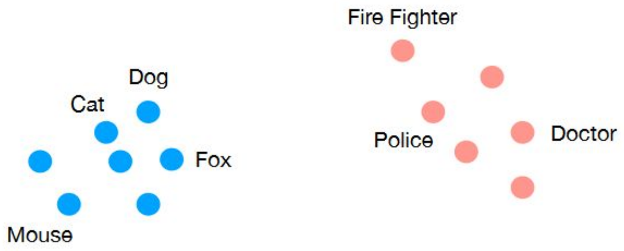
Tutorial NLP

Outline Today

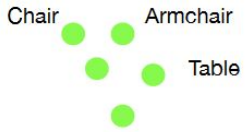
- Recap NLP Courseware
- Further questions (Last Topic/Homework/ect.)

1. Recap

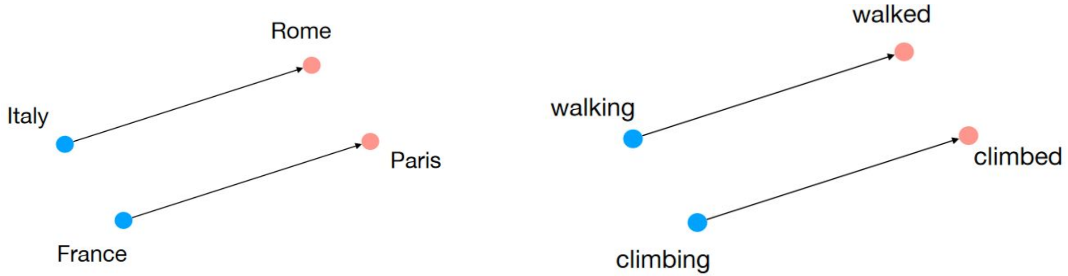
Word Embeddings



- mapping words to lower dim space → enforcing a meaningful representation, similarity is based on their distance

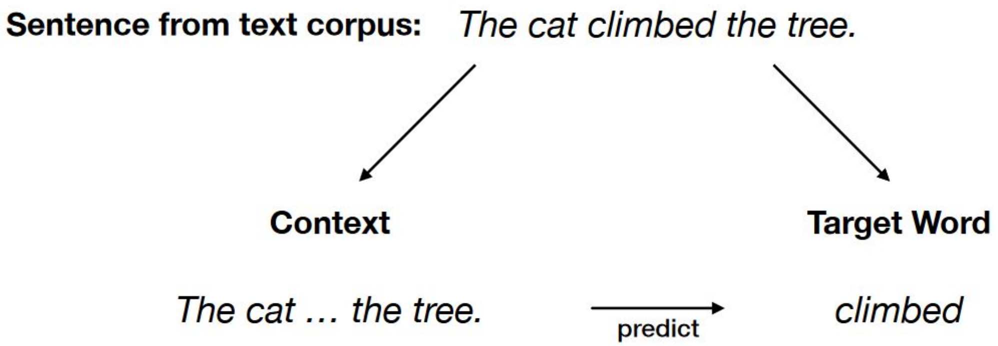


- founded on distributional hypothesis:
  Similar words occur in similar contexts.
- ideally, embedding captures semantic + syntactic information



Continuous Bag of Words

- embed a bag of words (= the context window), order does not matter
  → try to predict the target word



CBOW

Step 0: the embedding matrix

- embedding matrix is the weight matrix of encoder network (feed-forward net)
  dimension = vocab_size x hidden_size

Step 1: Receive the embedding of the context words

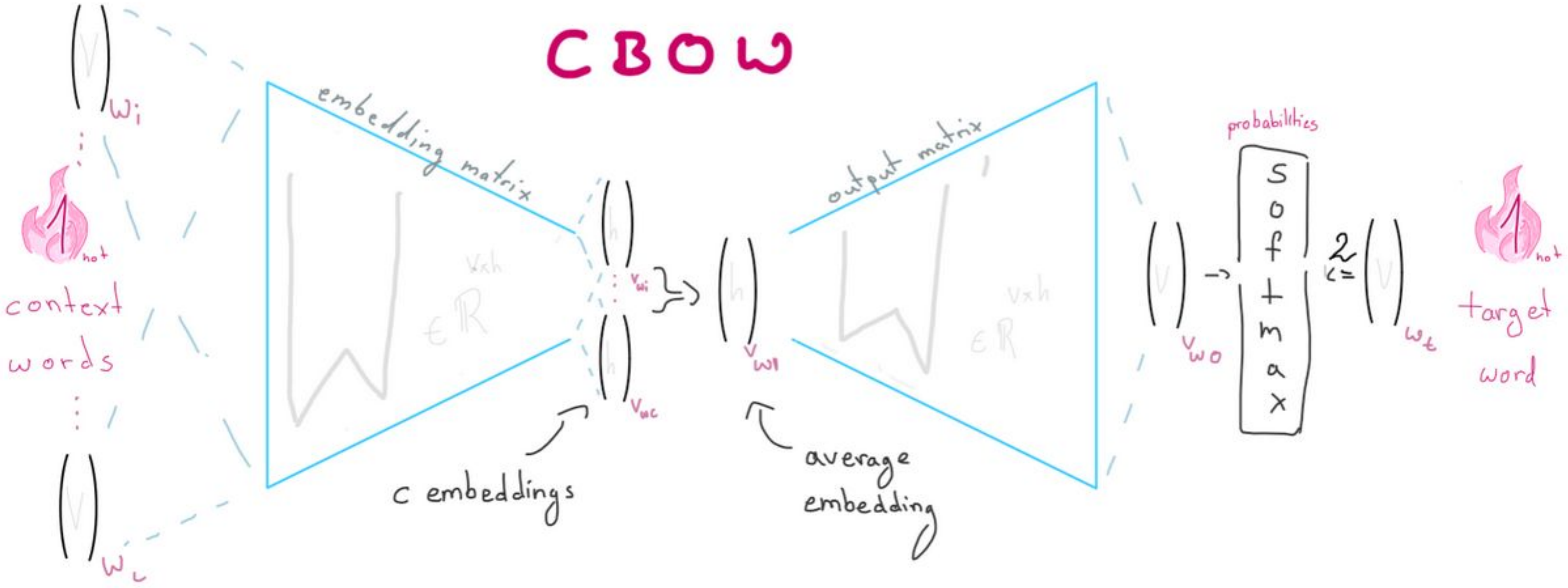
- matrix multiply one-hot encoded word with the embedding matrix

Step 2: Predict the target word from the context

- average embedded context words
- pass it to “decoder” feed-forward network → learns to map the embedding to score vector
- scoring vector (size = vocab_size) → softmax → pred probabilities for each word

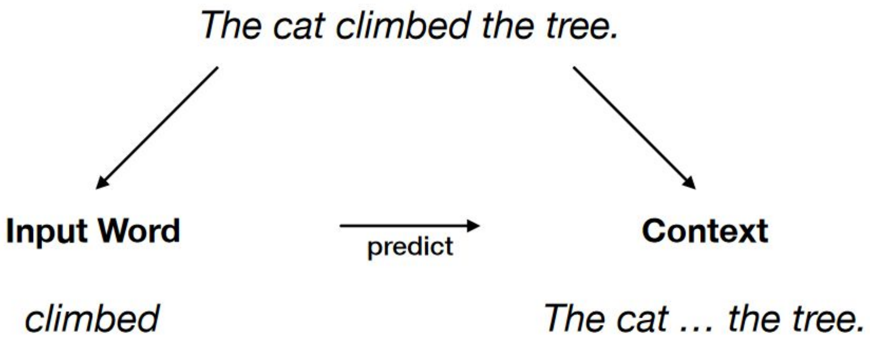
Step 3: Improving the embedding

- get prediction, compare to real target word, compute cross-entropy-loss, gradient descent as usual

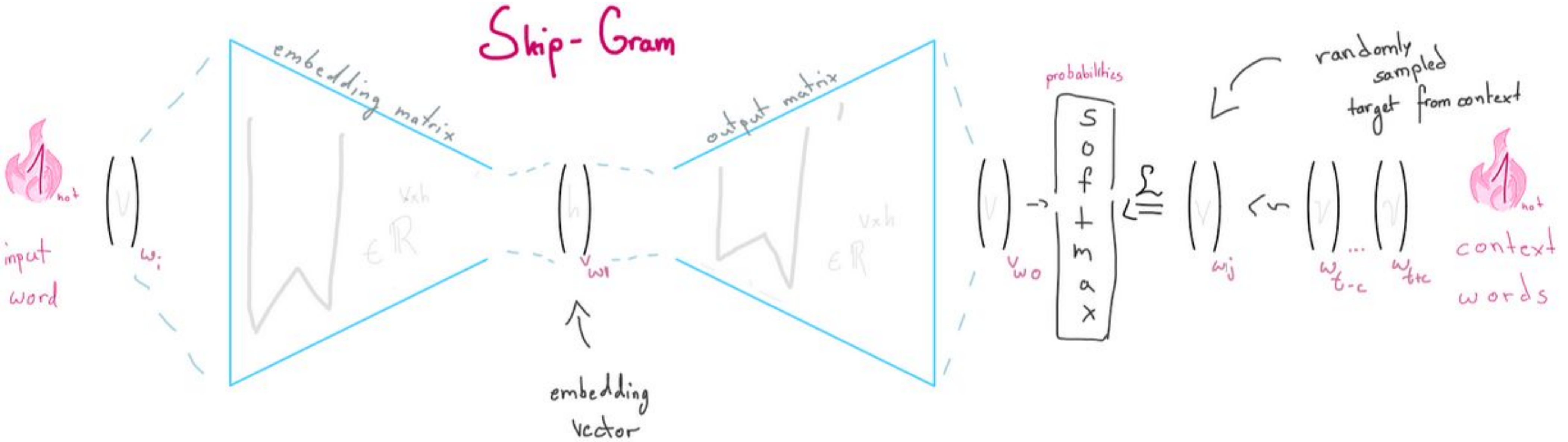

CBOW
embedding matrix
output matrix
probabilities
softmax
context words
target word
c embeddings
average embedding
1 hot
1 hot

SkipGram

- predict a set of context words from one word



Input	Target
climbed	*the*
climbed	*cat*
climbed	*the*
climbed	*tree*



Skip-Gram
embedding matrix
output matrix
probabilities
randomly sampled target from context
input word
context words
embedding vector
softmax

SkipGram

given: text corpus of vocab size V , context window size c

preprocessing: one_hot encode corpus, generate input-context pairs $(w_i, w_{i-c} \ldots . w_{i+c})$

for each input context pair:

- from context $(w_{i-c} \ldots w_{i+c})$ sample one target (uniform or distance based) w_t
- compute embedding v_{w_I} for input word using embedding matrix W
- compute score vector v'_{W_I} using output matrix W'
- compute softmax of score to obtain probabilities:
    - $p(w_t \sim P_c|w_i) = \frac{exp(v'^{\top}_{w_I} v_{w_I})}{\sum_{i=1}^{V} exp(v'_{w_{I_i}} \top v_{w_I})}$j
- compute loss using cross-entropy
    - $\mathcal{L}_{CBOW} = -\log p(w_t \sim P_c|w_i)$
- minimize loss using gradient descent

SkipGram

given: text corpus of vocab size V , context window size c

preprocessing: one_hot encode corpus, generate input-context pairs $(w_i, w_{i-c} \dots . w_{i+c})$

→ in practise only use the index and use lookup

for each input context pair:

- from context $(w_{i-c} \dots w_{i+c})$ sample one target (uniform or distance based) w_t
- compute embedding v_{w_I} for input word using embedding matrix W
- compute score vector v'_{W_I} using output matrix W'
- compute softmax of score to obtain probabilities:
  - $p(w_t \sim P_c | w_i) = \frac{exp(v'^{\top}_{w_I} v_{w_I})}{\sum_{i=1}^{V} exp(v'_{w_{I_i}} {}^{\top} v_{w_I})}$j
- compute loss using cross-entropy
  - $\mathcal{L}_{CBOW} = -\log p(w_t \sim P_c | w_i)$
- minimize loss using gradient descent

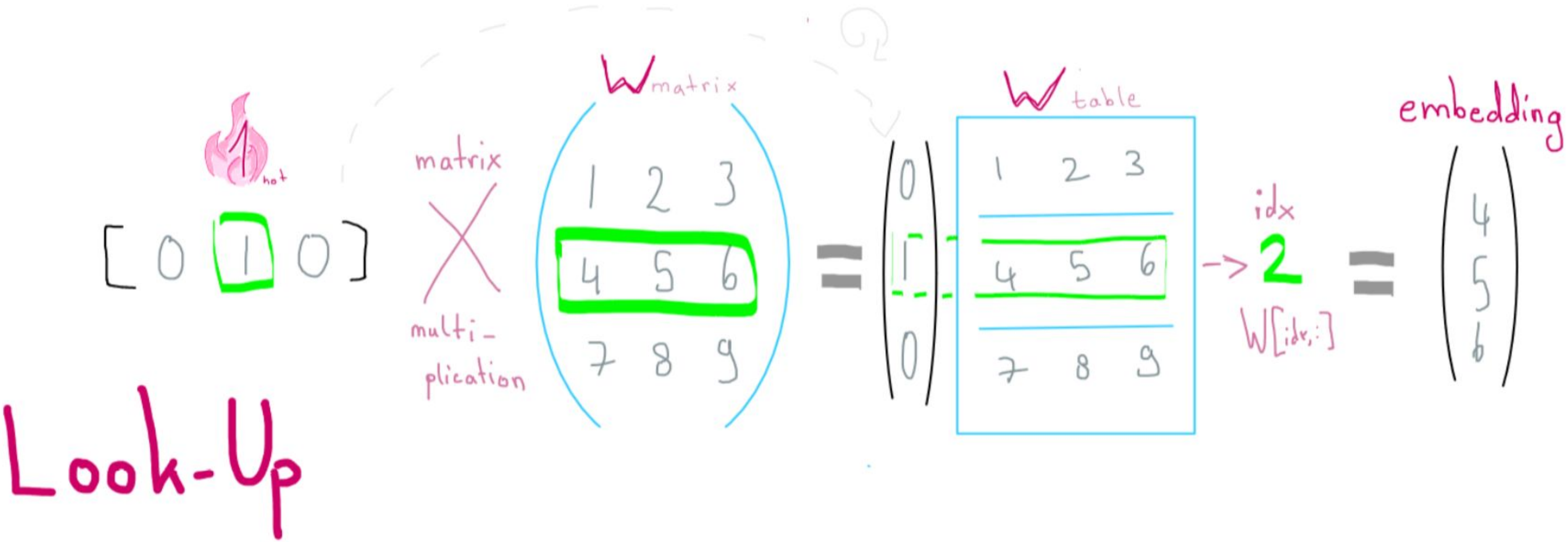
→ alternative: use each input-target pair (This is how we have described it in the homework sheet, you can decide how you want to do it.)

NGRAM

- takes into account order of the words
- The cat eats the cake. $\neq$ The cake eats the cat.
- approximate $p(w_i|w_{i-k}, ..., w_{i-1}, w_{i+1}, ..., w_{i+k})$
-

Receiving Embeddings

- lookup computationally more efficient than matrix multiplication



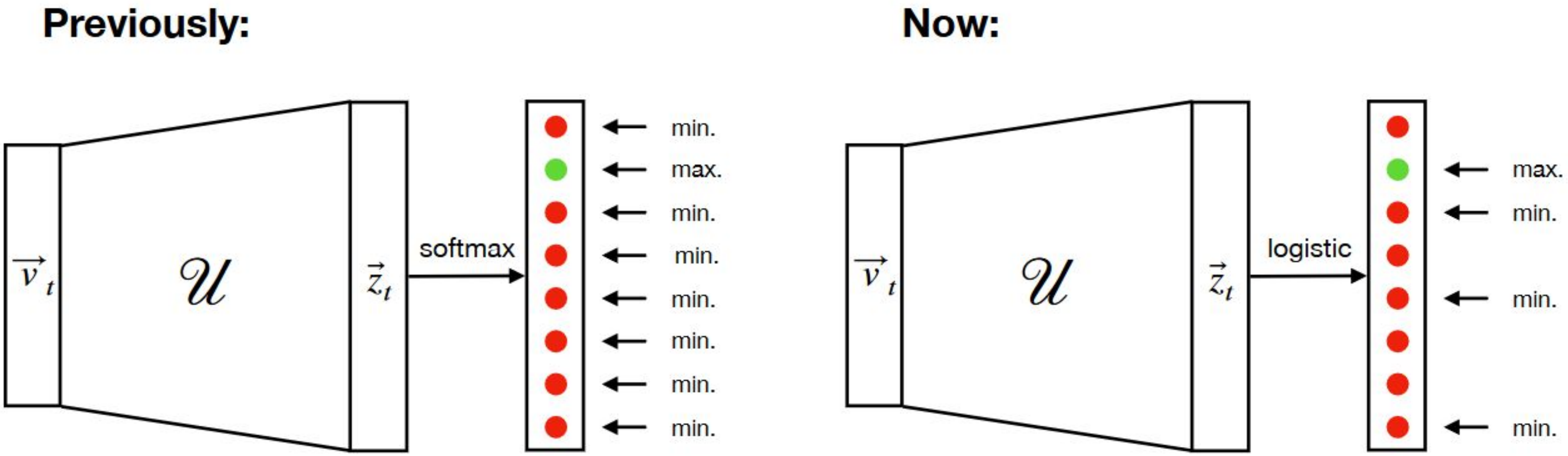
Subsampling

- removing frequent words like ‘the’, ‘a’, ‘and’
- probability to discard a word based on its frequency $z(w_i)$ and a hyperparameter s (often s = 0.001)

$$P(w_i) = (\sqrt{\frac{z(w_i)}{s}} + 1) \cdot \frac{s}{z(w_i)}$$

Negative Sampling

- computationally expensive to minimize all outputs, therefore sample a few to minimize
- use logistic loss instead of softmax because softmax has to sum up to 1

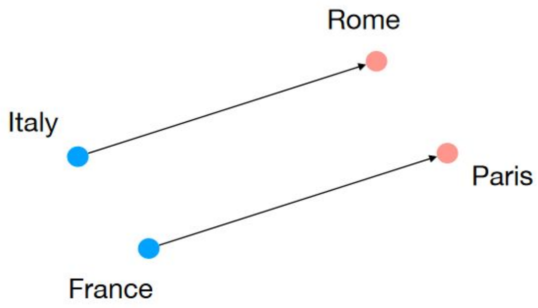


Distances and Analogies

- calculate **distances** between vectors using cosine similarity
- cosine similarity: inner product of normed vectors

- dimension reduction with maintaining the distances → t-sne

Semantic analogies



$$\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{king}} - \overrightarrow{\text{queen}}$$

$$\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{computer programmer}} - \overrightarrow{\text{homemaker}}. \quad ?$$

Why else might that be interesting?

Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings, Bolukbasi et al.
https://arxiv.org/pdf/1607.06520.pdf

Supervised NLP

Sequence-to-Classification

- e.g. sentiment analysis - “Is this movie review positive or negative”
- or language identification - “what language is this text?”

Sequence-to-Sequence

- e.g. : Text summarization - “Summarize this paper in 2 sentences.”
- question answering “Who was the worst president of the USA?”
- translation “Here is a text on how to correctly pet your cat, translate it to German please”.

Further questions?