FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Learning Word Embeddings

Navid Rekabsaz

navid.rekabsaz@idiap.ch

Agenda

- Neural Networks - Recap
- Language Modeling
- Neural Word Embedding - Revisited
  - word2vec Skip-Gram with Negative Sampling
  - Bias in Word Embeddings
- Contextualized Word Embeddings
  - ELMo
  - BERT

Agenda

- Neural Networks - Recap
- Language Modeling
- Neural Word Embedding - Revisited
  - word2vec Skip-Gram with Negative Sampling
  - Bias in Word Embeddings
- Contextualized Word Embeddings
  - ELMo
  - BERT

Linear Algebra

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

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

    dimensions: 1×d · d×e =
  - $\boldsymbol{A} \cdot \boldsymbol{B} =$

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

Neural Networks

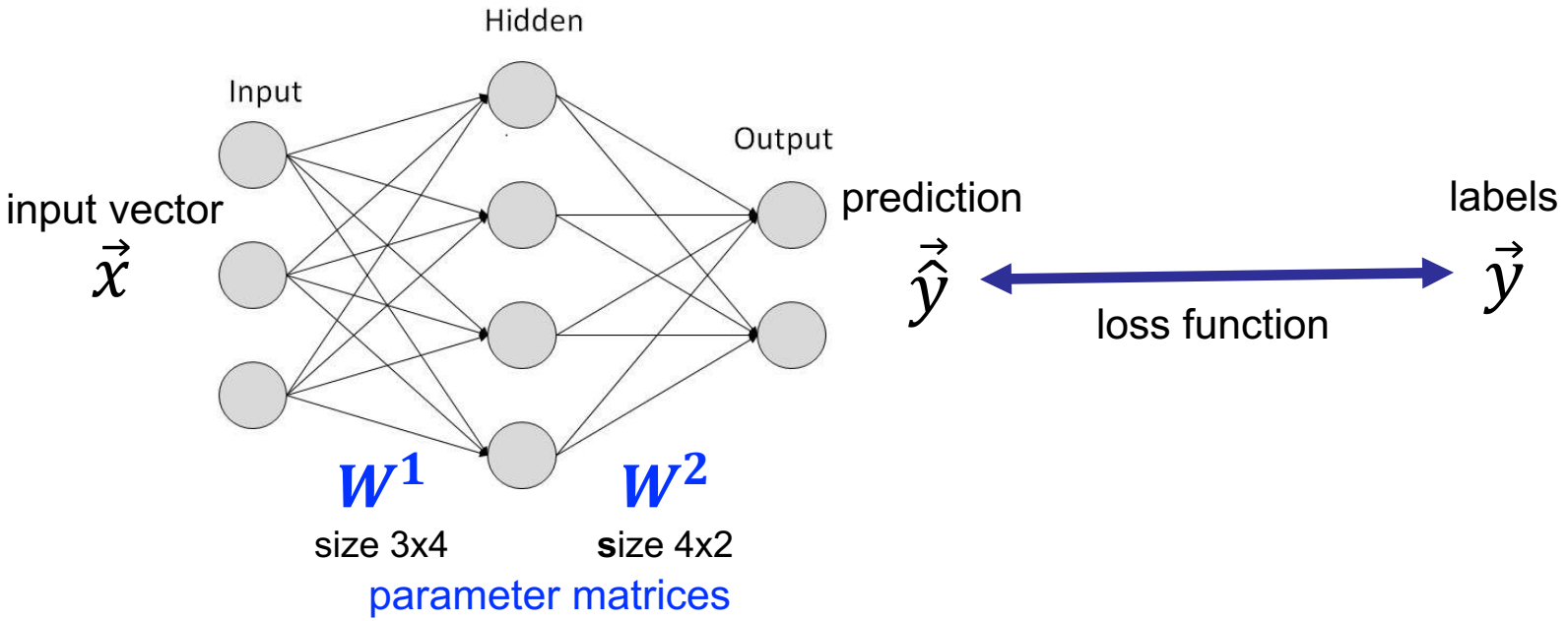
- Neural Networks are non-linear functions with many parameters

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

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

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

- Optimized using gradient-based algorithms like Stochastic Gradient Descent

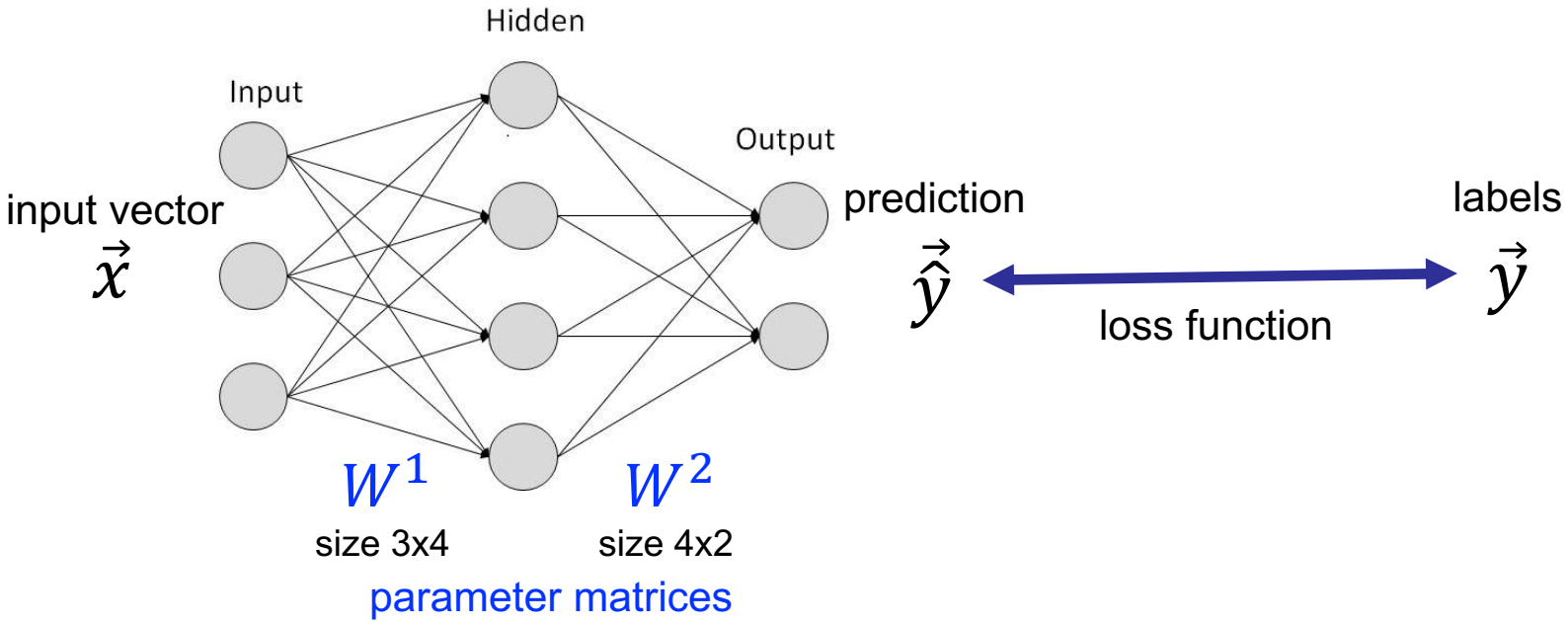


Training a Neural Network

Initialize parameters

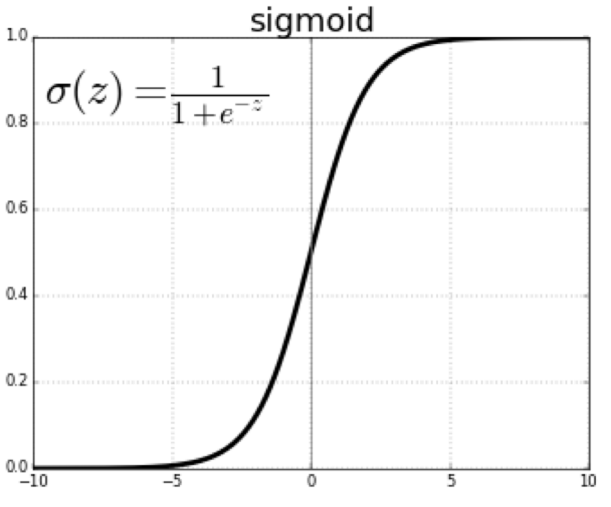
Loop over training data (or minibatches)

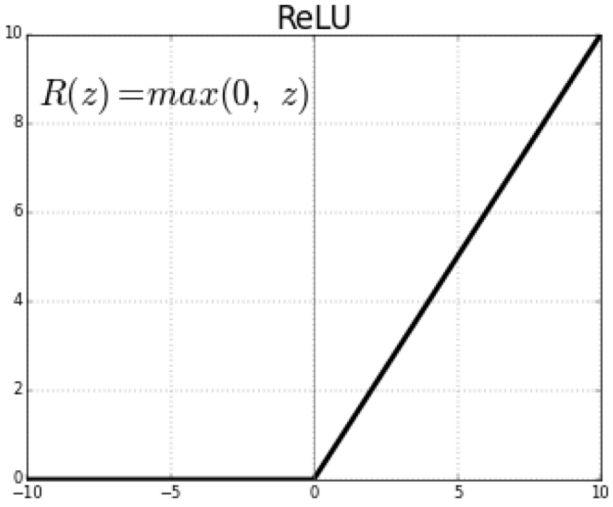
1. Do forward pass: given input $\vec{x}$ predict output $\hat{y}$
2. Calculate loss by comparing $\hat{y}$ with labels y
3. Do backpropagation: calculate the gradient of each parameter in regard to the loss function
4. Update parameters in the direction of gradient
5. Exit if some stopping criteria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

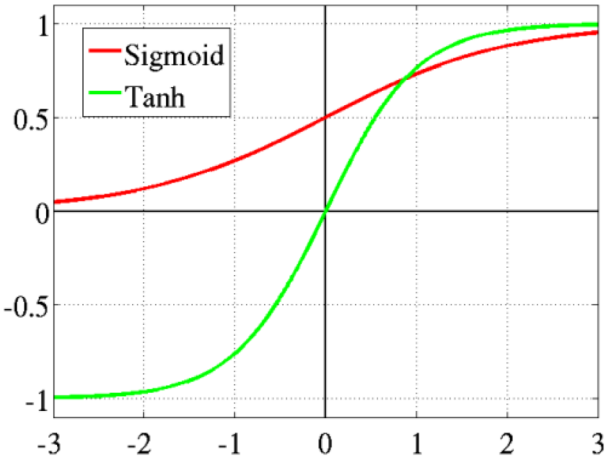


Non-linearities

- 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

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

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

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

- For example, given four classes:

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

Recurrent Neural Networks

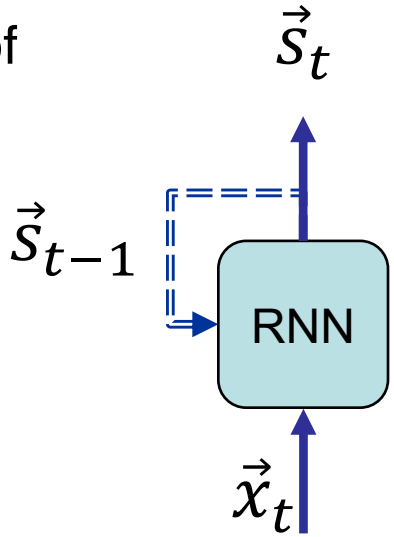
- Encodes/Embeds a sequence of entities (vectors) such as …
  - Sequence of word vectors
  - Time series

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

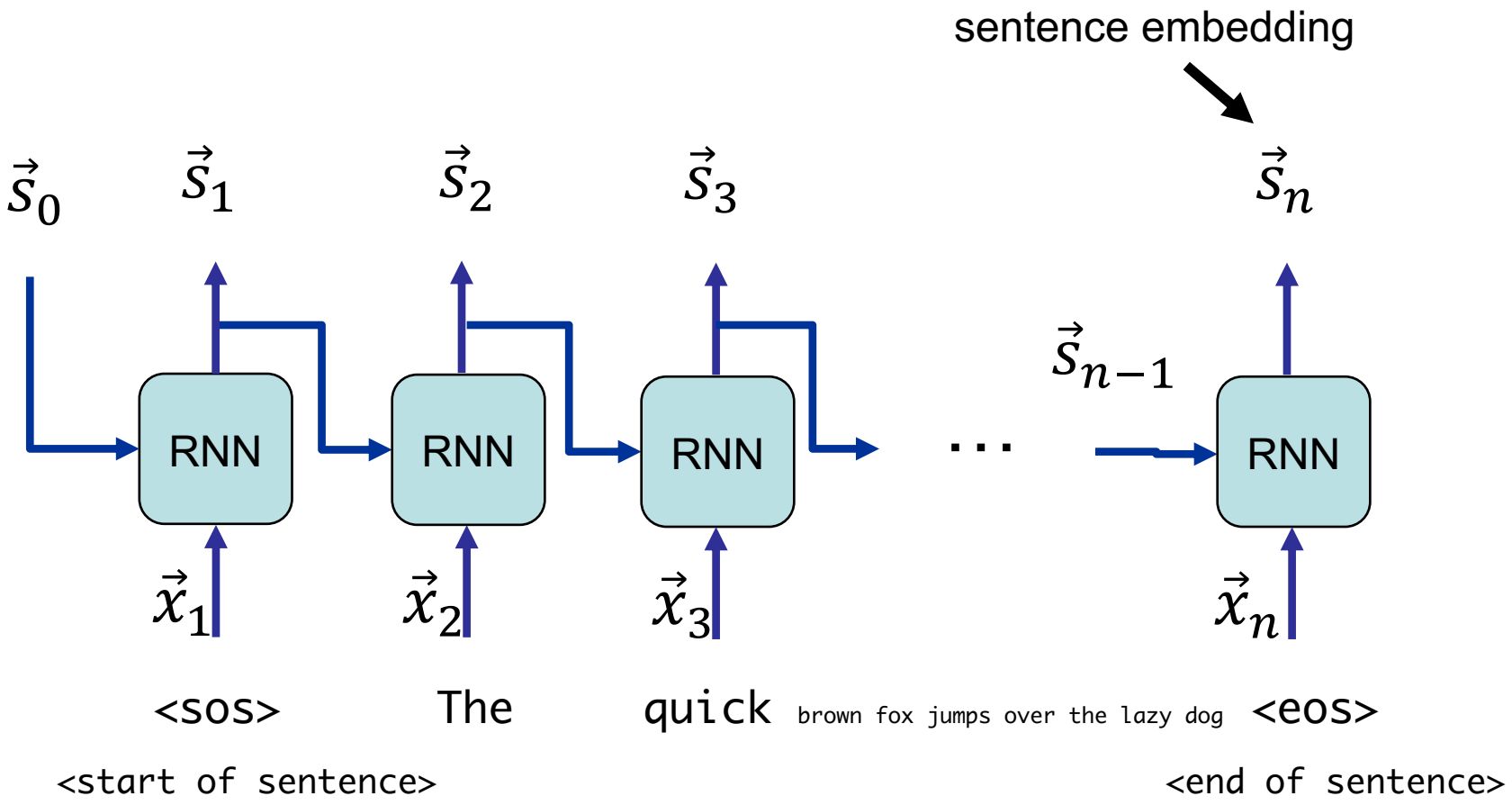
- The output $\vec{s}_t$ is a function of input and the output of the previous time step

- Output is also called hidden state

- Hidden state is in fact a "memory" of the previous entities



RNN - unfolded

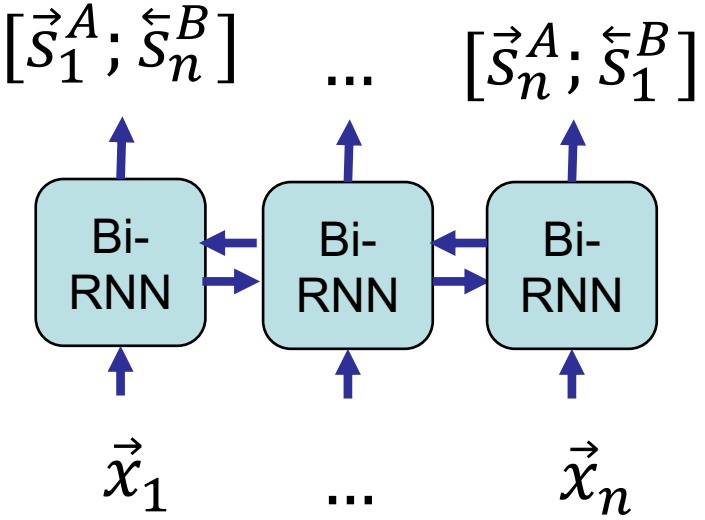


Types and Bidirectional

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

Bidirectional RNN

- Two RNNs together: *A* and *B*
- Reading the sequence from
  - Beginning to end with RNN *A*
  - End to beginning with RNN *B*
- Output at time step *t* is the concatenation of two hidden states $\vec{s}_t^A$ and $\overleftarrow{s}_{n-t+1}^B$



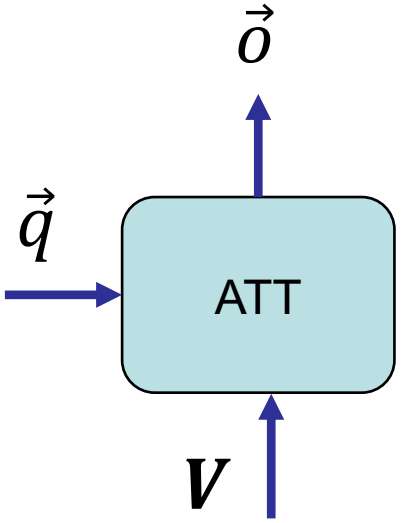
Attention Networks

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

- Given a query vector $\vec{q}$ and a matrix of values $\boldsymbol{V}$, an attention network assigns weights (attentions) to the vectors of $\boldsymbol{V}$, and produces output vector $\vec{o}$

- General form of an attention network

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



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

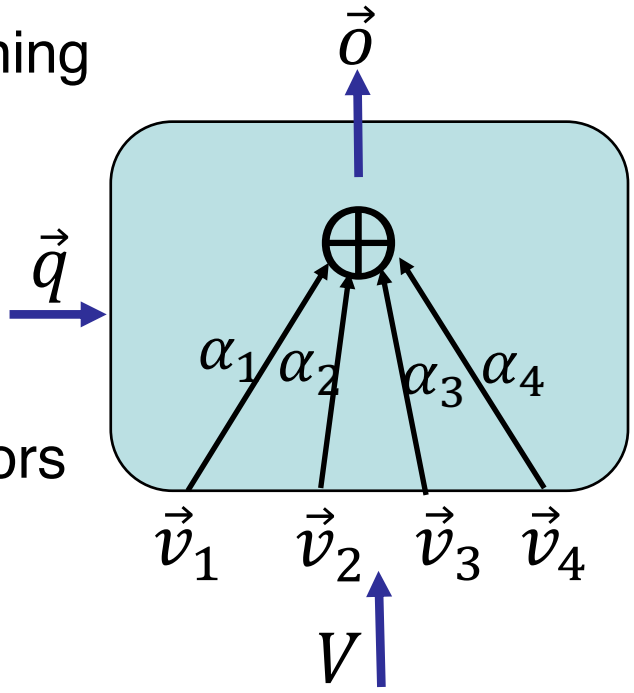
Attention Networks - details

- 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

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

Multi-Layer Perceptron Attention

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

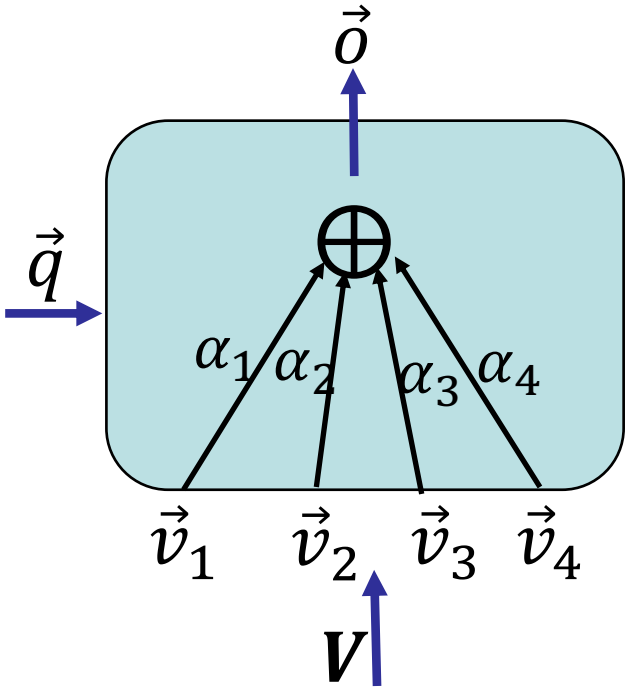
$$\tilde{a}_i = \vec{u} \cdot \tanh(\vec{q} \cdot W_1 + \vec{v}_i \cdot W_2)$$

- As before, attentions are normalized with softmax

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

- and output is …

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



➢ Model parameters are shown in red

Agenda

- Neural Networks - Recap
- Language Modeling
- Neural Word Embedding - Revisited
  - word2vec Skip-Gram with Negative Sampling
  - Bias in word embeddings
- Contextualized Word Embeddings
  - ELMo
  - BERT

Language Model

- Ideally, a Language Model aims to model the language!
- In practic, Language Models use the statistics of the collection to calculate the probability:

$$P(\mathrm{w}|\mathrm{context})$$

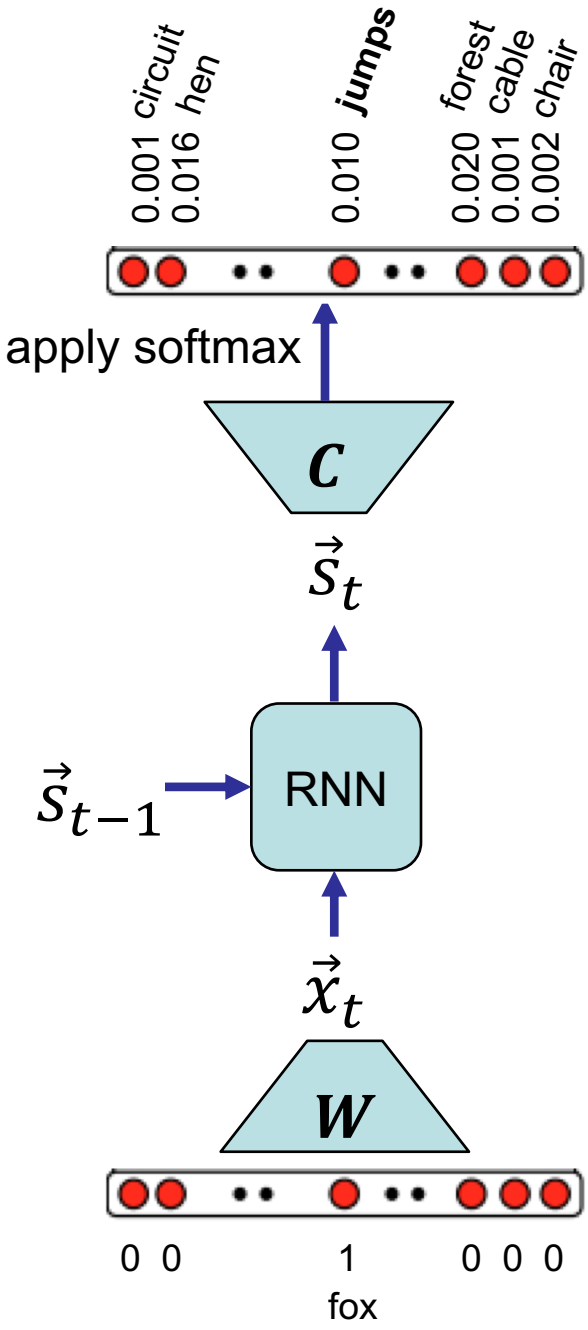
- A Language Model can answer the questions like
  - Given “The quick brown fox jumps over the lazy …”, what is the probability of the next word:

$$P(w|\text{The quick brown fox jumps over the lazy})$$

- Neural Networks is one way of calculating this probability

Training Language Model with RNN

- Input:
  - a text corpus with n words and $|\mathbb{V}|$ unique words: $[\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_n]$
  - a RNN model
  - encoder embedding $\boldsymbol{W}$ with dimensions: $|\mathbb{V}| \times \mathrm{d}$
  - decode embedding $\boldsymbol{C}$ with dimensions: $|\mathbb{V}| \times \mathrm{d}$
- Loop over the text corpus; at time step t:
  a. Create the one-hot vector of the word w_t
  b. Calculate dot product of the one-hot vector to $\boldsymbol{W}$, resulting to $\vec{w}_t$
  c. Input $\vec{w}_t$ and $\vec{s}_{t-1}$ to RNN and calculate $\vec{s}_t$
  d. Calculate dot product of $\vec{s}_t$ to $\boldsymbol{C}^{\mathbf{T}}$, followed by softmax, achieving
  $$P(\mathrm{w}_{t+1} | \mathrm{w}_t, \vec{s}_{t-1})$$
  e. Calculate Negative Log Likelihood loss $\mathcal{L} = -\log p(\mathrm{w}_{t+1} | \mathrm{w}_t, \vec{s}_{t-1})$
  f. Update parameters using the gradient of the loss



- Calculate loss
- Update the parameters

Transfer Learning with Embeddings

- Language modeling is a common aspect in various language processing tasks

- We train neural embedding models on large-scale text corpora using the unsupervised objective of language modeling
  - takes relatively longer time, but is done once

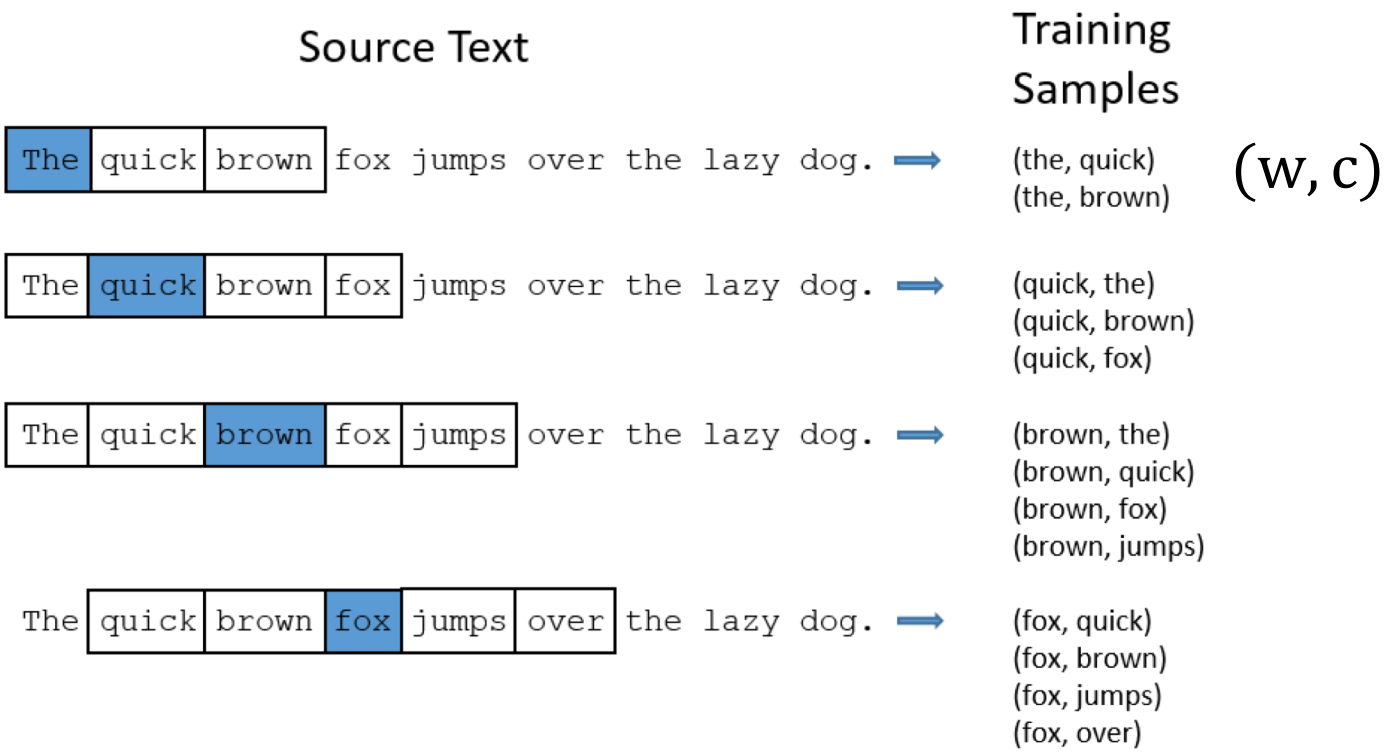
- The pre-trained parameters of the models are then used in various supervised tasks
  - Almost all language processing tasks such as Information Retrieval, Summarization, Question Answering, etc.
  - Usually followed by a fine-tuning of the pre-trained parameters

Agenda

- Neural Networks - Recap
- Language Modeling
- Neural Word Embedding - Revisited
  - word2vec Skip-Gram with Negative Sampling
  - Bias in Word Embeddings
- Contextualized Word Embeddings
  - ELMo
  - BERT

Recap: Skip-Grams

- Word pairs found in the text within a context window of size 2
- Language Model objective → predict context word from the center word: $P(c|w)$



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

Tesgüino!

Example from J. R. Firth

bottle

fermentation

grain

brew

Ale

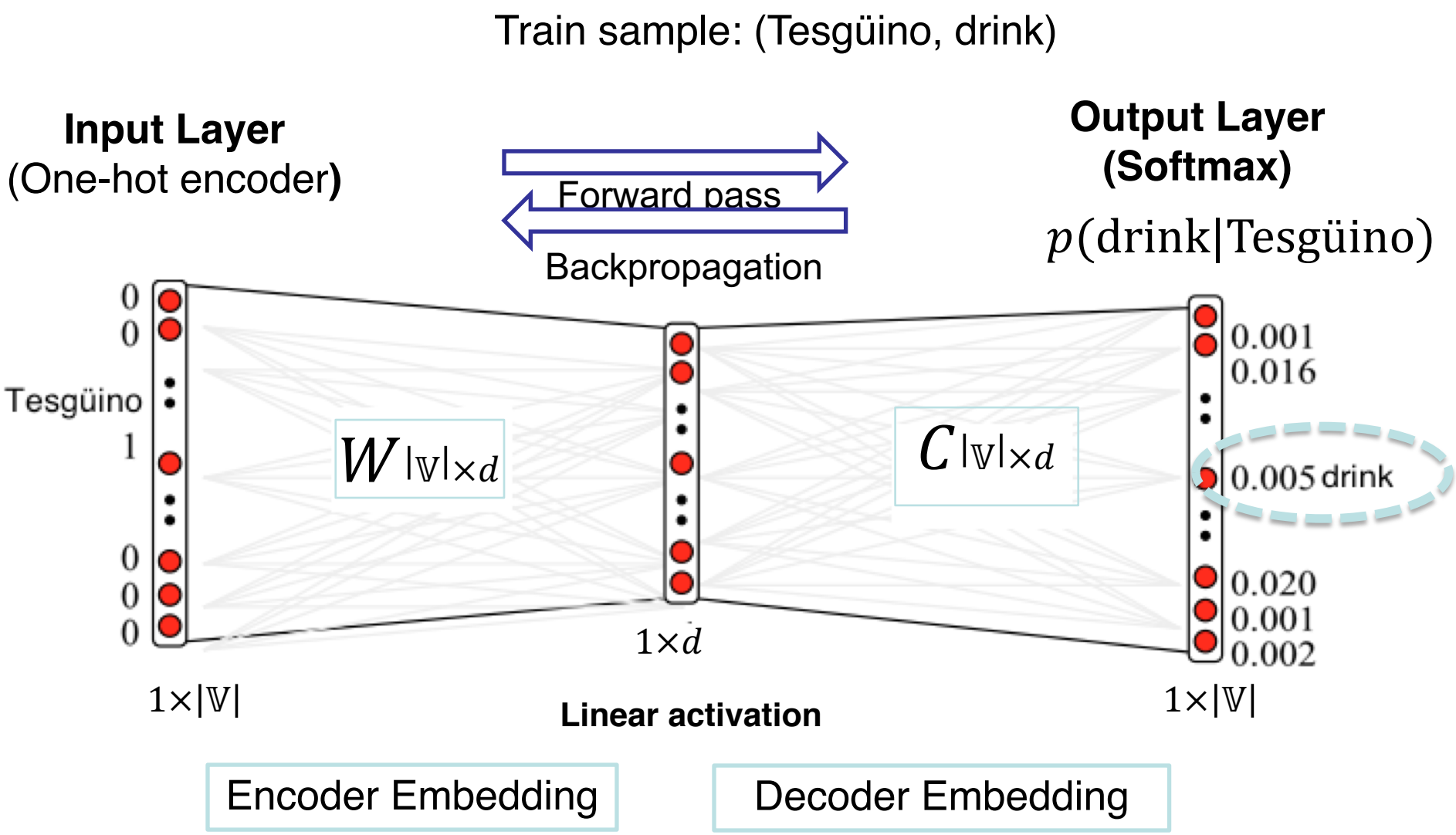
medieval

pale

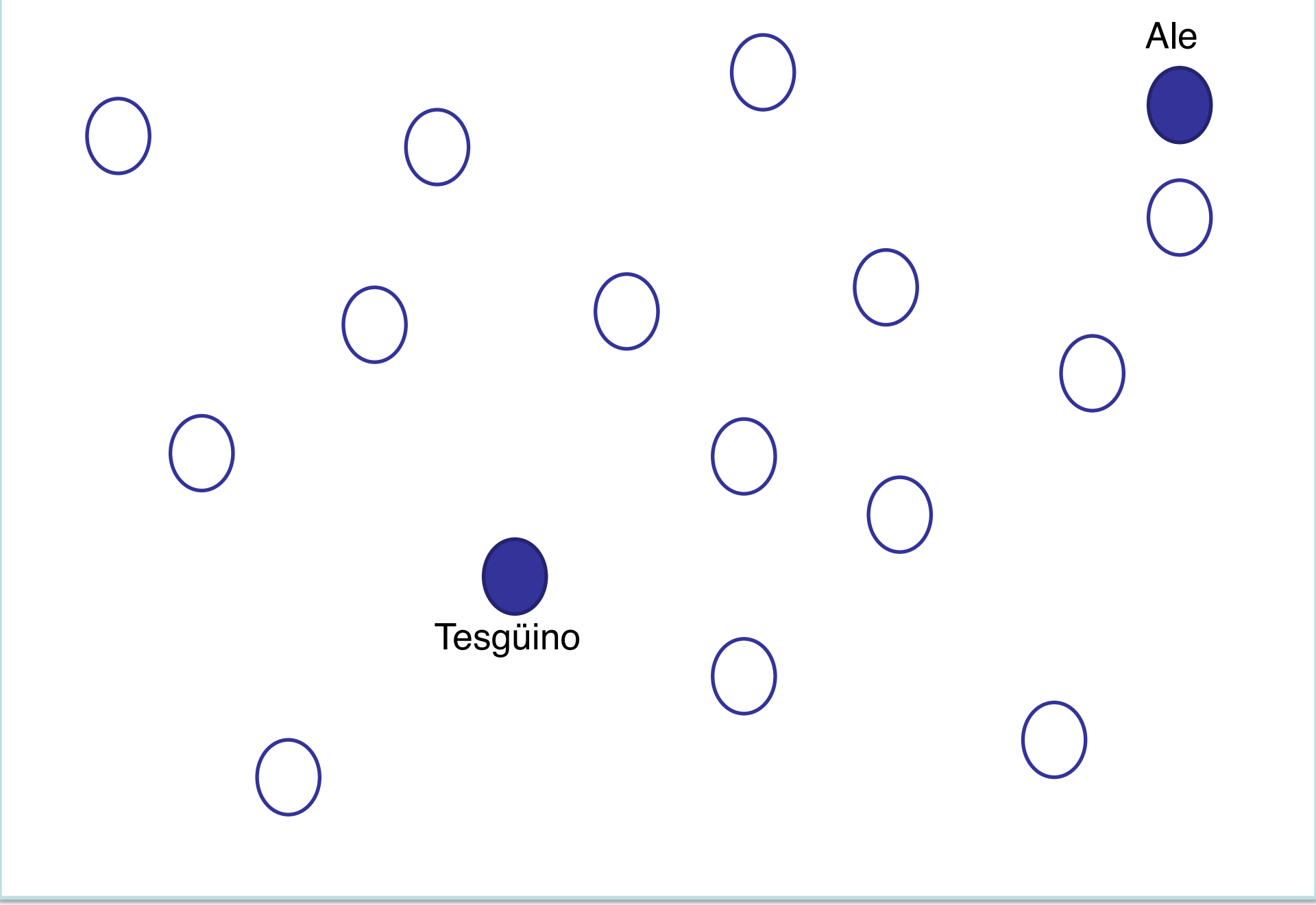
bar

drink

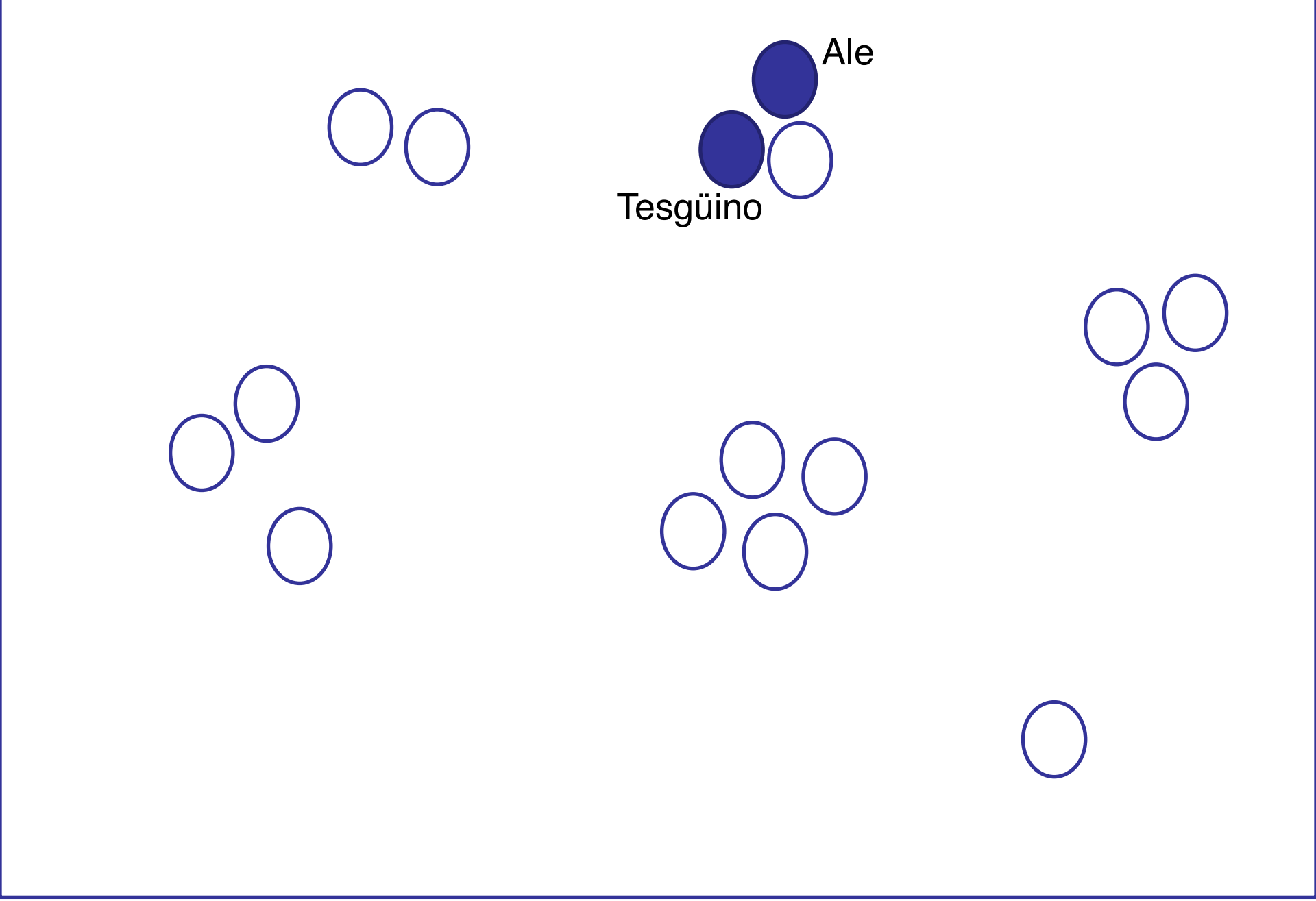
Recap: Neural Word Embedding Architecture



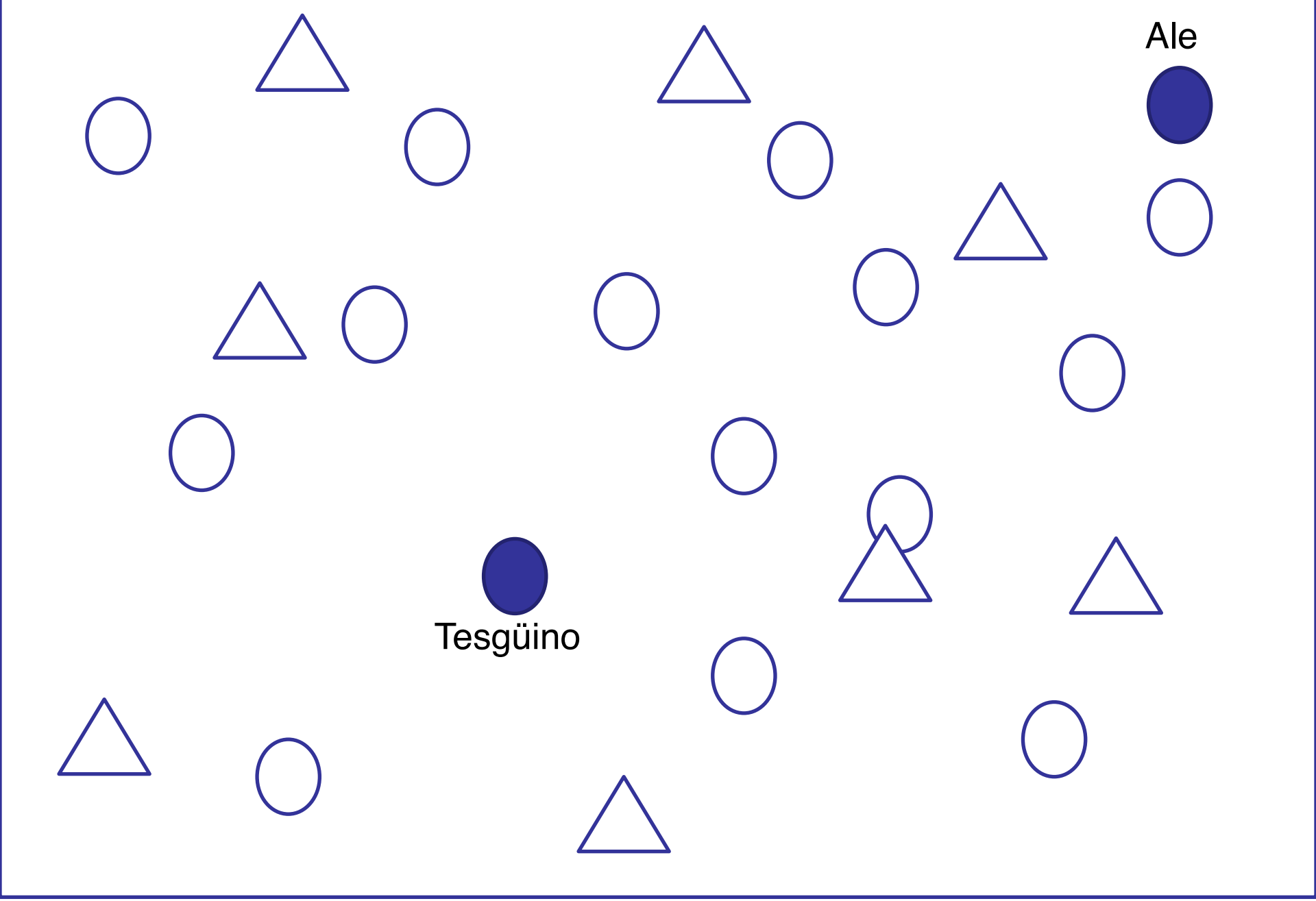
https://web.stanford.edu/~jurafsky/slp3/



Encoder Embedding

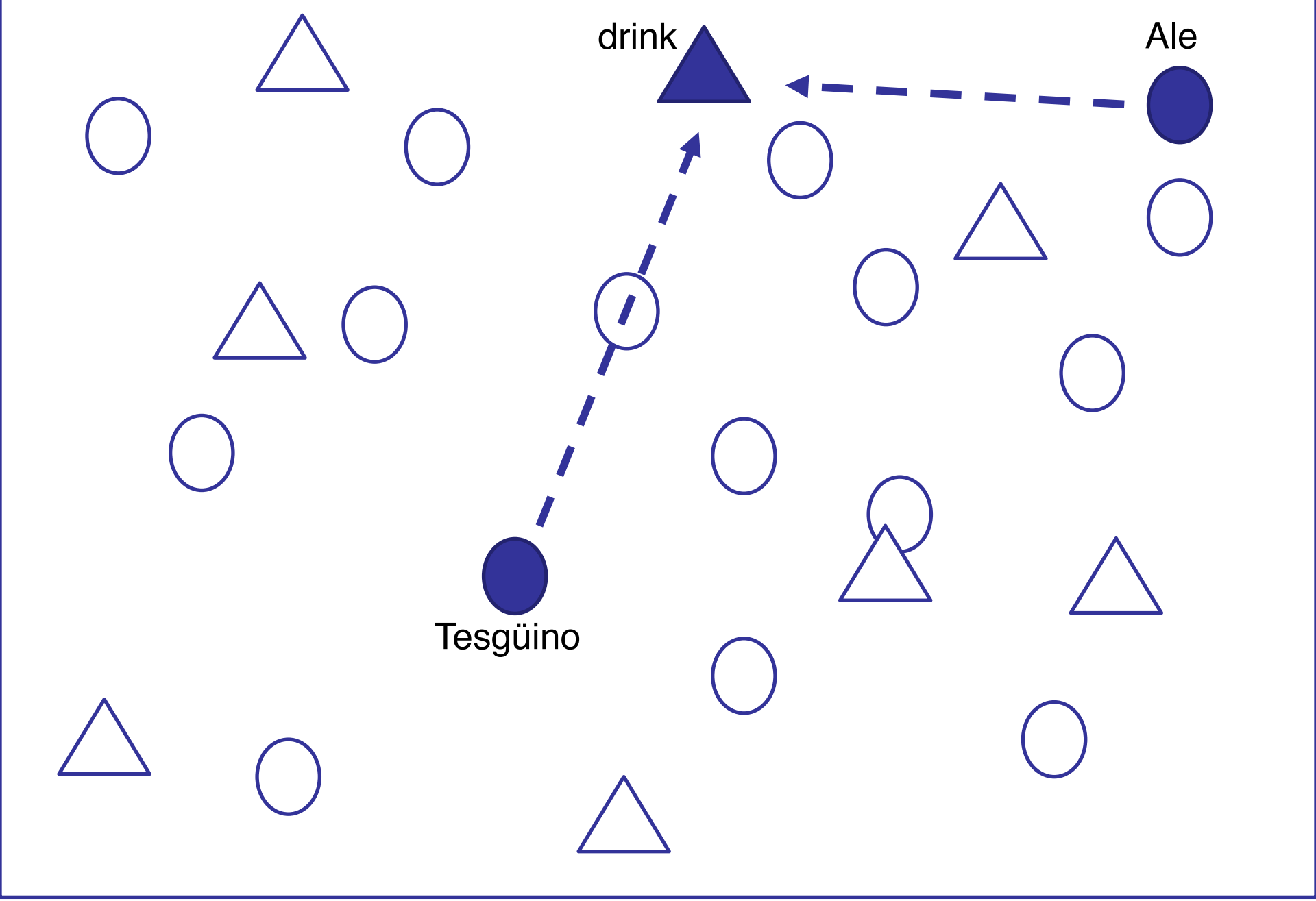


Encoder Embedding



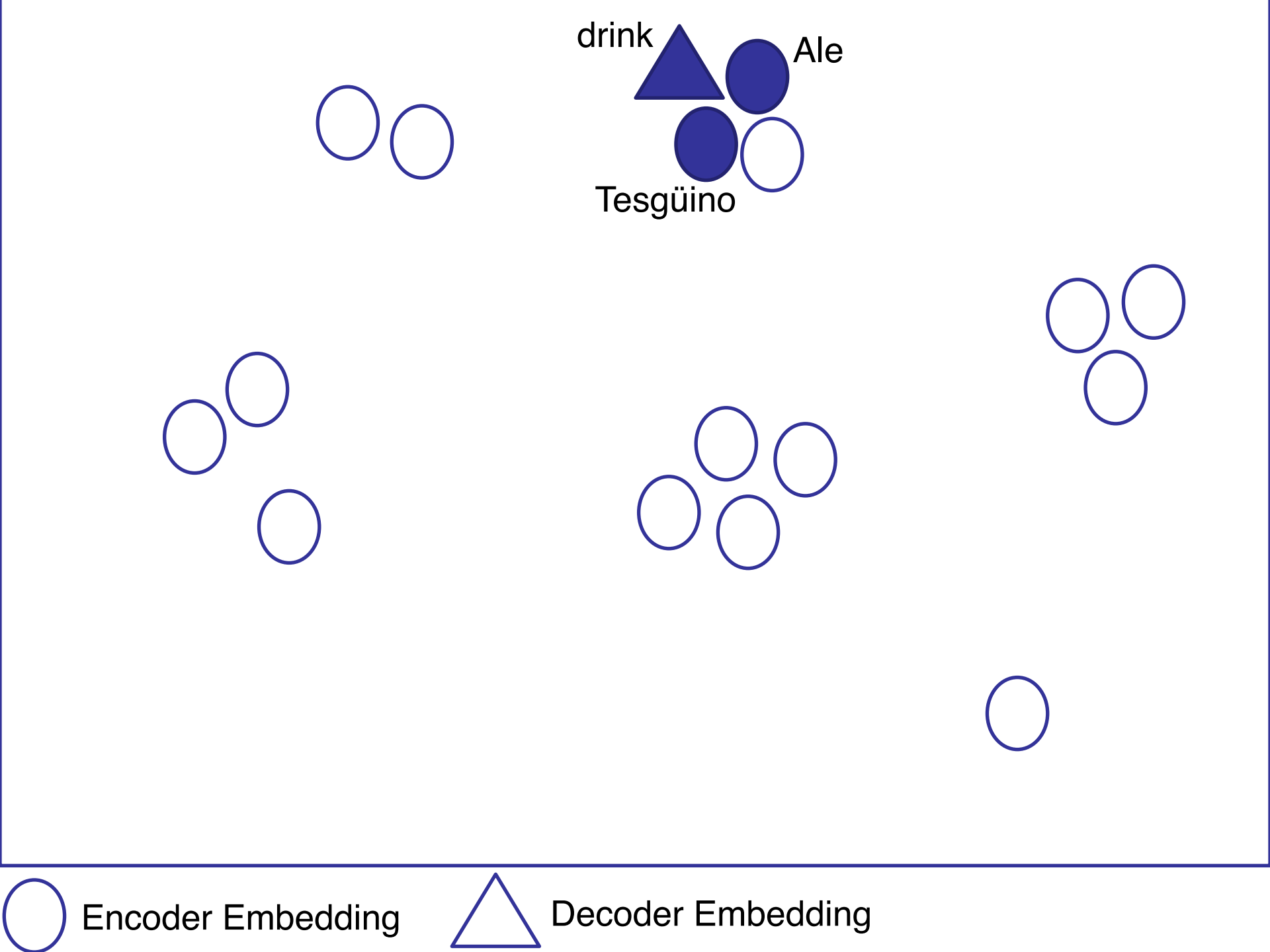
Encoder Embedding

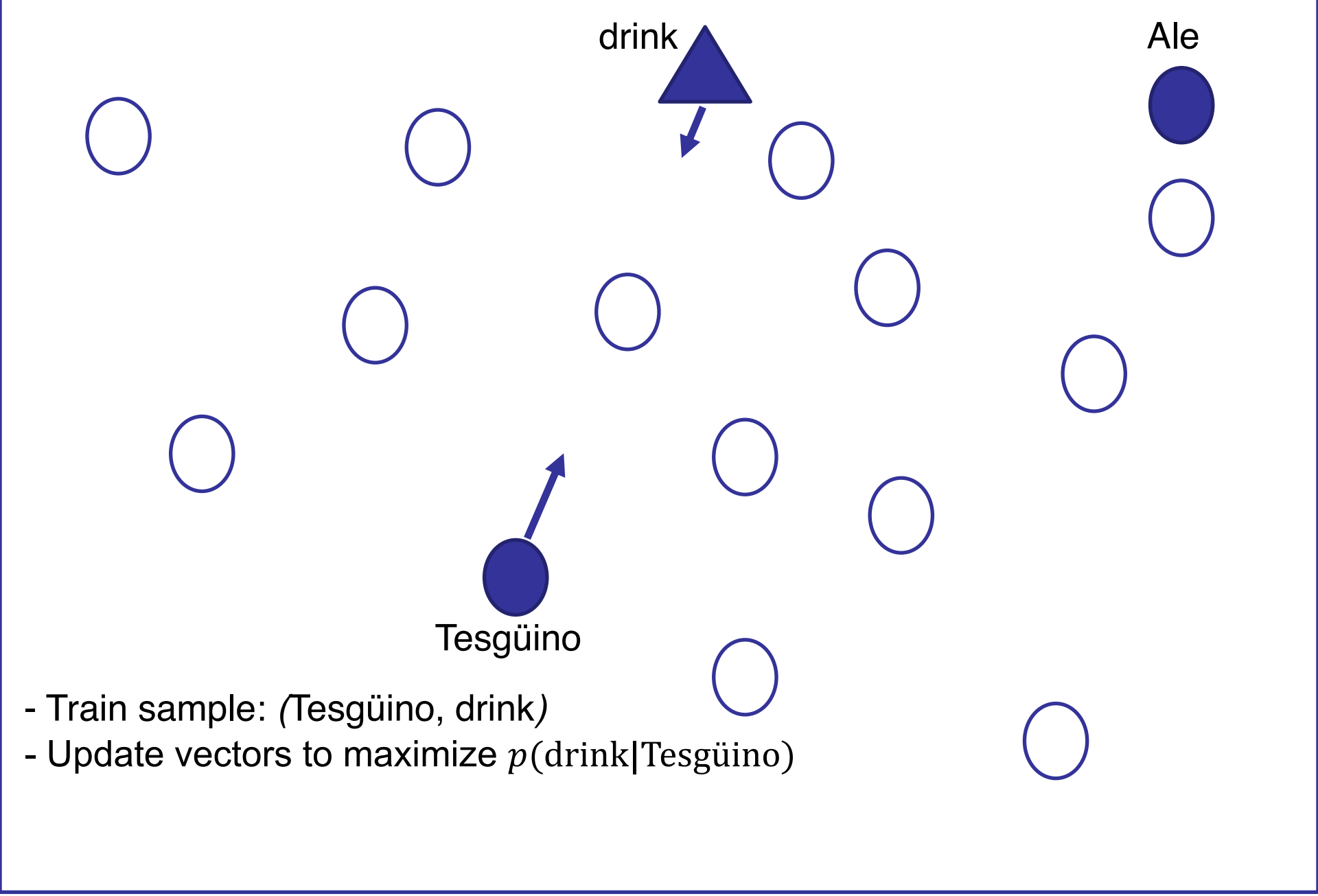
Decoder Embedding



Encoder Embedding

Decoder Embedding


drink
Ale
Tesgüino
Encoder Embedding
Decoder Embedding



- 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: $\vec{w}_{\text{Tesgüino}} \cdot \vec{c}_{\text{drink}}$

- Output layer is normalized with Softmax

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

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

Sorry! Denominator is too expensive!

- Loss function is the Negative Log Likelihood over all training samples *T*

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

word2vec Skip-Gram with Negative Sampling

- word2vec an efficient and effective algorithm

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

$$p(y=1|\mathrm{w},\mathrm{c}) = \underset{\text{sigmoid}}{\sigma}(\vec{w}_{\mathrm{w}} \cdot \vec{c}_{\mathrm{c}})$$

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

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

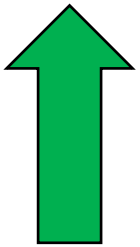
- To do it, word2vec draws *k* negative samples $\check{\mathrm{c}}$ by randomly sampling from the words distribution → why randomly?

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

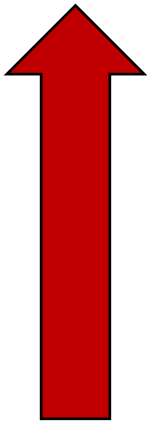
word2vec Skip-Gram with Negative Sampling

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

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

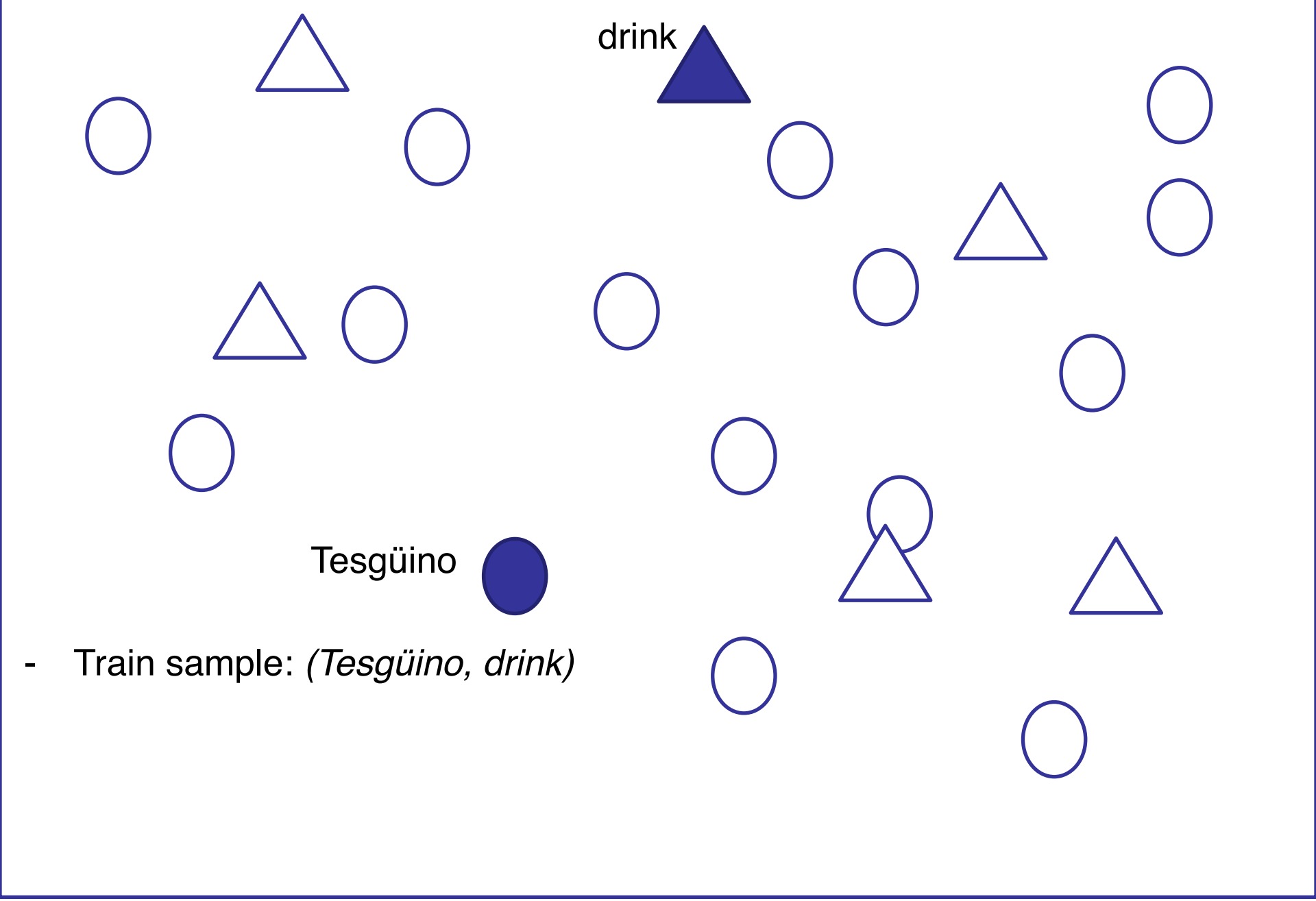


Training Samples



Negative Samples

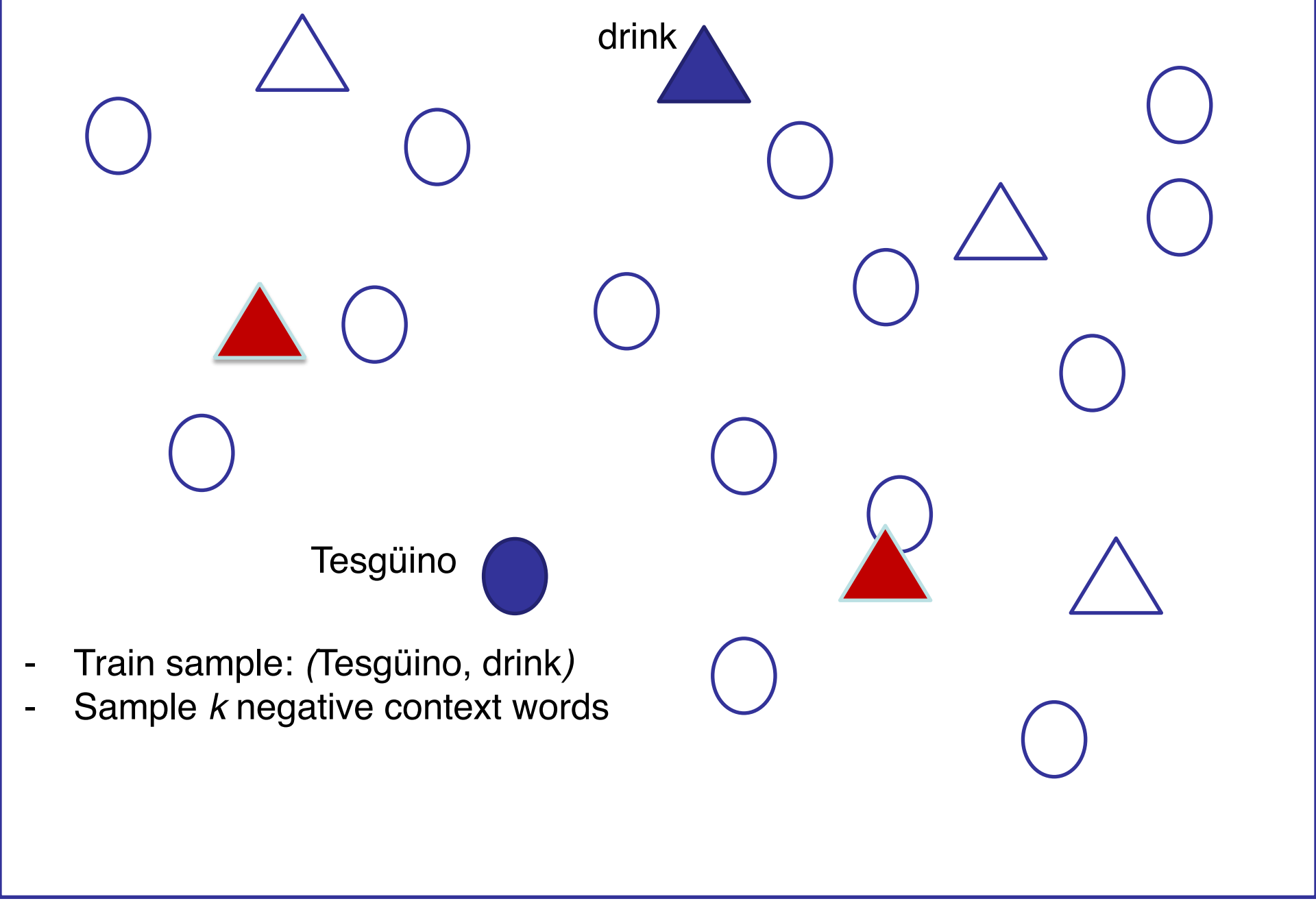
k ~ 2-10



- Train sample: *(Tesgüino, drink)*

Encoder Embedding

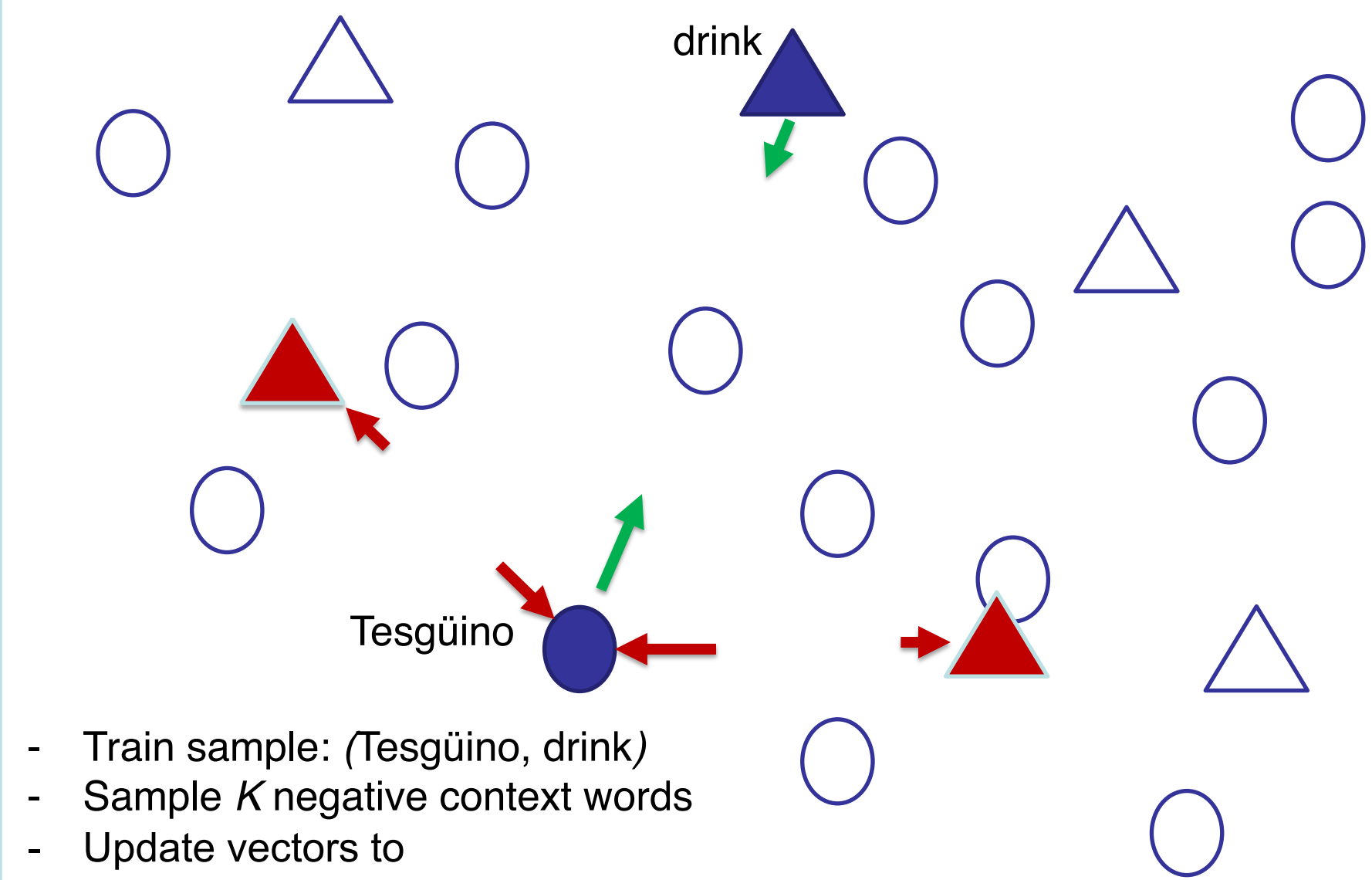
Decoder Embedding



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

Encoder Embedding

Decoder Embedding



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

Encoder Embedding

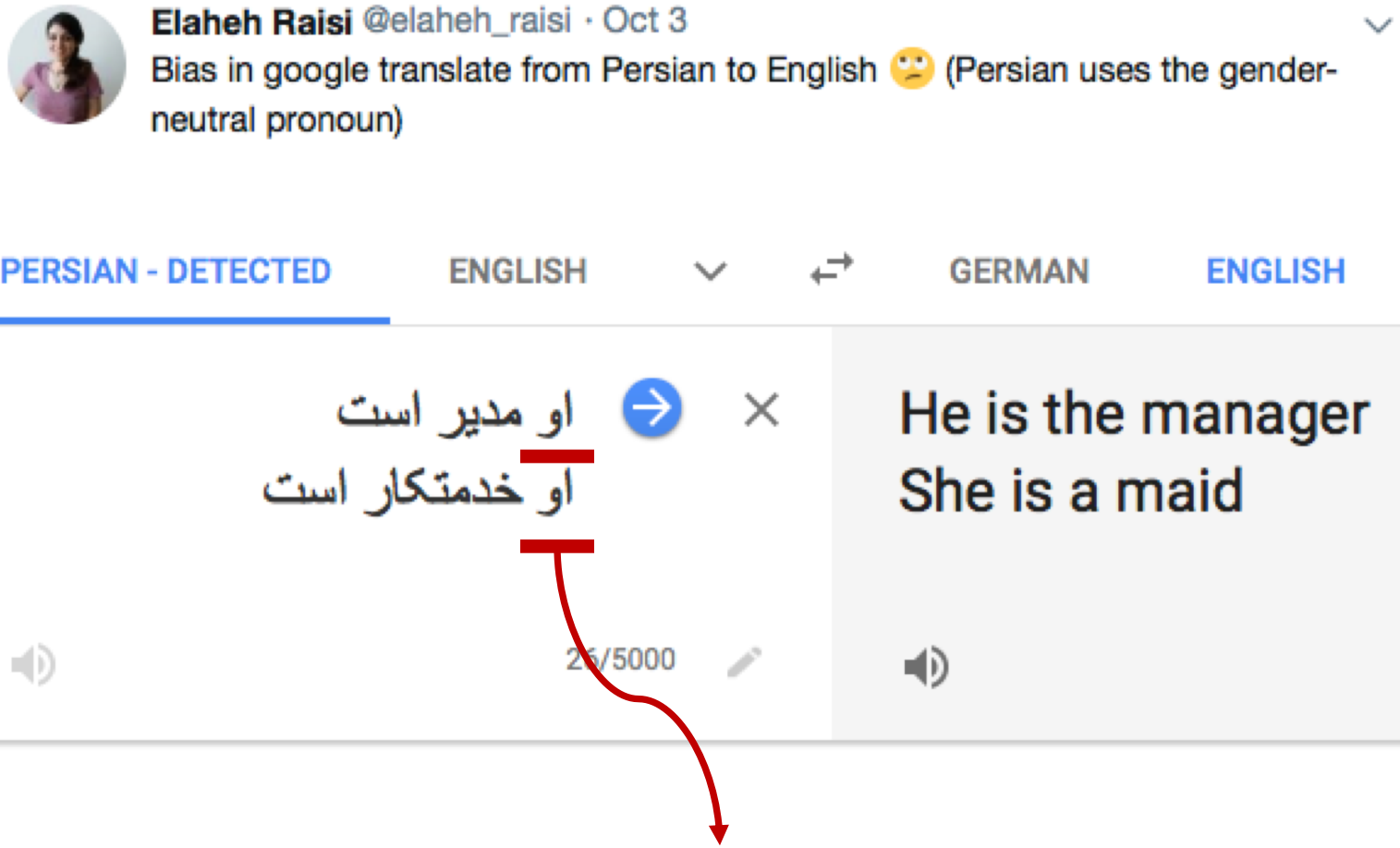
Decoder Embedding

Discussion about Bias in Data

- A word embedding model captures intrinsic patterns of the given text corpus
- If the data is biased towards a specific demographic, the embedding model also encodes it in the vectors
- This bias in word embeddings can be propagated to down-stream tasks

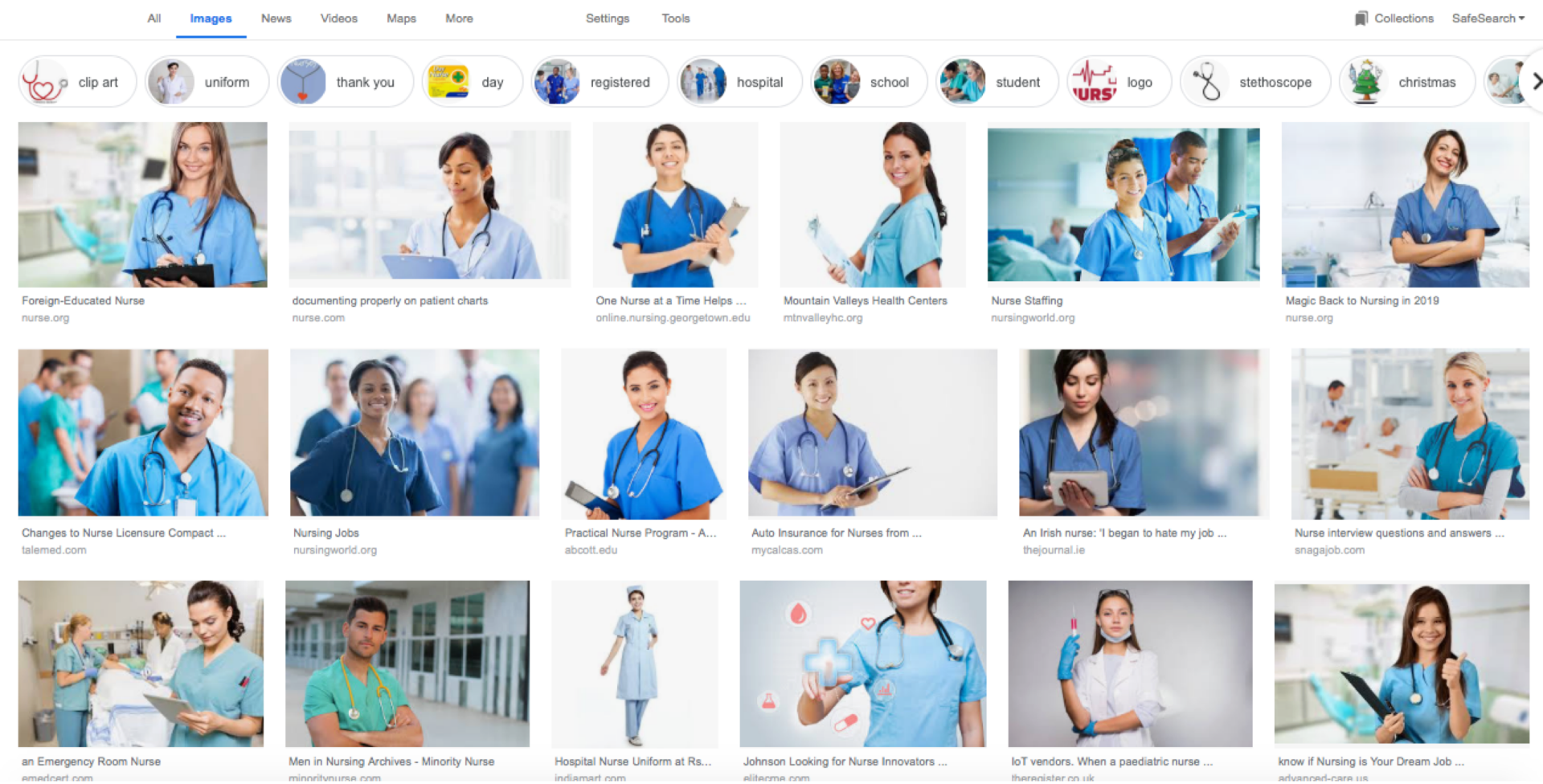
"I think your test grading is biased in favor of students who answer the test questions correctly."

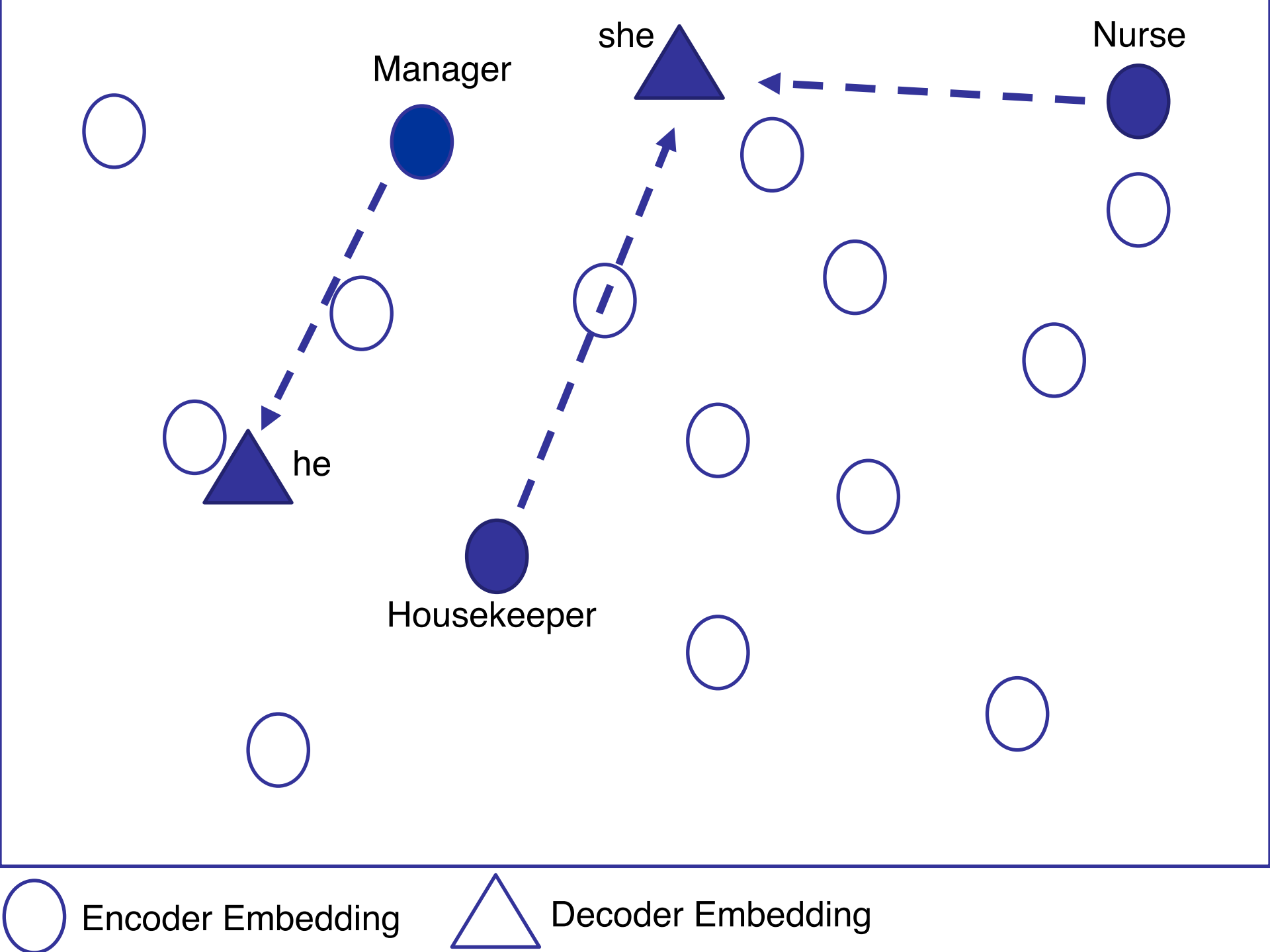
Bias in Machine Translation

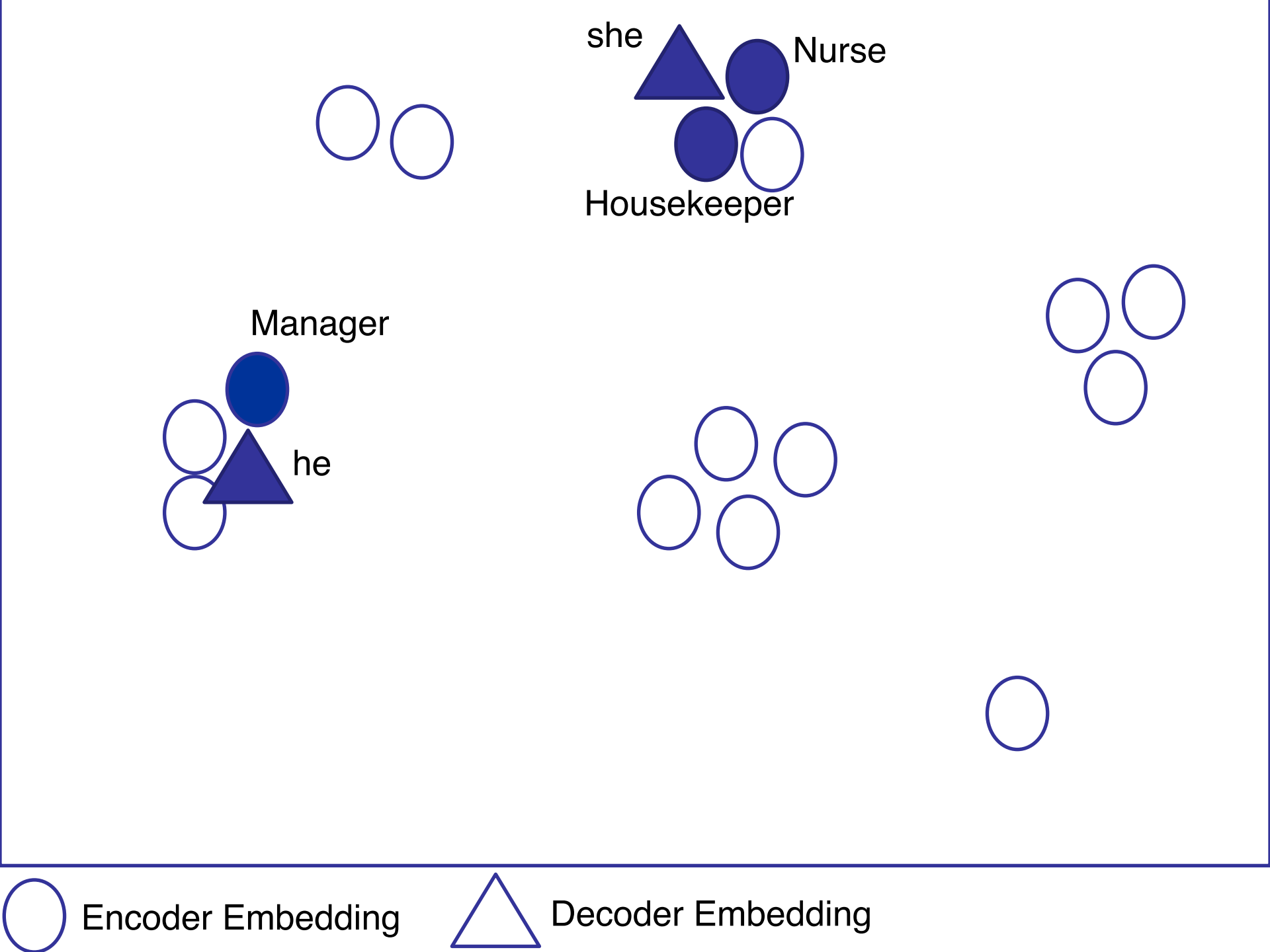


Bias in Information Retrieval

Search: nurse

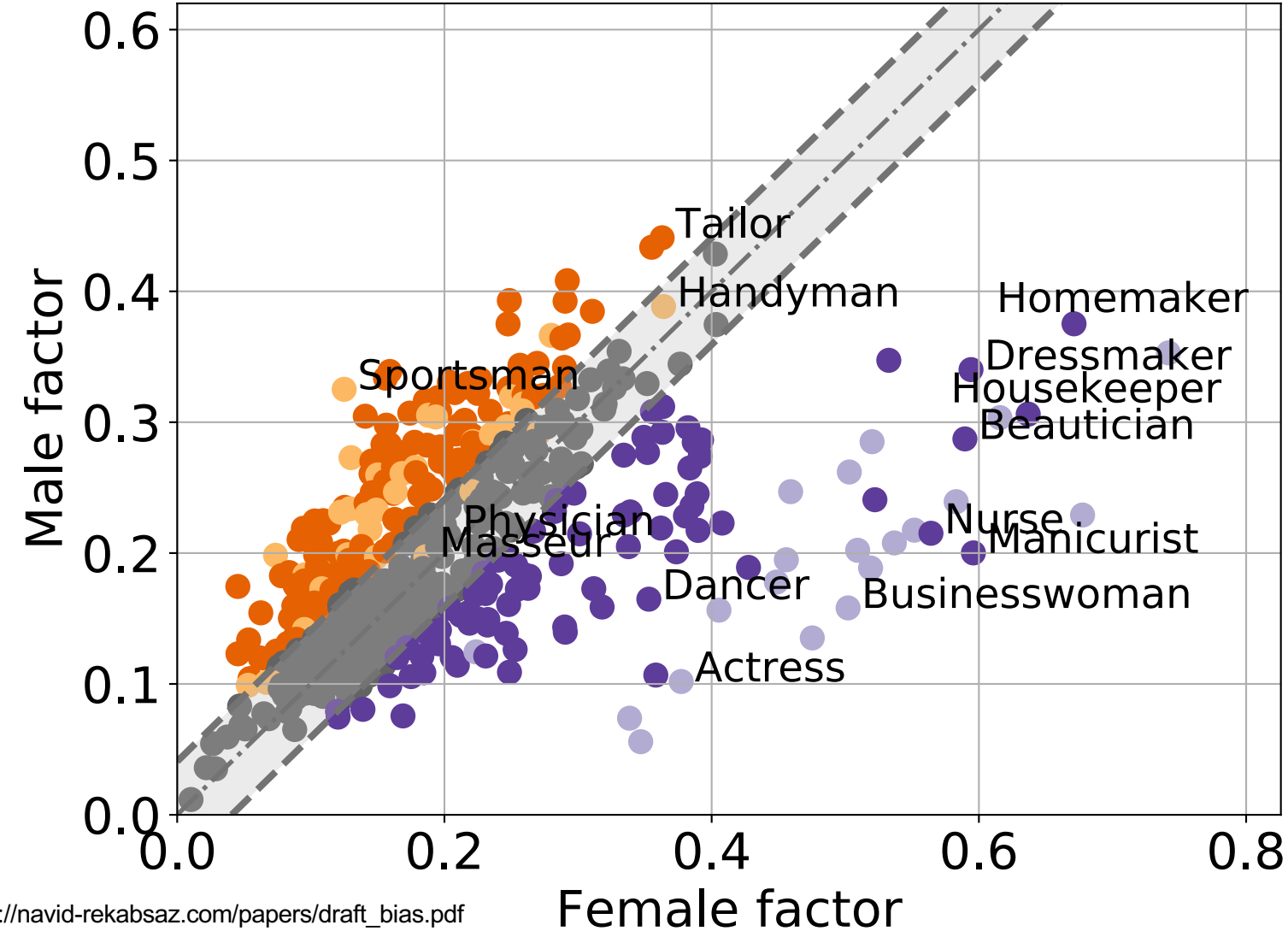



Manager
she
Nurse
he
Housekeeper
Encoder Embedding
Decoder Embedding


she
Nurse
Housekeeper
Manager
he
Encoder Embedding
Decoder Embedding

Gender Bias in Wikipedia

- The bias of 500 occupations to female/male in the word2vec Skip-Gram model, created using an English Wikipedia corpus



Pre-print: http://navid-rekabsaz.com/papers/draft_bias.pdf

Agenda

- Neural Networks - Recap
- Language Modeling
- Neural Word Embedding - Revisited
  - word2vec Skip-Gram with Negative Sampling
  - Bias in word embeddings
- Contextualized Word Embeddings
  - ELMo
  - BERT

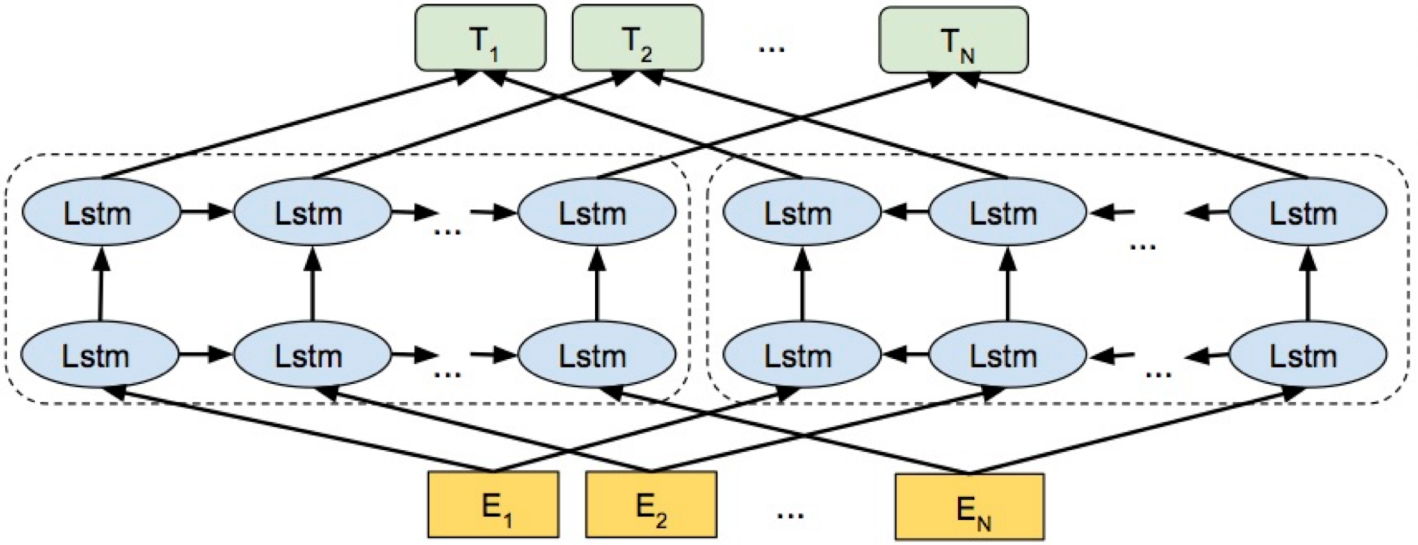
Contextualized Word Embedding

- What is *Apple*?!
  - a fruit or a company?
- The meanings of words become more clear, when we read them in their contexts
  - *eating an apple* vs. *share of the Apple company*
- In the 'standard' word embeddings, various meanings (senses) of a word are all encoded in one vector

- In contextualized word embedding, the representation of each word is defined in its context
- The input is a set of words (like a sentence) and the output is a set of vectors, each vector representing a word

ELMo (Embeddings from Language Models)

- ELMo is a Multi-layer Bidirectional LSTM



- Training is similar to training a Language Model
  - The given sentence is read one time from left to right and one time from right to left (bidirectional)
  - In each step, the model inputs a word and predicts its next word using the hidden state of the previous step

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. **Deep Contextualized Word Representations**. In *Proc. of NAACL-HTL 2018*

ELMo

- In a supervised task (such as document classification), the final representations are the weighted sum of the hidden states of the intermediary RNN layers

- For an ELMo with L layers, the representation of the word w is:

$$\text{ELMo}_w = \gamma \sum_{j=1}^{L} \theta_j \vec{s}_w^j$$

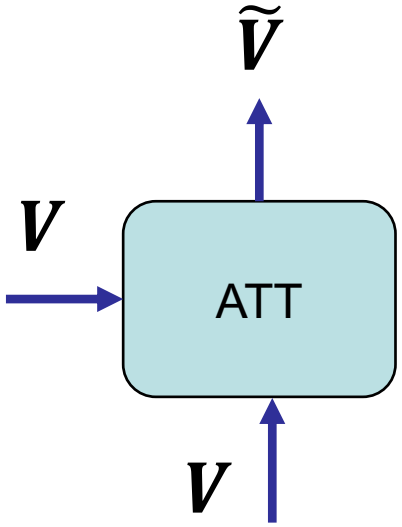
where γ and θ are parameters, learned during the task

Self-Attention Networks

- The query matrix is the same as the value matrix
- Self-attention outputs a new matrix $\widetilde{\boldsymbol{V}}$, such that each vector of $\boldsymbol{V}$ attends to all other vectors $\boldsymbol{V}$
- In contrast to RNNs, all the input vectors are fed to the model at the same time and not one by one

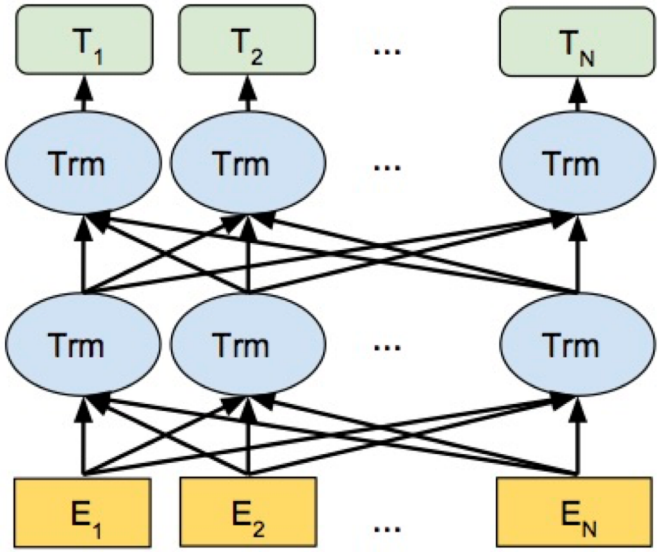
- Transformers are a special architecture of (self-)attention networks
  - Reference for more details:

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "**Attention is all you need**." In *Advances in neural information processing systems*,



BERT (Bidirectional Encoder Representation)

- Multi-layer Transformer network



- For training, for each sentence, the model masks some of the input words and tries to predict the masked ones

Input: The man went to the [MASK]$_1$. He bought a [MASK]$_2$ of milk .
Labels: [MASK]$_1$ = store; [MASK]$_2$ = gallon

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K.. **Bert: Pre-training of deep bidirectional transformers for language understanding.** In *Proc. of NAACL-HTL 2019*

BERT

- BERT is quite big!
  - **BERT-Base:** 12-layer, 110M parameters
  - **BERT-Large:** 24-layer, 340M parameters

- Though the results of fine-tuning in down-stream tasks such as Information Retrieval, Natural Language Inference, Question/Answering, Document classification, etc. are impressive

- Topics for further learning:
  - Learning sentence relations
  - Positional embeddings
  - Fine-tuning procedure