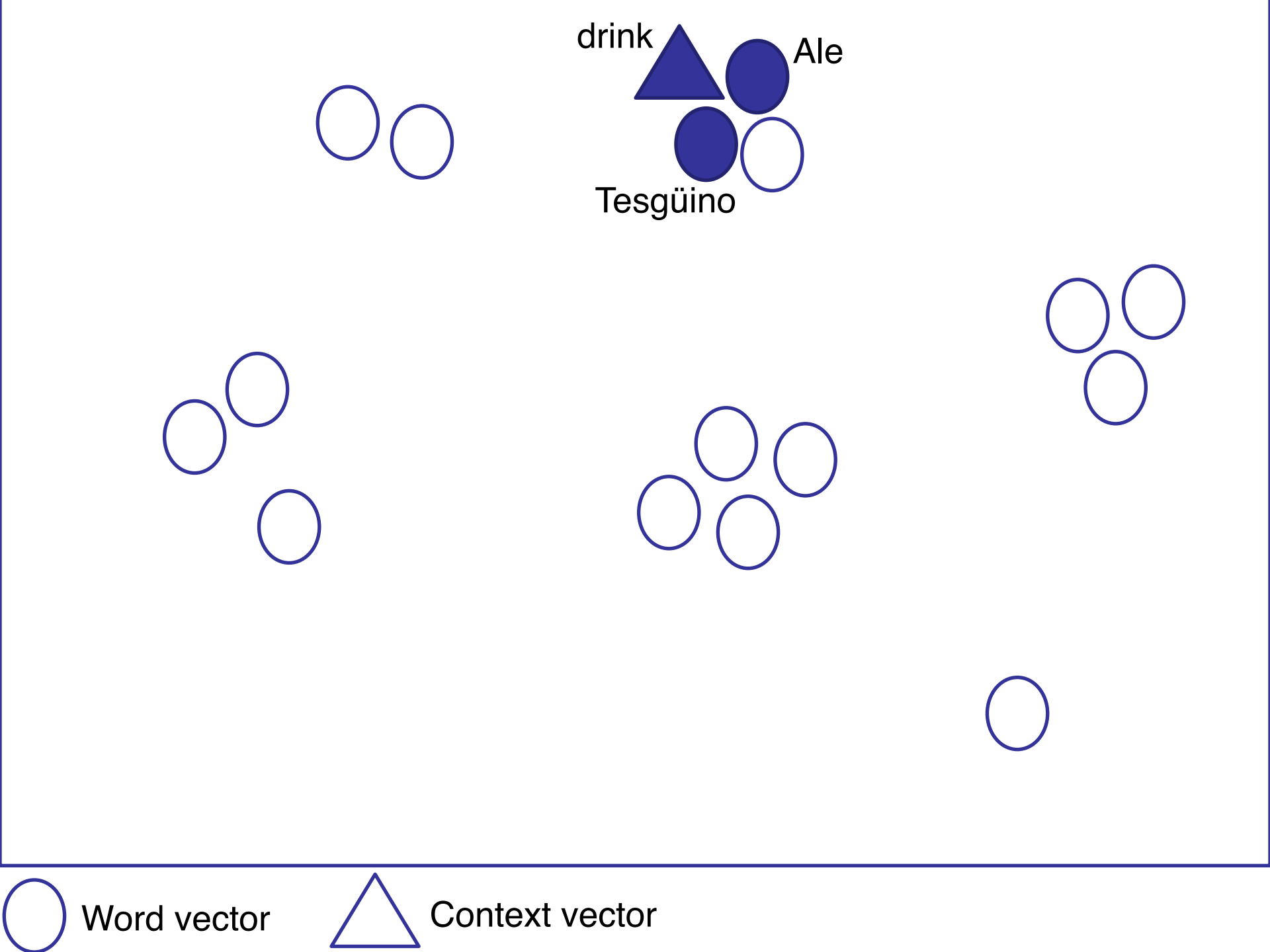
Word vector

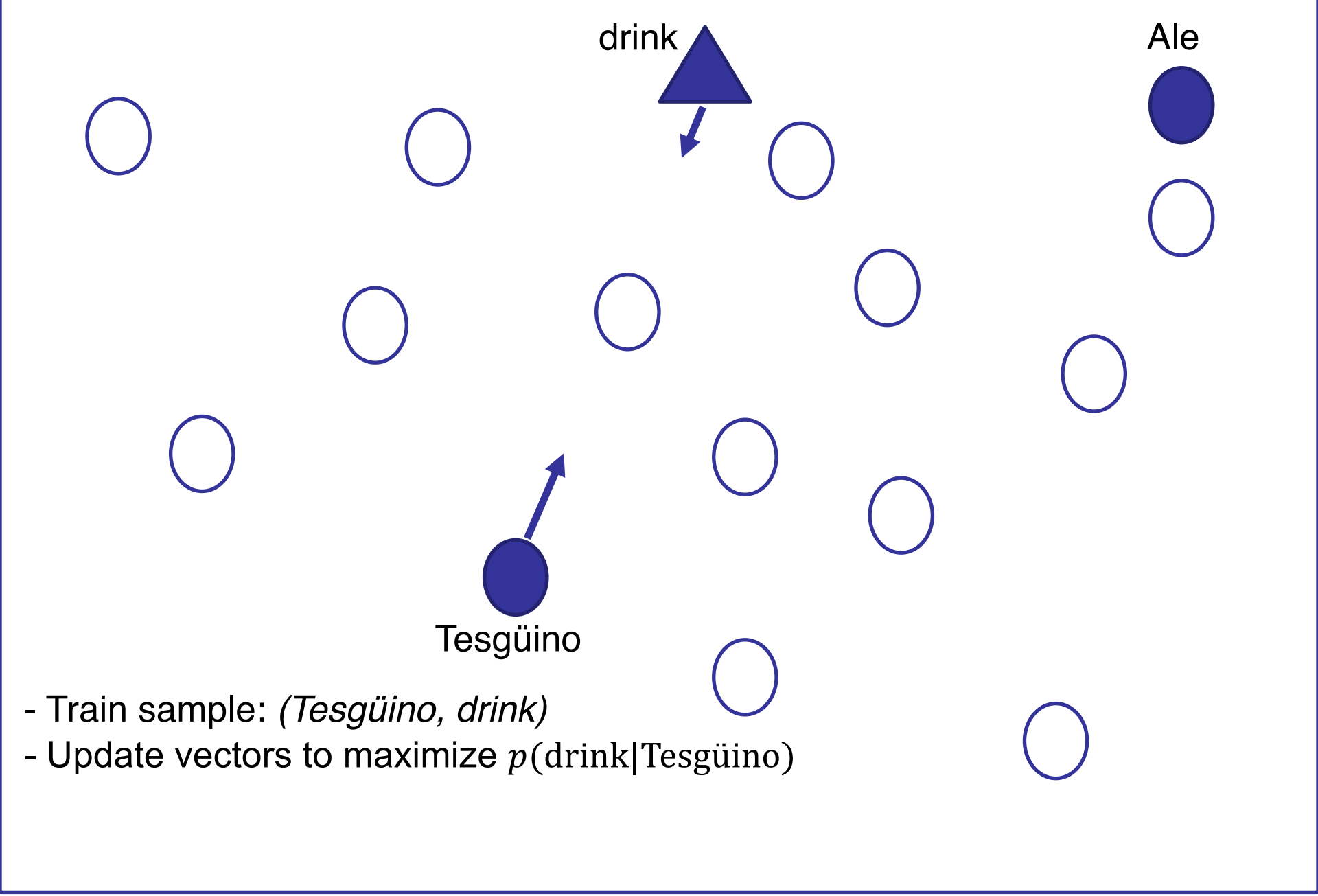

Ale
Tesgüino
Word vector
Context vector



Word vector

Context vector


drink
Ale
Tesgüino
Word vector
Context vector



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

Word vector

Context vector

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)

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

sigmoid

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

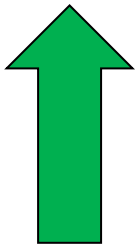
- word2vec algorithm tries to distinguish between the co-occurrence probability of a positive sample from any negative sample

- 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

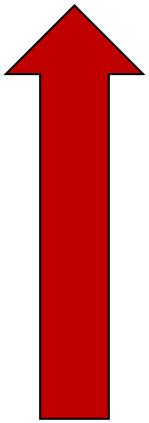
- The objective function
  - increases the probability for the positive sample (w, c)
  - decreases the probability for the k negative samples $(w, \check{c})$
- Loss function:

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

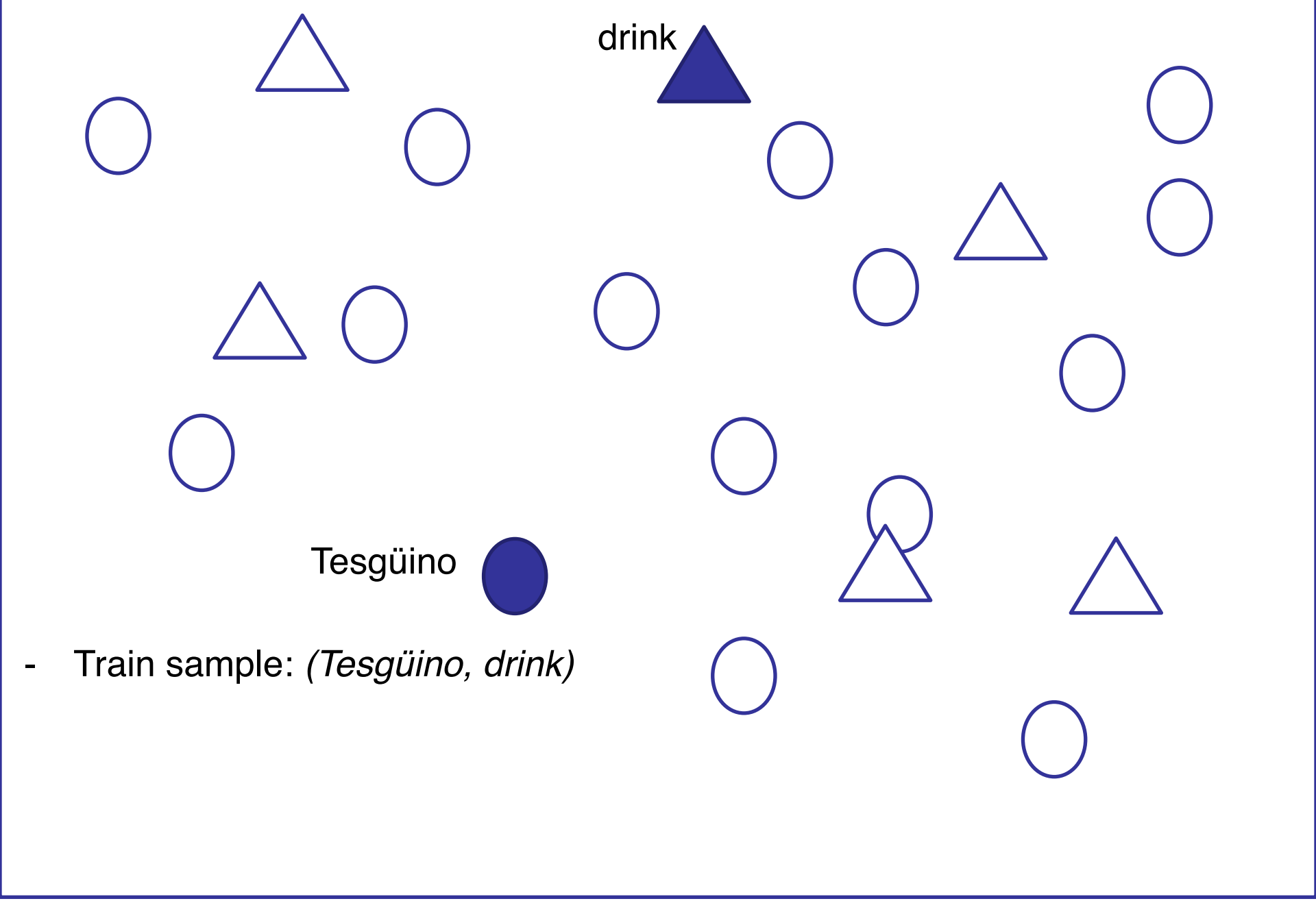


Training Samples

$k \sim$ 2-10



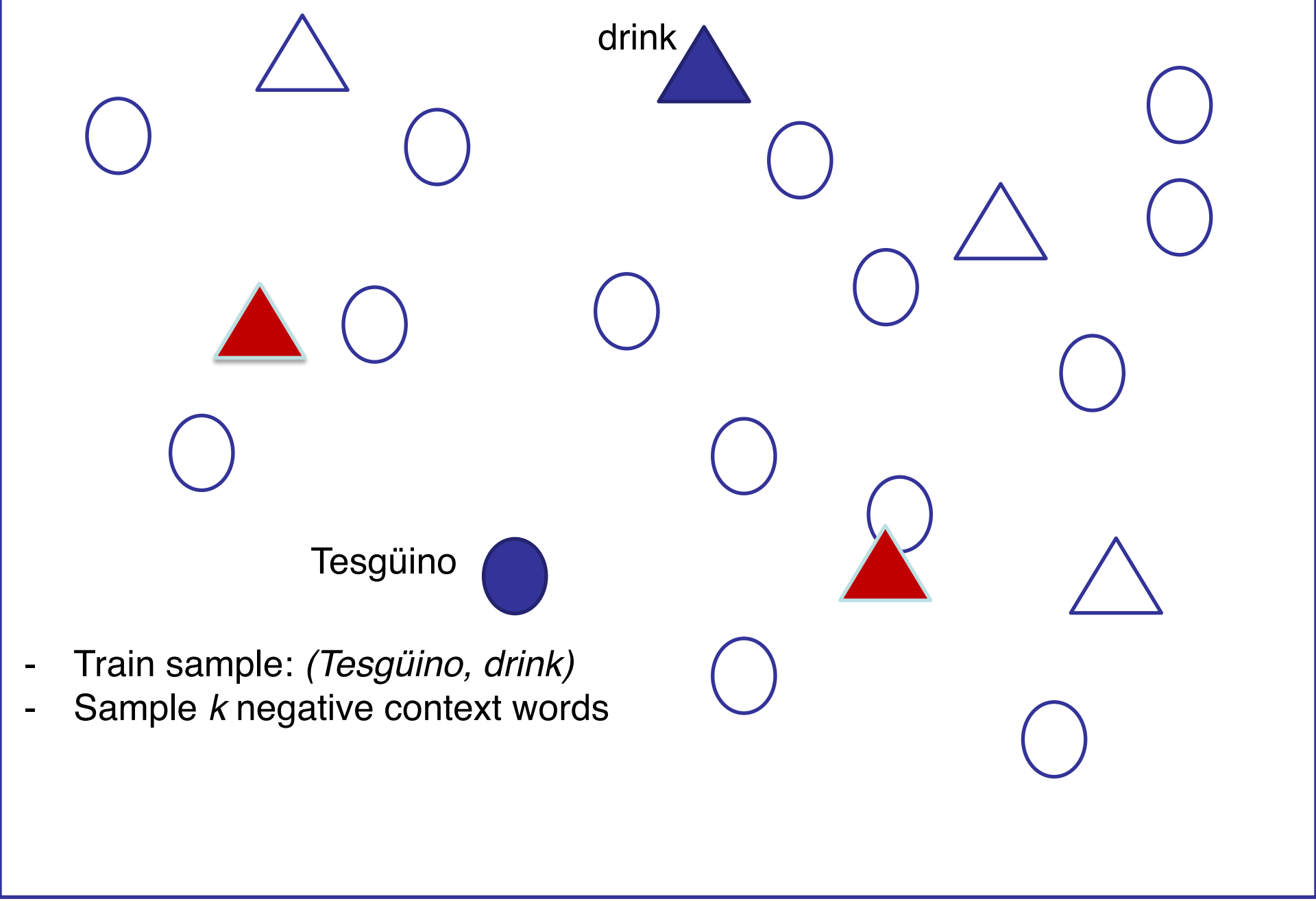
Negative Samples



- Train sample: *(Tesgüino, drink)*

Word vector

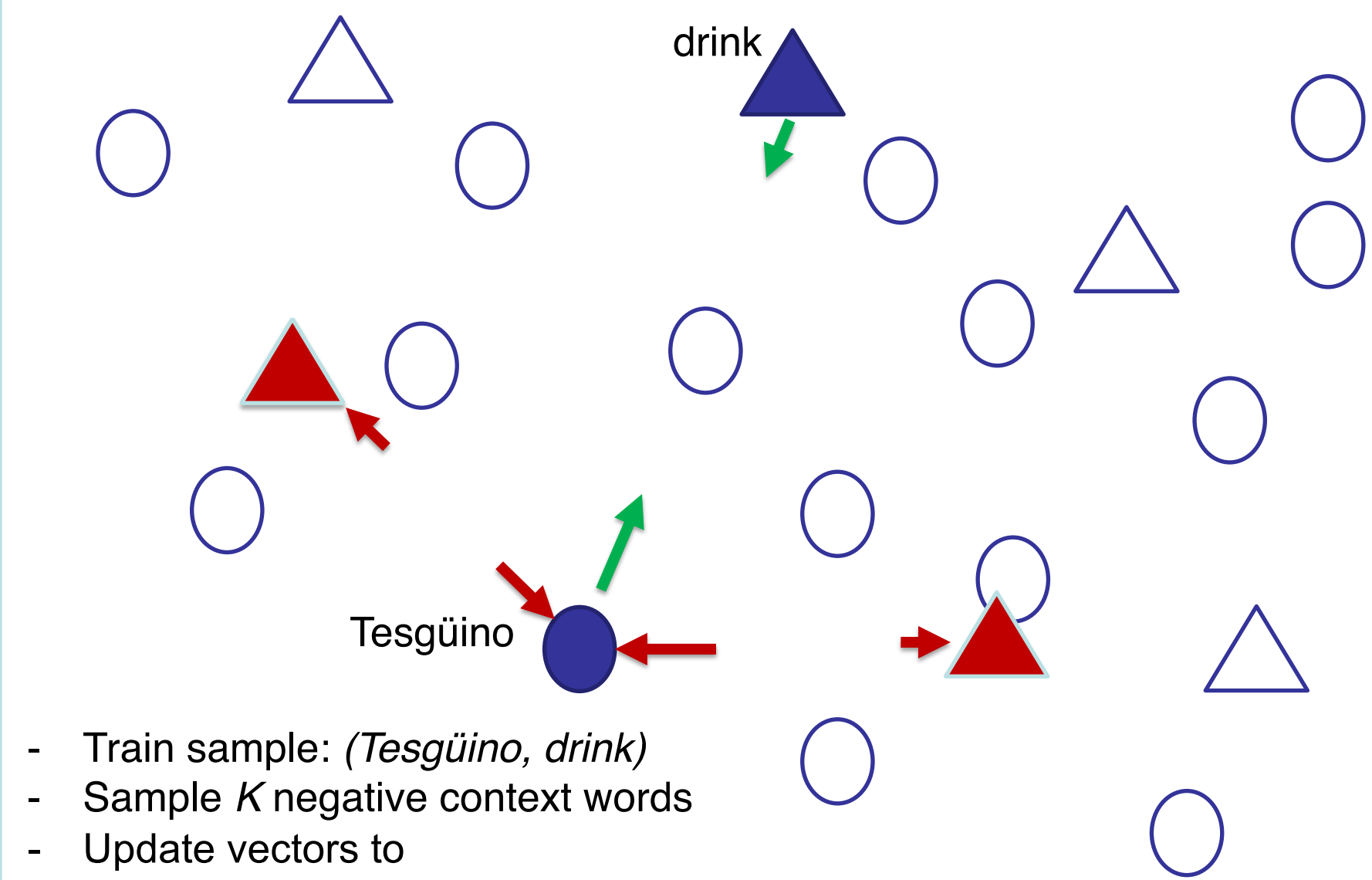
Context vector



- 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 $p(y = 1|\text{Tesgüino}, \check{c})$

Word vector

Context vector

Discussion about Bias in Data

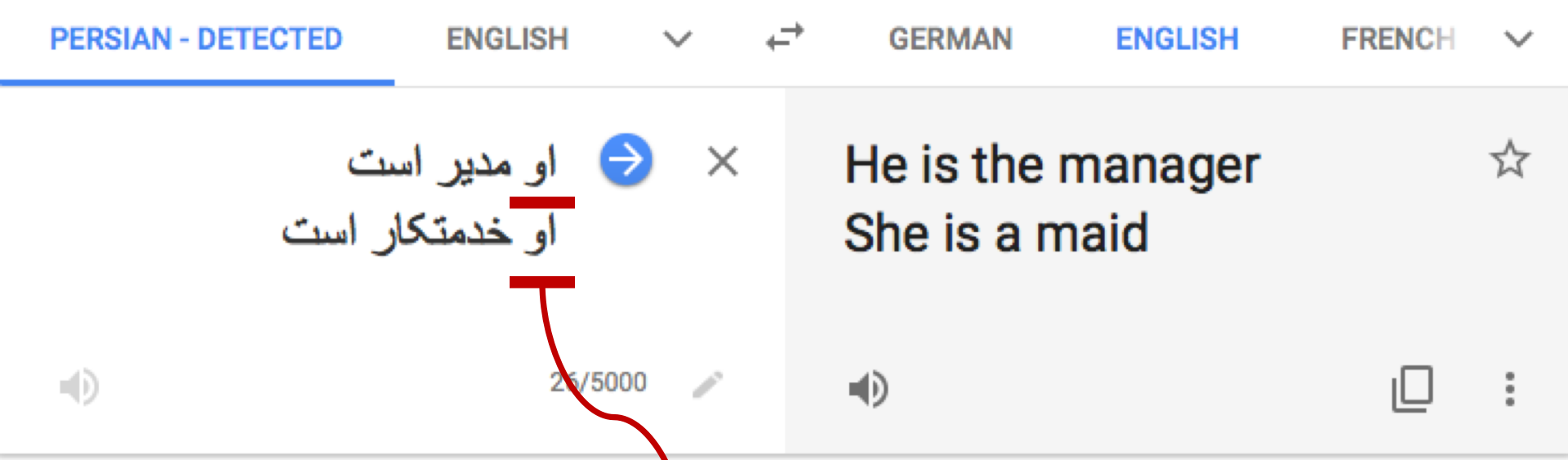
- A word embedding model captures intrinsic patterns of the given text corpus
- If the data contains (ethical) bias, the algorithm also encodes the bias in the embedding vectors
- Such bias can be propagated from word embedding to end-user NLP applications

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

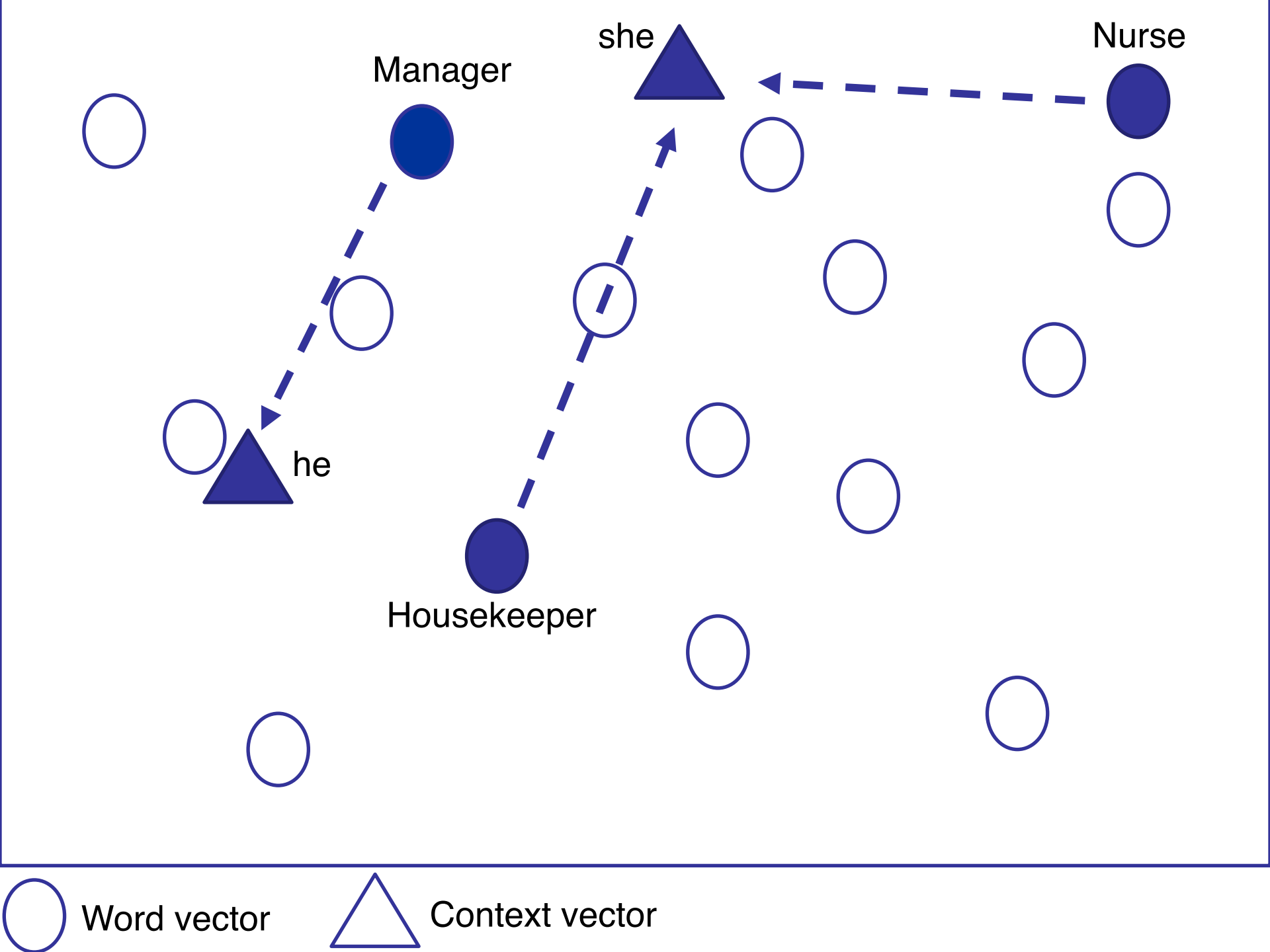
Bias in Machine Translation

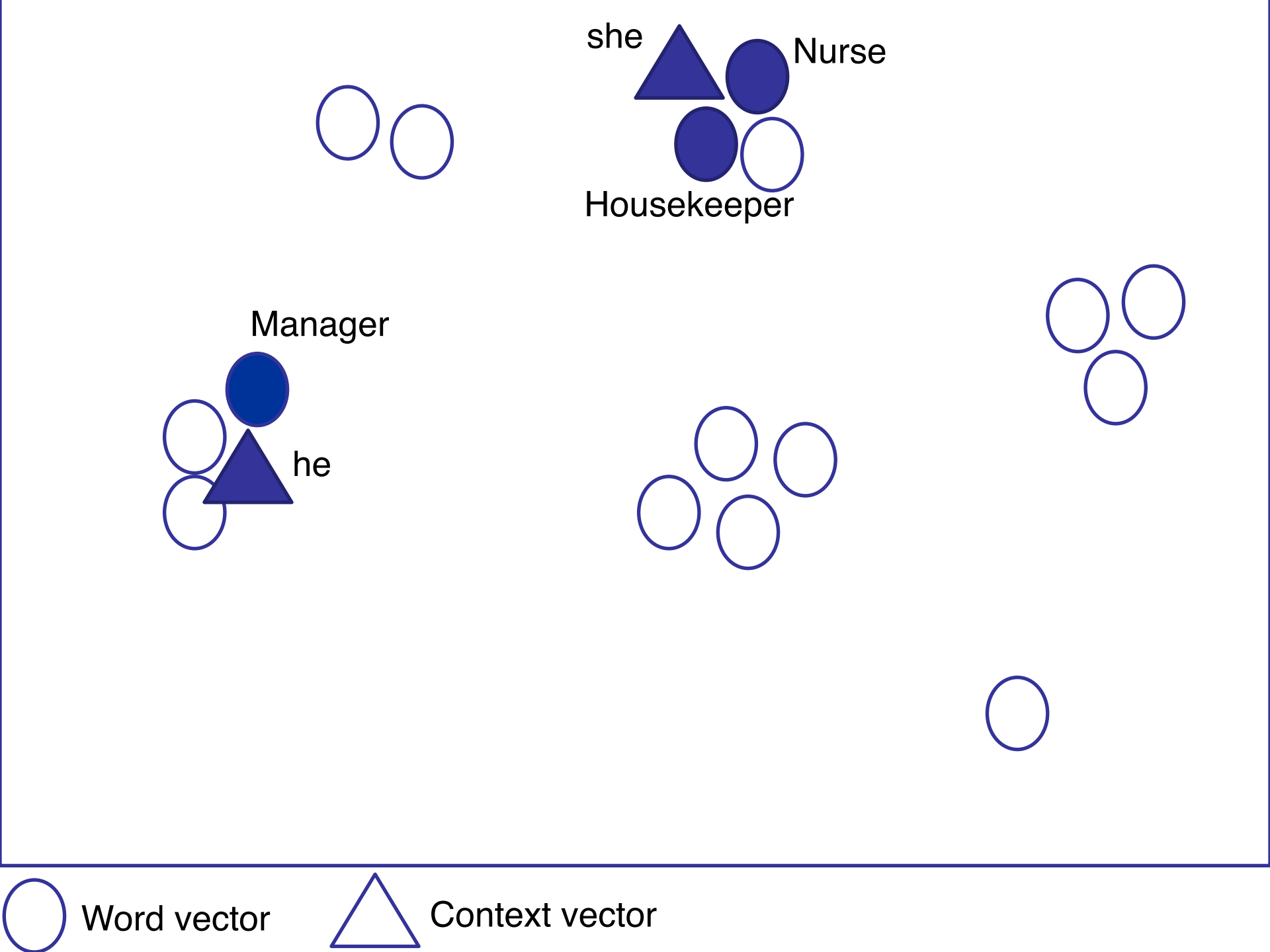
Elaheh Raisi @elaheh_raisi · Oct 3

Bias in google translate from Persian to English 😕 (Persian uses the gender-neutral pronoun)



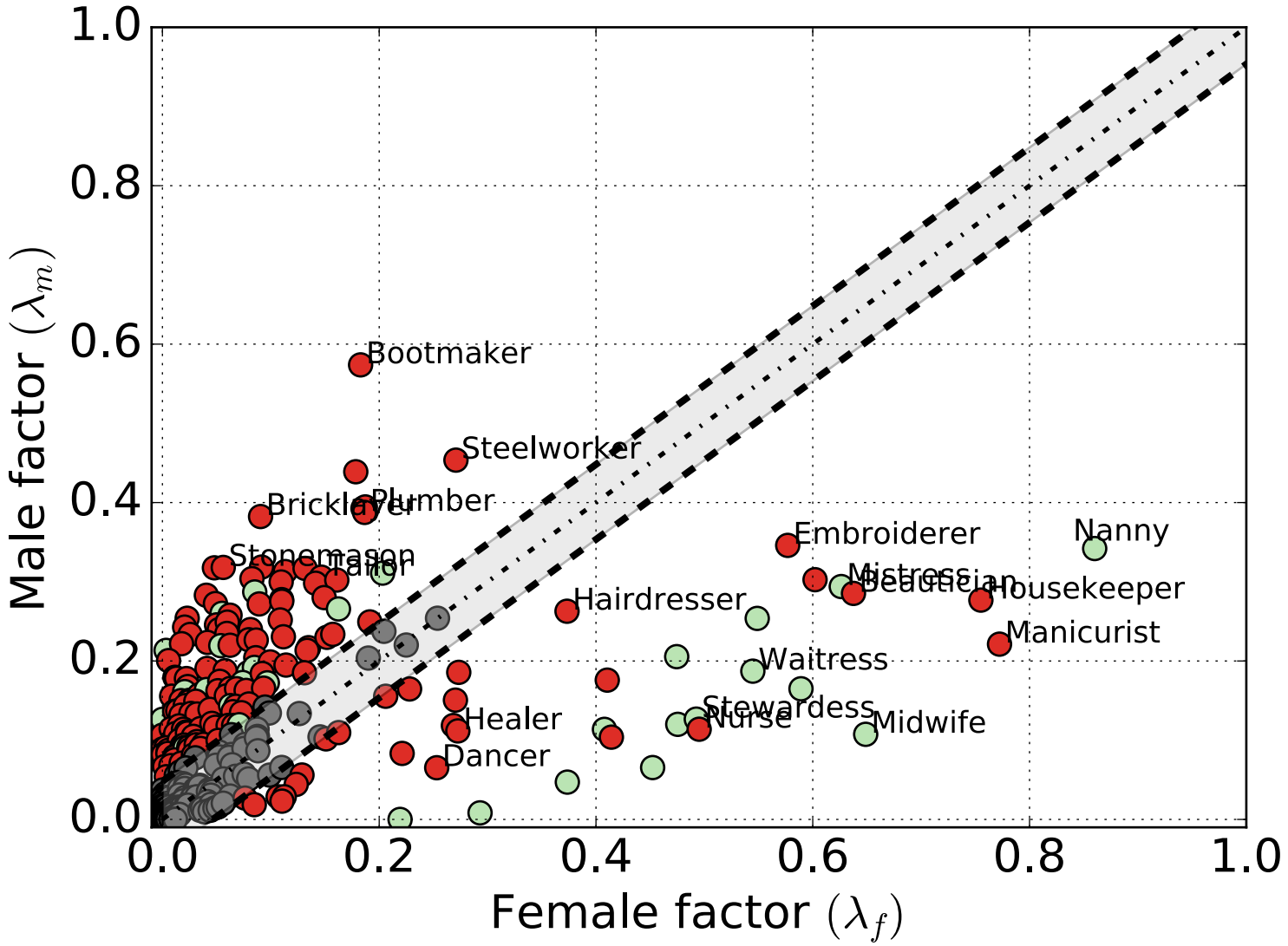
same gender-neutral pronoun


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



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

Recurrent Neural Networks

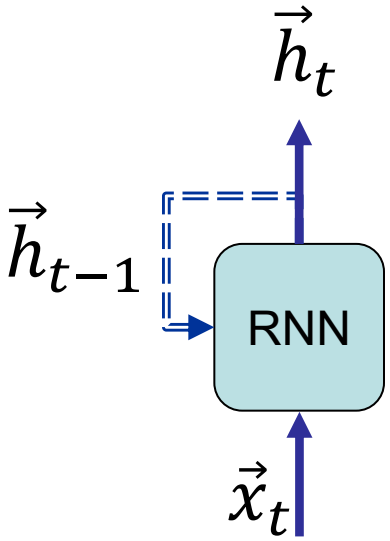
- Encodes/Embeds a sequence of entities (vectors) such as …
  - Sequence of word vectors
  - Time series

- … to a final composed vector as well as intermediary vectors on each time step

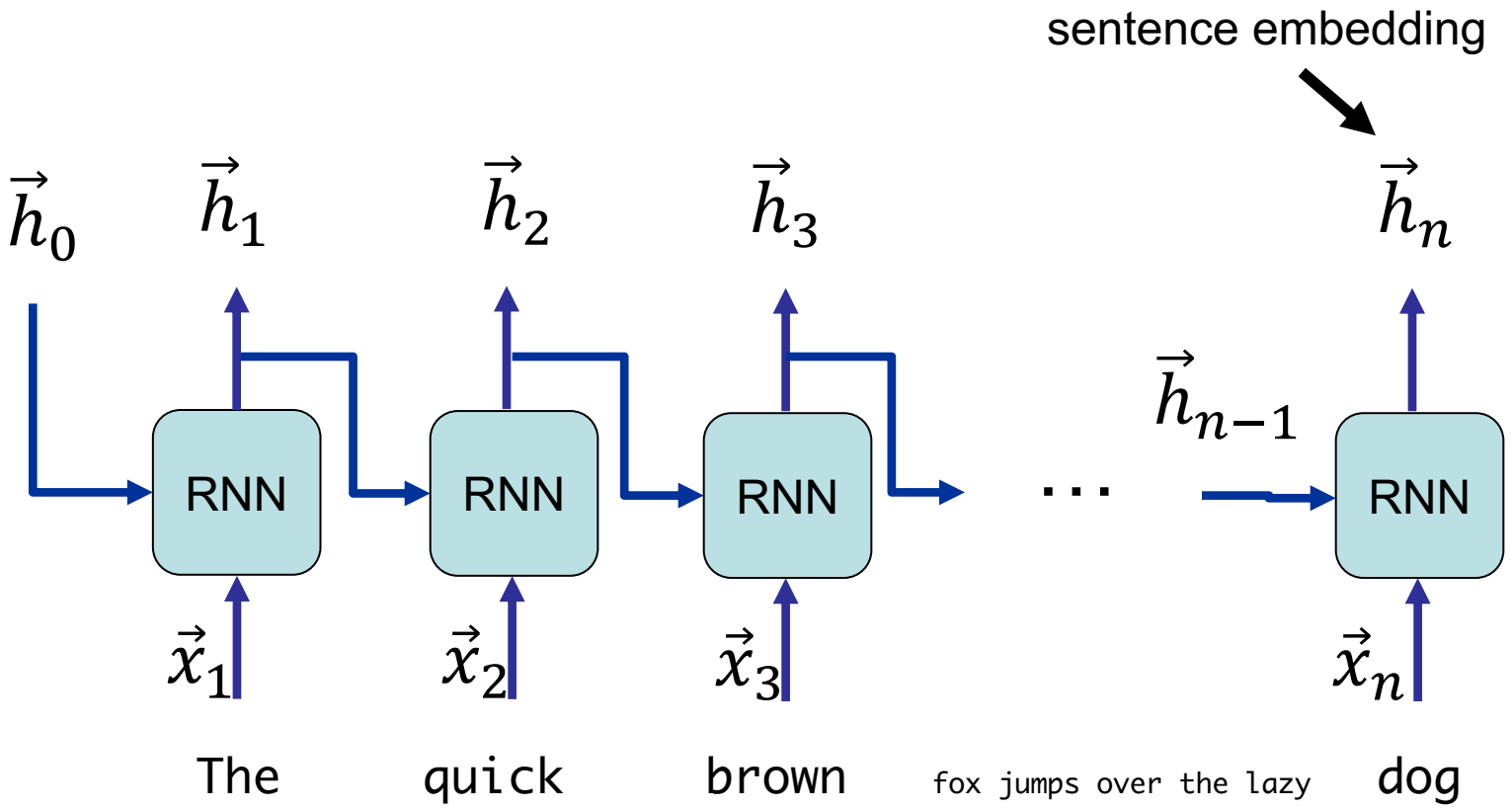
- The output is a function of input and the output of the previous time step

- Output $\vec{h}_t$ is also called hidden state

- With hidden state $\vec{h}$, the network access to a sort of memory from previous entities



RNN - unfolded

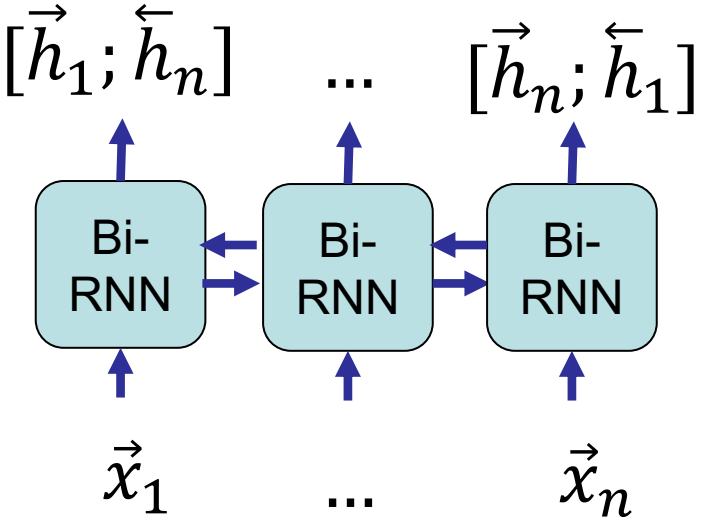


Types and Bidirectional

- Common types of RNN
  - Standard (Elman) RNN
  - Gated Recurrent Unit (GRU)
  - Long Short-Term Memory (LSTM)

Bidirectional RNN

- Reading the sequence from
  - Beginning to end $\vec{h}_1$ to $\vec{h}_n$
  - End to beginning $\overleftarrow{h}_n$ to $\overleftarrow{h}_1$
- Output at time step *t* is the concatenation of two hidden states $\vec{h}_t$ and $\overleftarrow{h}_{n-t}$



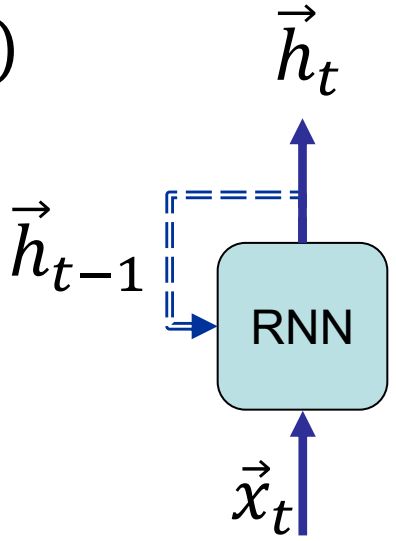
Standard RNN

- General form of the RNN function

$$\vec{h}_t = RNN(\vec{x}_t, \vec{h}_{t-1})$$

- First projects input $\vec{x}_t$ with parameter matrix W_{ih} , and previous hidden state $\vec{h}_{t-1}$ with parameter matrix W_{hh}, and then applies a non-linearity function on their sum:

$$\vec{h}_t = \tanh(\vec{x}_t \cdot W_{ih} + \vec{h}_{t-1} \cdot W_{hh})$$

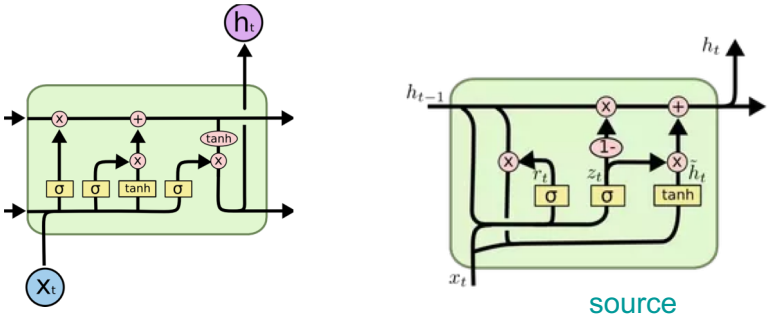


➢ Parameters are shown in red

➢ For simplicity biases are removed from the formulas

RNNs with Gates

- Two problems of Standard RNN:
  - Exploding gradient: approached with gradient clipping
  - Vanishing gradient: approached with gated RNNs such as GRU and LSTM by learning to "forget" the some parts of the memory



source

Gate Vector

- Commonly, a vector with values between 0 and 1, used for elementwise multiplication to an entity vector $\vec{v}$

$$\vec{g} \odot \vec{v}$$

- Acts as a gate on information flow of the entity vector

Gated Recurrent Unit (GRU)

- Calculate reset gate from input $\vec{x}_t$ and previous hidden state $\vec{h}_{t-1}$

$$\vec{r}_t = \sigma(\vec{x}_t \cdot W_{ir} + \vec{h}_{t-1} \cdot W_{hr})$$

- Calculate novel information vector $\vec{n}_t$ from input $\vec{x}_t$ and a "forgotten" part of previous hidden state $\vec{h}_{t-1}$

$$\vec{n}_t = tanh(\vec{x}_t \cdot W_{ih} + \vec{r}_t \odot (\vec{h}_{t-1} \cdot W_{hh}))$$

- Calculate update gate from input $\vec{x}_t$ and previous hidden state $\vec{h}_{t-1}$

$$\vec{z}_t = \sigma(\vec{x}_t \cdot W_{iz} + \vec{h}_{t-1} \cdot W_{hz})$$

- Finally output is composed of the novel information $\vec{n}_t$ and previous hidden state $\vec{h}_{t-1}$, decided by update gate

$$\vec{h}_t = (1 - \vec{z}_t) \odot \vec{n}_t + \vec{z}_t \odot \vec{h}_{t-1}$$

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

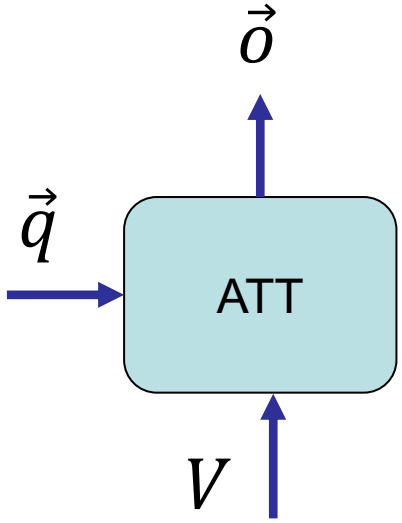
Attention Networks

- Encodes/Embeds a set of vectors to a composed vector

- Given a query vector $\vec{q}$ and a matrix of values V, an attention network "looks up" the query in the values, and produce output vector $\vec{o}$

- General form of an attention network as a function

$$\vec{o} = ATT(\vec{q}, V)$$



➢ In general, query is also a matrix. Here it is assumed as a vector for simplicity

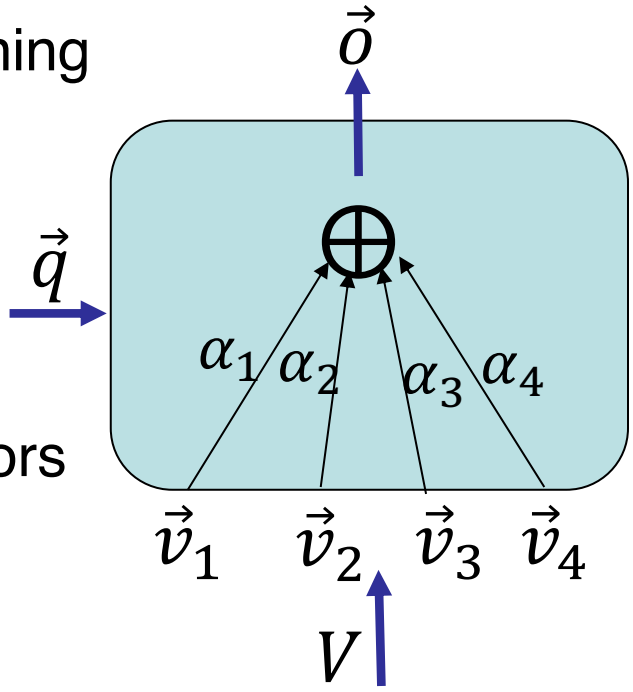
Attention Networks - details

- Given the query, an attention network learns to assign some amount of attention α_i on each value vector $\vec{v}_i$ using the attention function f

$$\alpha_i = f(\vec{q}, \vec{v}_i)$$

- where α is a probability distribution, meaning

$$\sum_{i=1}^{n} \alpha_i = 1$$



- Output is the weighted sum of value vectors

$$\vec{o} = \sum_{i=1}^{n} \alpha_i . \overrightarrow{v_i}$$

Method 1 - Scaled Dot-Product Attention

- First unnormalized attention $\tilde{a}_i$ is calculated by a simple dot product

$$\tilde{a}_i = \frac{\vec{q}.\vec{v}_i}{\sqrt{d}}$$

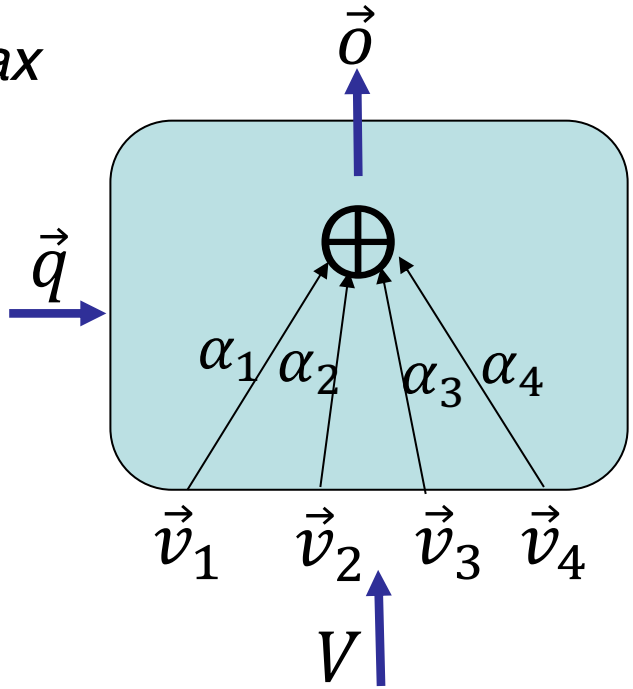
- where *d* is the dimension of vectors

- Attentions are then normalized with *softmax*

$$\alpha_i = softmax(\vec{\tilde{a}})_i$$

- As before, output is the weighted sum

$$\vec{o} = \sum_{i=1}^{n} \alpha_i . \vec{v_i}$$



Method 2 - Multi-Layer Perceptron Attention

- First unnormalized attention $\tilde{a}_i$ is calculated by a neural network

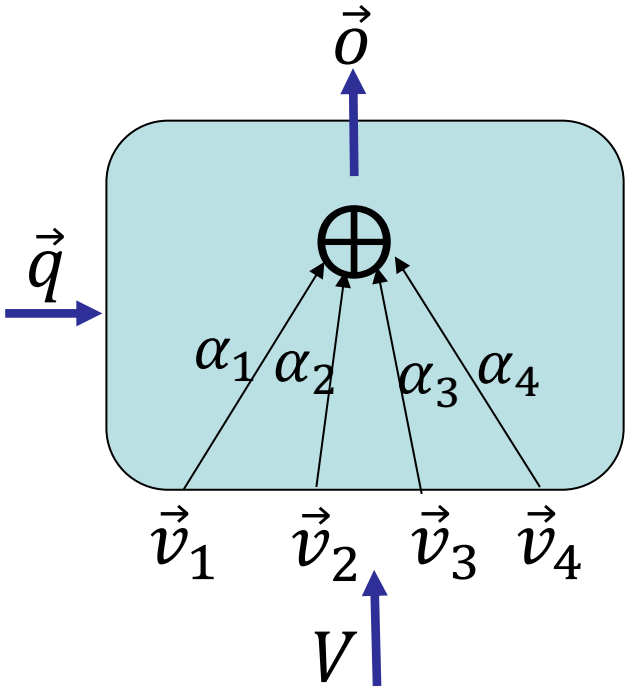
$$\tilde{a}_i = \vec{u} \cdot tanh(\vec{q} \cdot W_1 + \vec{v}_i \cdot W_2)$$

- As before, attentions are normalized with *softmax*

$$\alpha_i = softmax(\vec{\tilde{a}})_i$$

- and output is …

$$\vec{o} = \sum_{i=1}^{n} \alpha_i . \vec{v_i}$$



➢ Model parameters are shown in red

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**

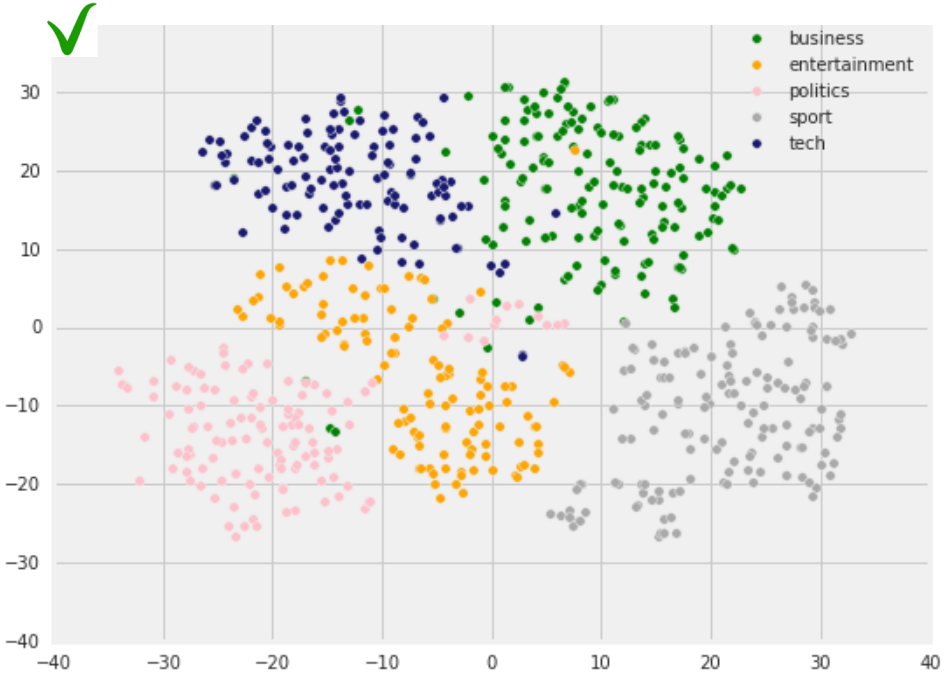
Document Classification (Recap)

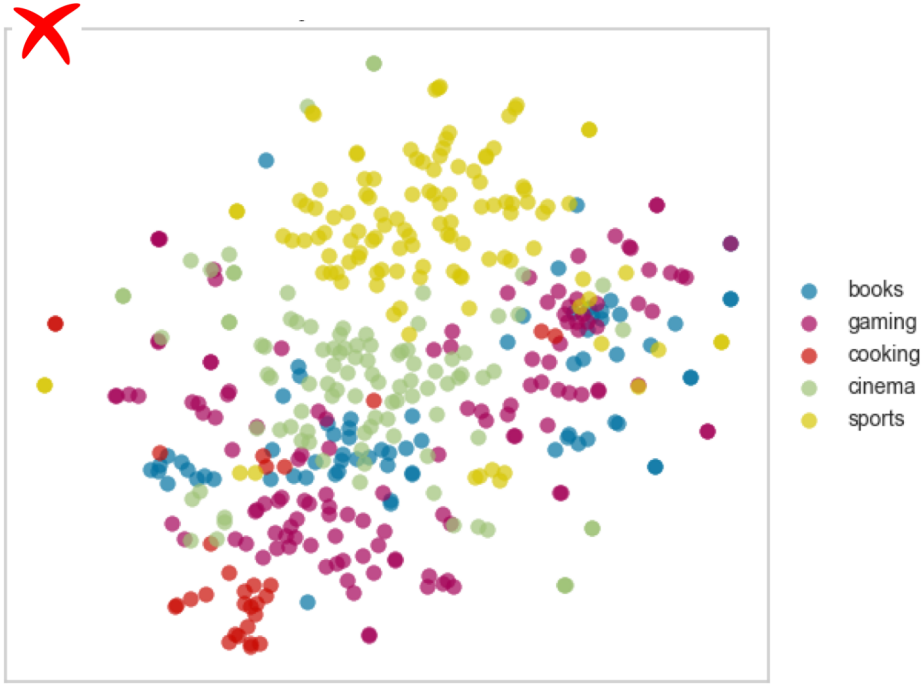
- Create a document representation, e.g. using
  - TF-IDF → spare vectors
  - Principle Component Analysis (PCA) → dimensionality reduction
  - Latent Semantic Indexing (LSI) → semantic vectors
  - Latent Dirichlet Allocation (LDA) → topic-based vectors
  - Deep Learning

- Steps
  - Given document representations of training data, learn a classifier to predict the classes
  - Use the classification model to predict the classes of the test-set documents
  - Evaluate the predictions

Document Representation

- Document representation is the key!
- The classes can be more effectively predicted when document representations are linearly separable.





Two sample document representation sets, projected to two-dimensional spaces

source source

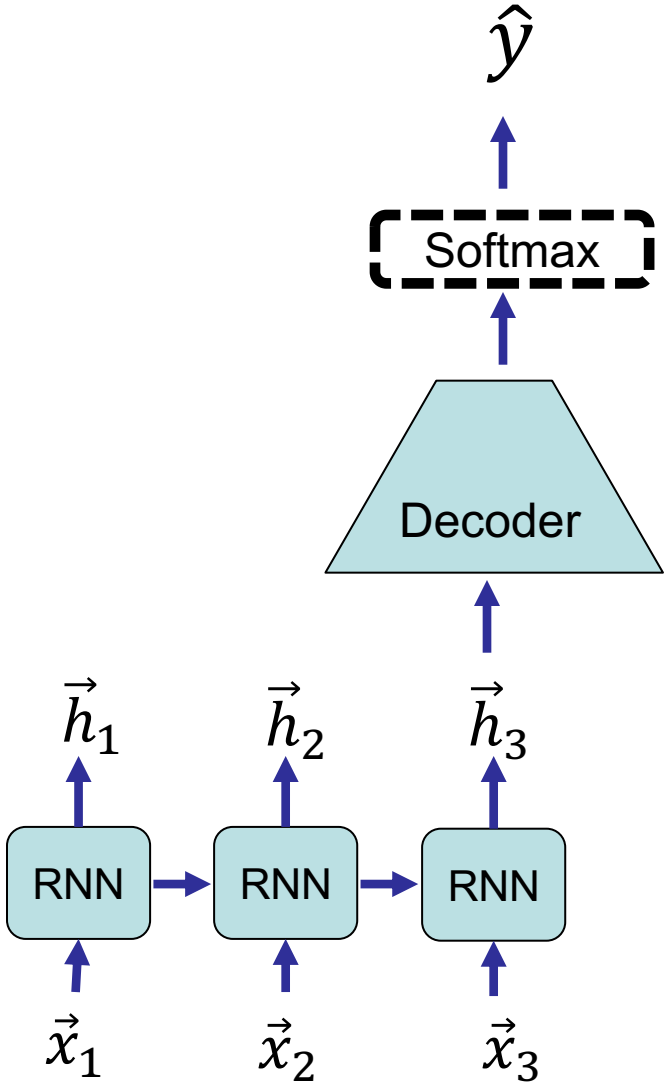
Document Classification with DL

RNNClassModel (practical session)

$\vec{x}$ word embedding

$\hat{y}$ probability distribution of predicted output

Decoder: linear projection to output classes



Document Classification with DL

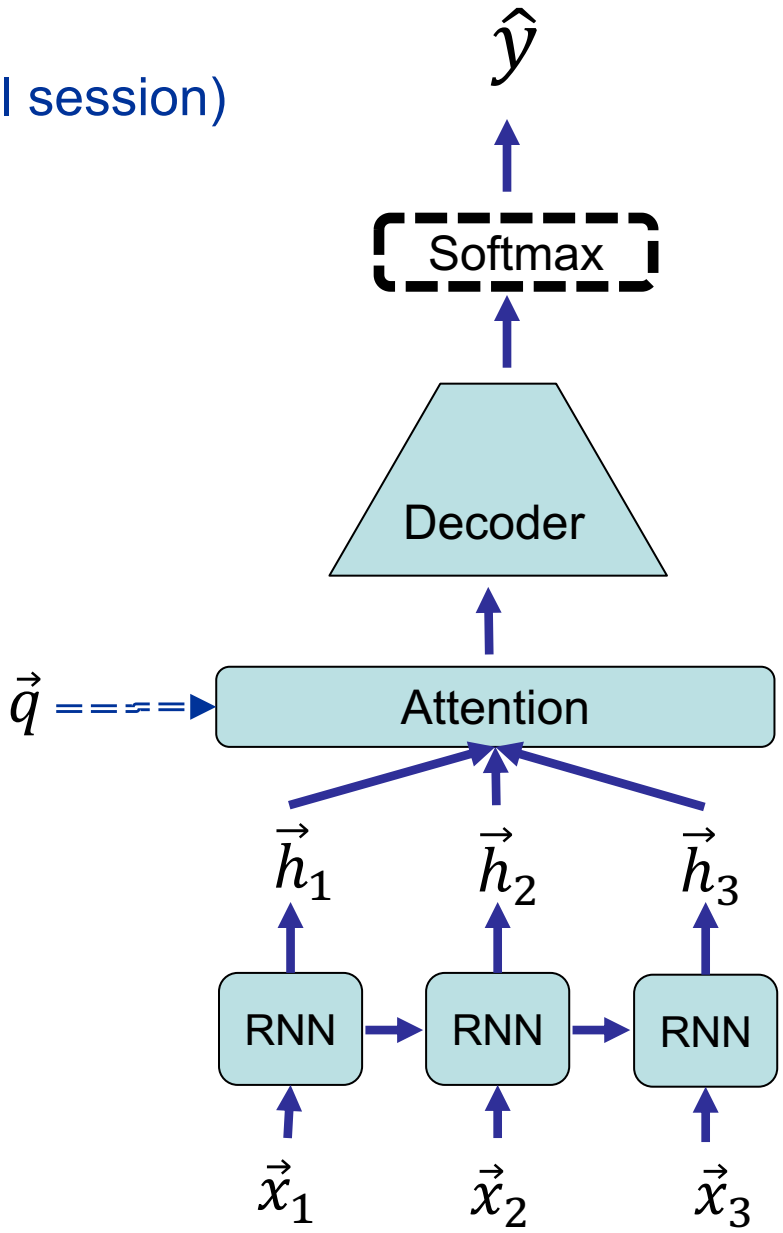
ATTClassModel (practical session)

$\vec{x}$ word embedding

$\hat{y}$ probability distribution of predicted output

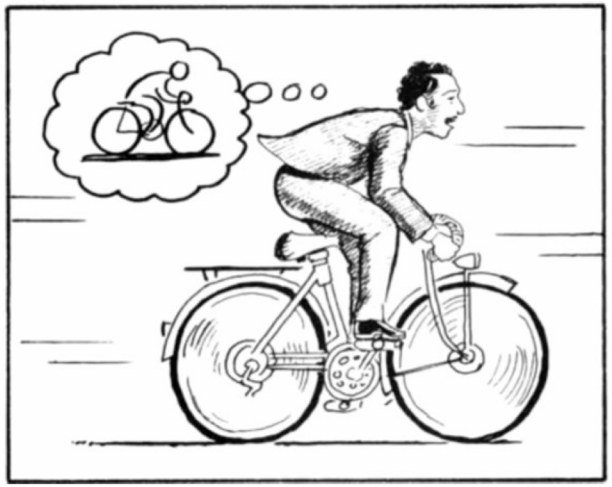
Decoder: linear projection to output classes

$\vec{q}$ attention query on words: *which word is informative?*



Challenges

- Semantics of language and the world



[8]

"The image of the world around us, which we carry in our head, is just a model. Nobody in his[/her] head imagines all the world, government or country. He[/She] has only selected concepts, and relationships between them, and uses those to represent the real system." Mental Model [7]

- Representation Learning
  - Abstract representation of spatial and temporal aspects of information
  - Various granularities → abstraction level
  - Task-specific, domain specific → transfer learning
  - Commonalities among languages → multilingual models

Challenges

- Understanding information contents
  - Information Retrieval
  - Summarization
  - Question/Answering

- Aspects of information tailoring and provision
  - Personalization vs. De-personalization
  - Controversy detection
  - Multiple points of view

Challenges

- Exploring aspects of society
  - Computational Social Science with NLP
    - Economy
    - Sociology
    - Psychology
- Ethics, fairness, and transparency
  - implications of the new technology on the society
  - Ownership of data and models, laws, etc.
  - Ethical bias in data and algorithms
  - Interpretability of the models

References

[1] Jurafsky, Dan, and James H. Martin. *Speech and language processing*. Vol. 3. London: Pearson, 2014.
[2] Forrester, Jay Wright. Counterintuitive behavior of so- cial systems, 1971. URL https://en.wikipedia. org/wiki/Mental_model. [Online; accessed 01- Nov-2017].
[3] Ha, David, and Jürgen Schmidhuber. "World Models." arXiv preprint arXiv:1803.10122 (2018)