POLITECNICO
MILANO 1863

Artificial Neural Networks and Deep Learning

- Word Embedding-

Matteo Matteucci, PhD (matteo.matteucci@polimi.it)
Artificial Intelligence and Robotics Laboratory
Politecnico di Milano

AIRLAB
ARTIFICIAL INTELLIGENCE AND ROBOTICS LAB

Recall Machine Learning Paradigms

Immagine you have a certain experience E, and let's name it

$$D = x_1, x_2, x_3, \dots, x_N$$

- ***Supervised learning***: given the desired outputs $t_1, t_2, t_3, \dots, t_N$ learn to produce the correct output given a new set of input
- ***Unsupervised learning***: exploit regularities in D to build a representation to be used for reasoning or prediction
- ***Reinforcement learning***: producing actions $a_1, a_2, a_3, \dots, a$ the environment, and receiving rewards $r_1, r_2, r_3, \dots, r_N$ to maximize rewards in the long term

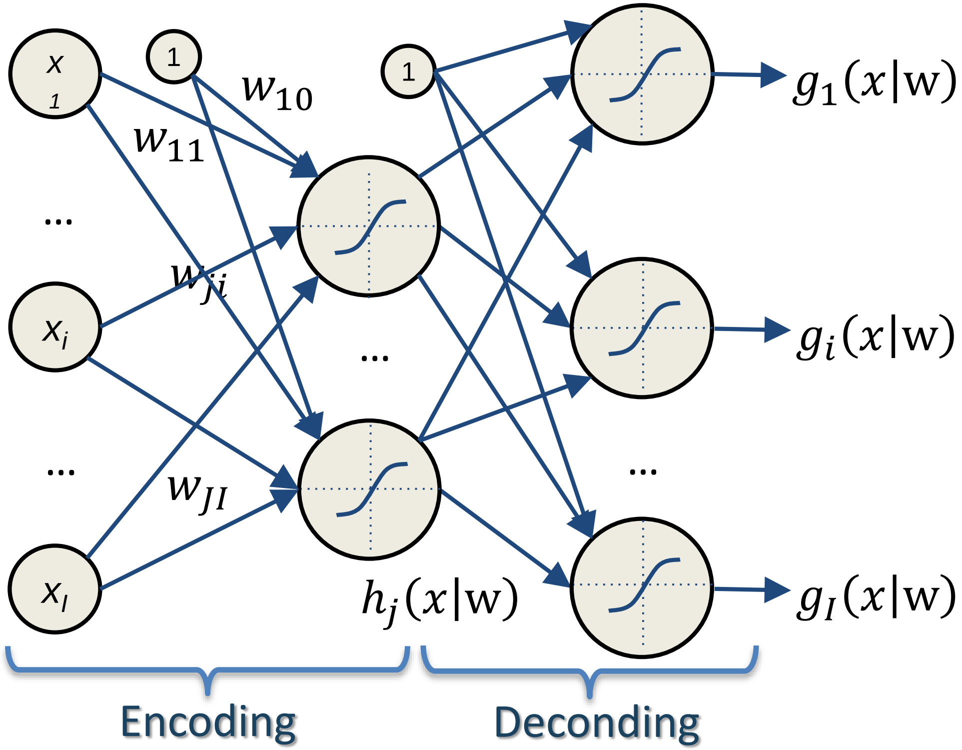
Haven't seen much of it, is it?

This course focuses mainly on Supervised and Unsupervised Learning …

Neural Autoencoder Recall

Network trained to output the input (i.e., to learn the identity function)

- Limited number of units in hidden layers (compressed representation)
- Constrain the representation to be sparse (sparse representation)



$$x \in \Re^I \xrightarrow{enc} h \in \Re^J \xrightarrow{dec} g \in \Re^I$$

$$I << J$$

$$E = \underbrace{\|g_i(x_i|w) - x_i\|^2}_{\text{Reconstruction error}} + \lambda \underbrace{\sum_j \left| h_j\left(\sum_i w_{ji}^{(1)} x_i\right)\right|}_{\text{Sparsity term}}$$

$$g_i(x_i|w) \sim x_i \qquad h_j(x_i|w) \sim 0$$

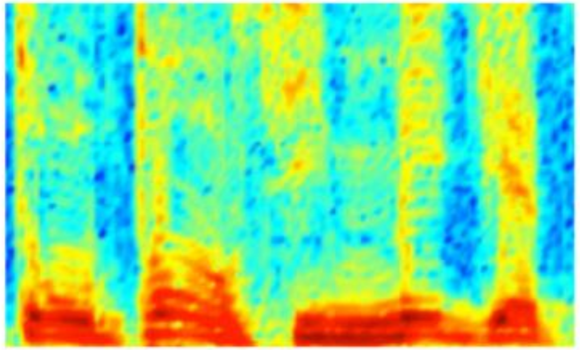
Word Embedding Motivation

Natural language processing treats words as discrete atomic symbols

- 'cat' is encoded as Id537
- 'dog' is encoded as Id143
- …

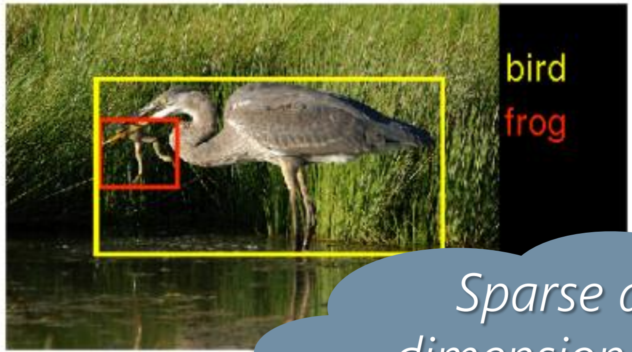
Items in a dictionary …

A document becomes a Bag of Words



Audio Spectrogram

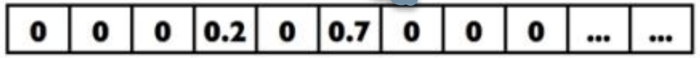
DENSE



Sparse and high dimensional -> Curse of Dimensionality!

Image pixels

DENSE



Word, context, or document vectors

SPARSE

Encoding Text is a Serious Thing

Performance of real-world applications (e.g., chatbot, document classifiers, information retrieval systems) depends on input encoding:

Local representations

- N-grams $\xrightarrow{\text{Language Model}}$
- Bag-of-words
- 1-of-N coding

Continuous representations

- Latent Semantic Analysis
- Latent Dirichlet Allocation
- Distributed Representations

Determine $P(s = w_1, \dots, w_k)$ in some domain of interest

$$P(s_k) = \prod_{i}^{k} P(w_i | w_1, \dots, w_{i-1})$$

In traditional n-gram language models "the probability of a word depends only on the context of n−1 previous words"

$$\hat{P}(s_k) = \prod_{i}^{k} P(w_i | w_{i-n+1}, \dots, w_{i-1})$$

Typical ML-smoothing learning process (e.g., Katz 1987):

- compute $\hat{P}(w_i | w_{i-n+1}, \dots, w_{i-1}) = \frac{\#w_{i-n+1}, \dots, w_{i-1}, w_i}{\#w_{i-n+1}, \dots, w_{i-1}}$
- smooth to avoid zero probabilities

N-gram Language Model: Curse of Dimensionality

Let’s assume a 10-gram LM on a corpus of 100.000 unique words

- The model lives in a 10D hypercube where each dimension has 100.000 slots
- Model training ↔ assigning a probability to each of the 100.000^{10} slots
- *Probability mass vanishes* → more data is needed to fill the huge space
- The more data, the more unique words! → Is not going to work ...

In practice:

- Corpuses can have 10^6 unique words
- Contexts are typically limited to size 2 (trigram model), e.g., famous Katz (1987) smoothed trigram model
- With short context length a lot of information is not captured

N-gram Language Model: Word Similarity Ignorance

Let assume we observe the following similar sentences

- *Obama speaks to the media in Illinois*
- *The President addresses the press in Chicago*

With classic one-hot vector space representations

- speaks = [0 0 1 0 ... 0 0 0 0]
- addresses = [0 0 0 0 ... 0 0 1 0]
- obama = [0 0 0 0 ... 0 1 0 0]
- president = [0 0 0 1 ... 0 0 0 0]
- illinois = [1 0 0 0 ... 0 0 0 0]
- chicago = [0 1 0 0 ... 0 0 0 0]

speaks ⊥ addresses

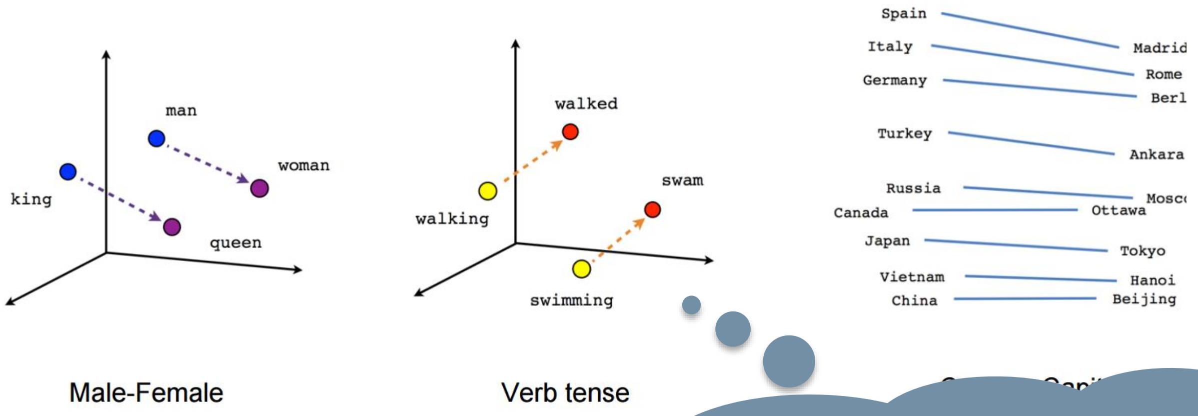
obama ⊥ president

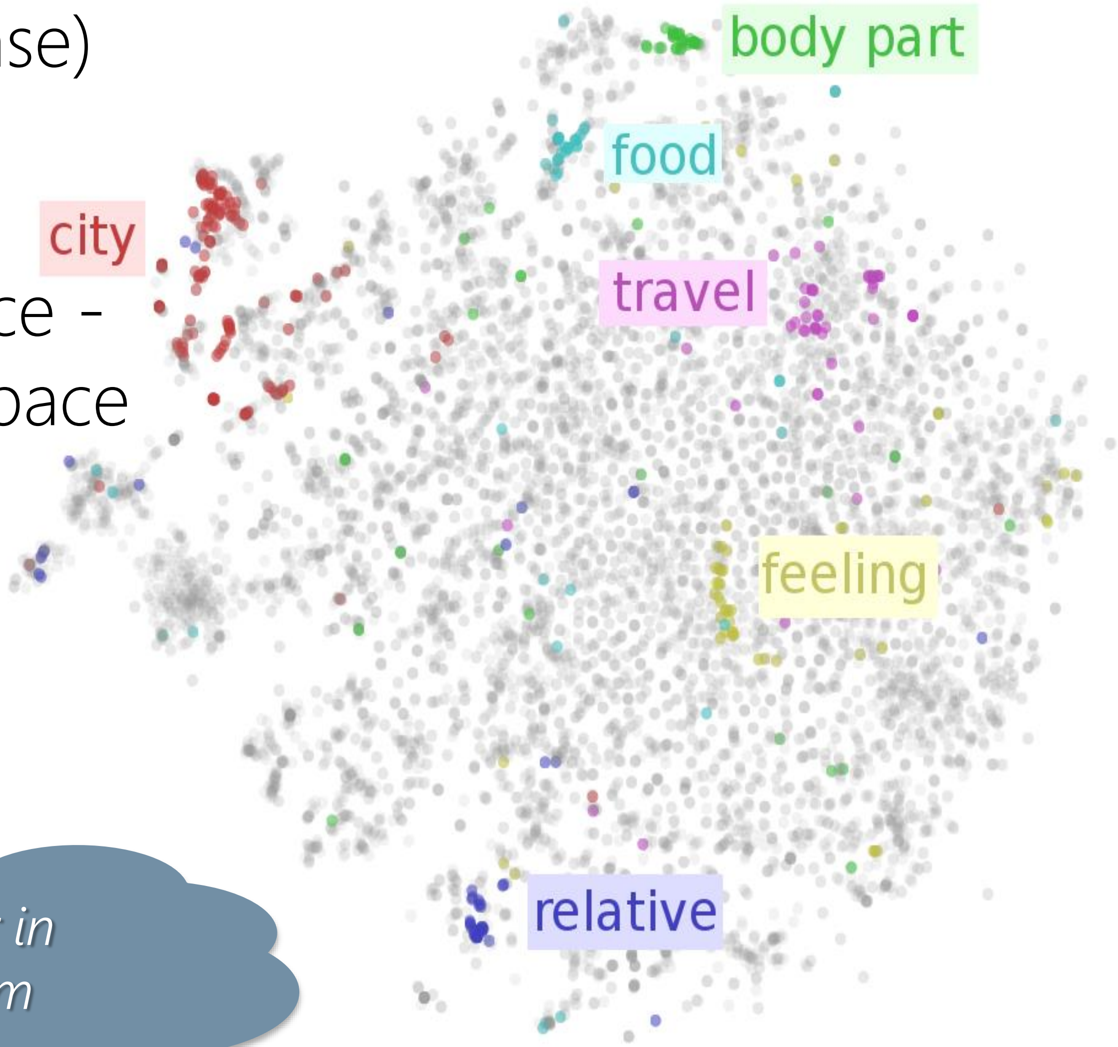
illinois ⊥ chicago

Word pairs share no similarity, and we need word similarity to generalize

Embedding

Any technique mapping a word (or phrase) from it's original high-dimensional input space (the body of all words) to a lower-dimensional numerical vector space - so one *embeds* the word in a different space





Closer points are closer in meaning and they form clusters …

Word Embedding: Distributed Representation

Each unique word $\boldsymbol{w}$ in a vocabulary V (typically $\|V\| > 10^6$) is mapped to a continuous m-dimensional space (typically $100 < m < 500$)

$$w \in V \xrightarrow{mapping\ C} \Re^m$$

$$\text{obama} = \underbrace{\begin{array}{c} \begin{array}{ccc} w_1 & \text{obama} & w_V \end{array} \\ [0\ 0 \ldots 0\ 1\ 0 \ldots 0\ 0] \end{array}}_{\text{«one-hot» encoding}} \Rightarrow \text{obama} = \underbrace{\begin{array}{c} \begin{array}{cc} f_1 & f_m \end{array} \\ [0.12 \ldots -0.25] \end{array}}_{\text{feature vector}}$$

Fighting the curse of dimensionality with:

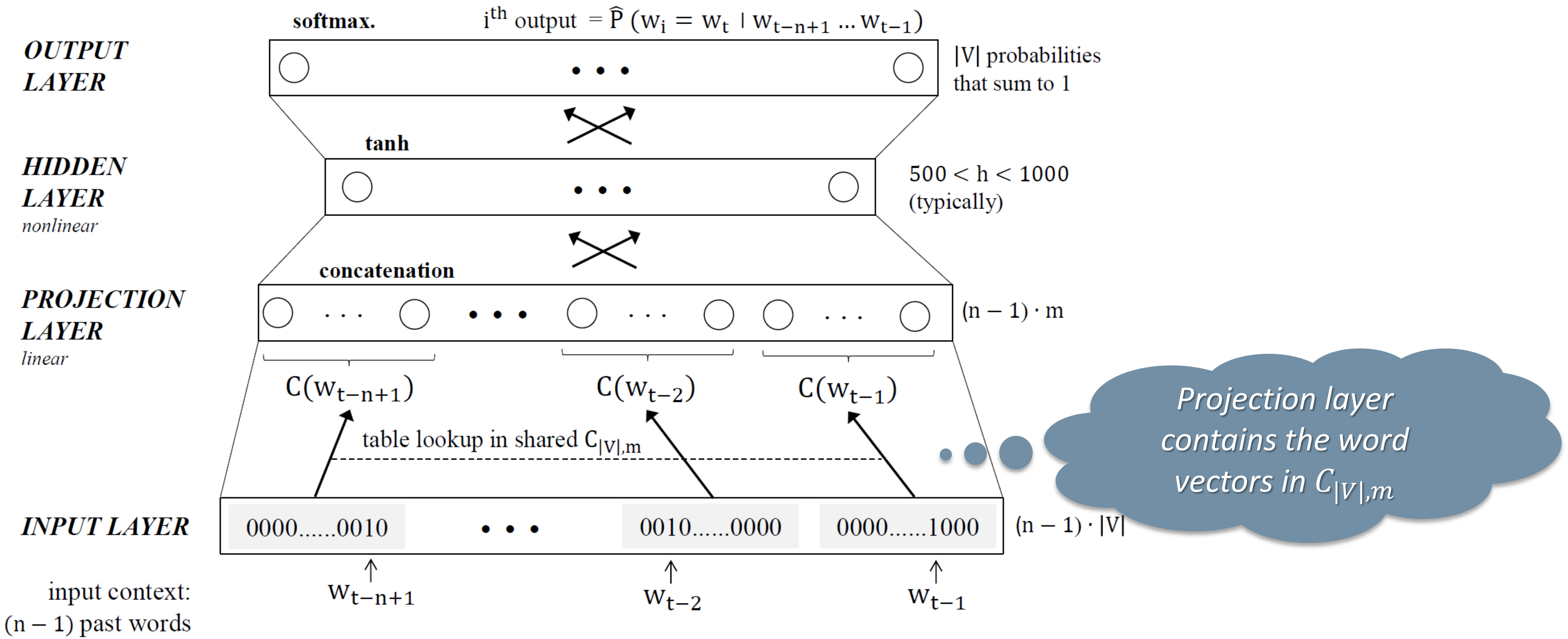
- Compression *(dimensionality reduction)*
- Smoothing *(discrete to continuous)*
- Densification *(sparse to dense)*

Similar words should end up to be close to each other in the feature space …

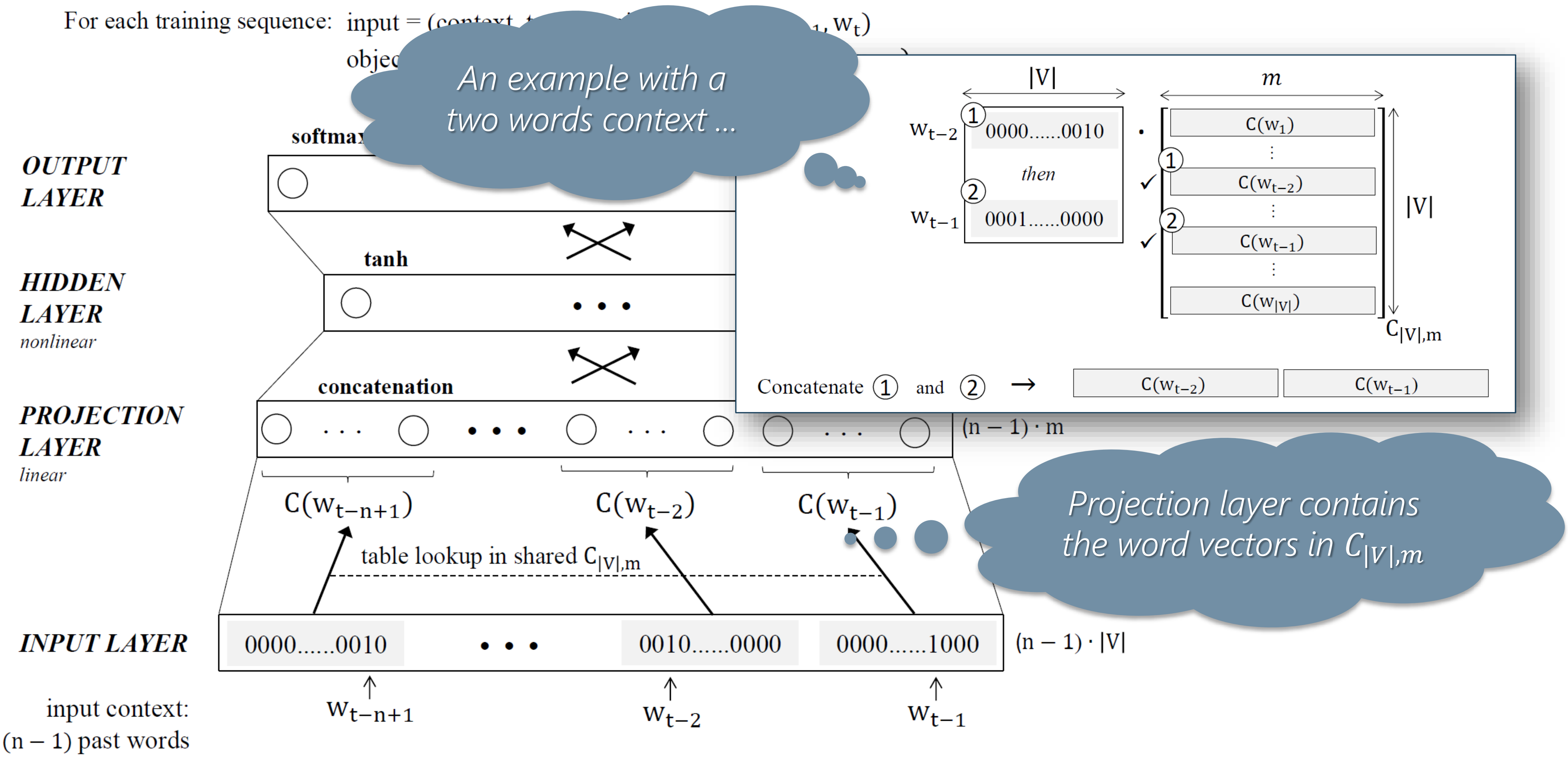
Neural Net Language Model (Bengio et al. 2003)

For each training sequence: input = (context, target) pair: $(w_{t-n+1} \dots w_{t-1}, w_t)$

objective: minimize $E = -\log \widehat{P}(w_t \mid w_{t-n+1} \dots w_{t-1})$



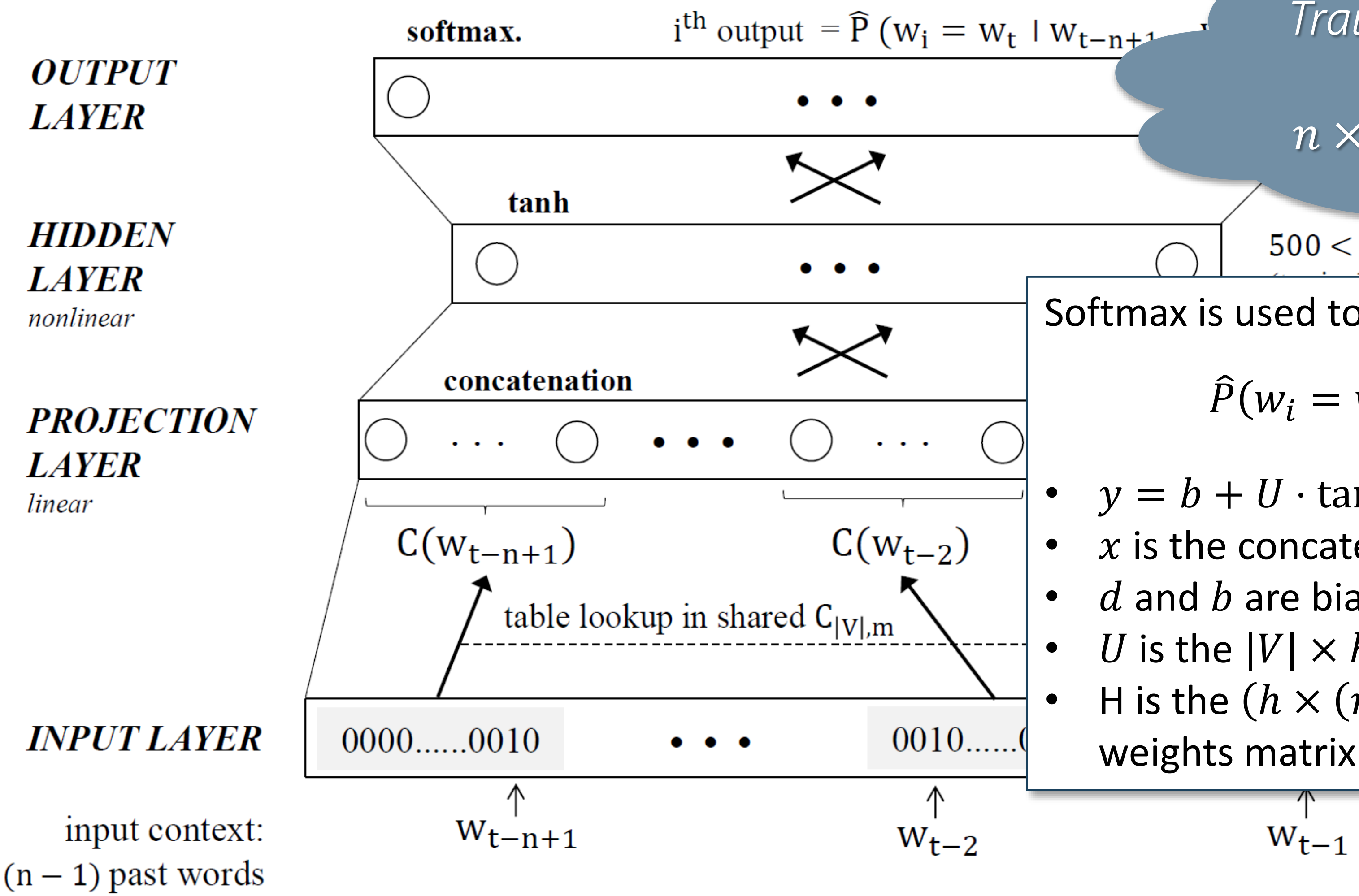
Neural Net Language Model (Bengio et al. 2003)



Neural Net Language Model (Bengio et al. 2003)

For each training sequence: input = (context, target) pair: $(w_{t-n+1} \dots w_{t-1}, w_t)$

objective: minimize $E = -\log \hat{P}(w_t \mid w_{t-n+1} \dots w_{t-1})$



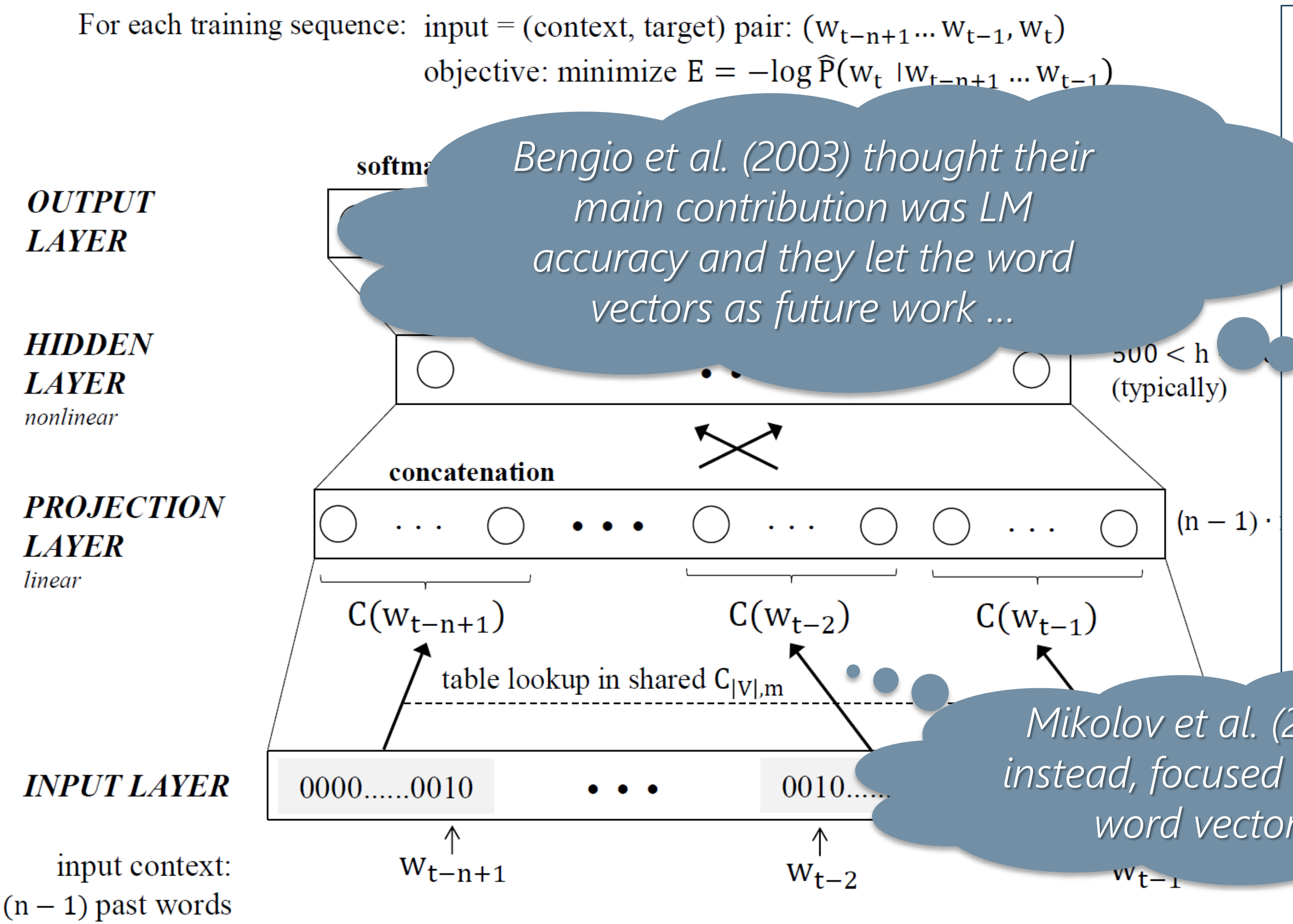
Training by stochastic gradient descent has complexity
$n \times m + n \times m \times h + \boldsymbol{h} \times |\boldsymbol{V}|$

Softmax is used to output a multinomial distribution

$$\hat{P}(w_i = w_t \mid w_{t-n+1}, \dots, w_{t-1}) = \frac{e^{y_{w_i}}}{\sum_{i'}^{|V|} e^{y_{w_{i'}}}}$$

- $y = b + U \cdot \tanh(d + H \cdot x)$
- x is the concatenation $C(w)$ of the context weight vectors
- d and b are biases (respectively h and $|V|$ elements)
- U is the $|V| \times h$ matrix with hidden-to-output weights
- H is the $(h \times (n-1) \cdot m)$ projection-to-hidden weights matrix

Neural Net Language Model (Bengio et al. 2003)



Bengio et al. (2003) thought their main contribution was LM accuracy and they let the word vectors as future work …

Tested on Brown (1.2M words, V≅ 16K, 200K test set) and AP News (14M words, V≅ 150K reduced to 18K, 1M test set)

- Brown: h=100, n=5, m=30
- AP News: h=60, n=6, m=100
- **3 week** training using **40 cores**
- 24% (Brown) and 8% (AP News) relative improvement wrt traditional smoothed n-gram in terms of test set perplexity

Due to **complexity**, NNLM can't be applied to large data sets and it shows on rare words

Mikolov et al. (2013), instead, focused on the word vectors

Google's word2vec (Mikolov et al. 2013a)

Idea: achieve better performance allowing a simpler (shallower) model to be trained on much larger amounts of data

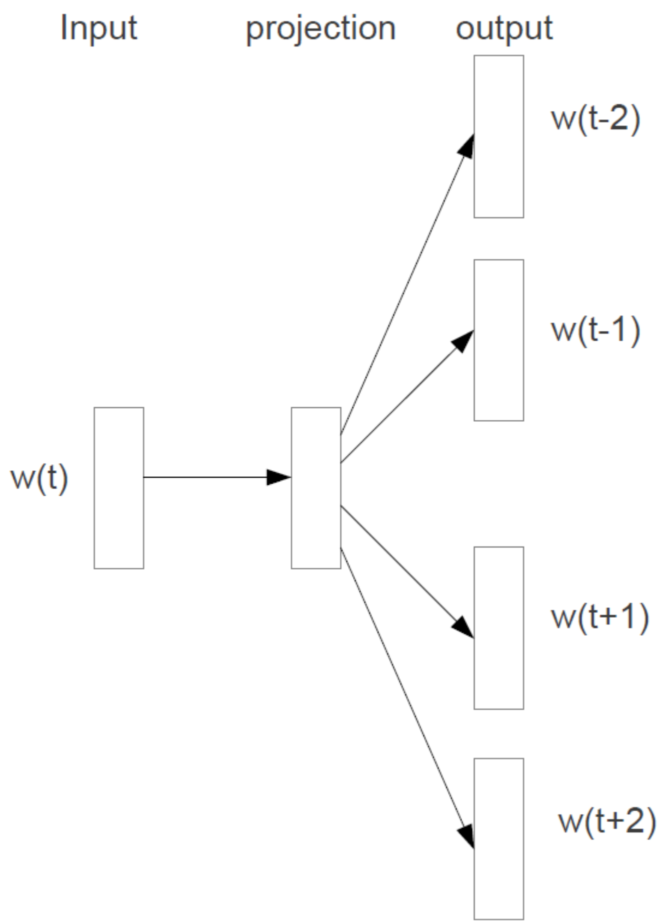
- No hidden layer (leads to 1000X speed up)
- Projection layer is shared (not just the weight m
- Context contain words both from history and futur

«You shall know a word
by the company it keeps»
John R. Firth, 1957:11.

```
…Pelé has called Neymar an excellent player…
…At the age of just 22 years, Neymar had scored 40 goals in 58 internationals…
…occasionally as an attacking midfielder, Neymar was called a true phenomenon…
```

These words will represent **Neymar**

Google word2vec Flavors

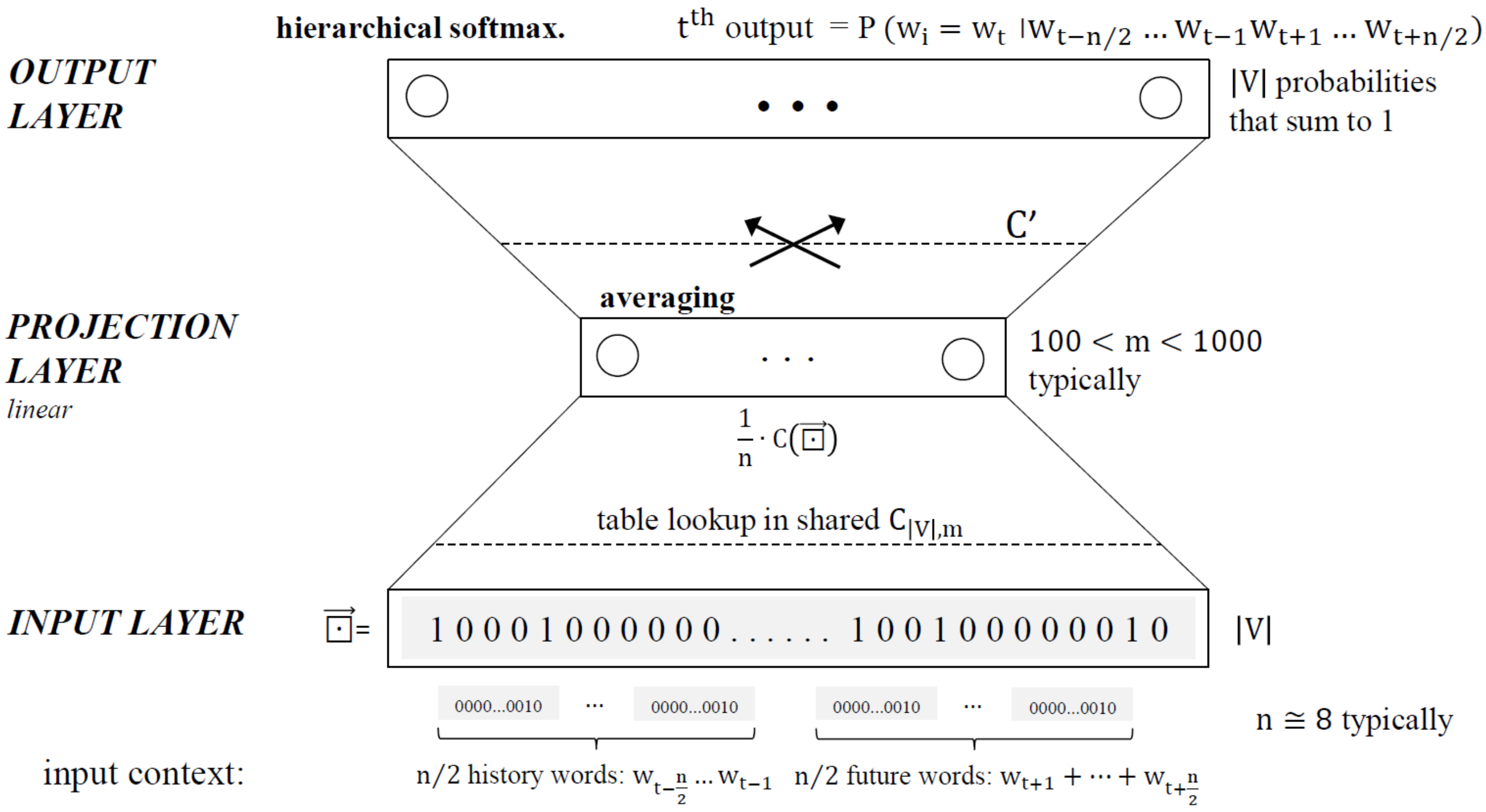


Skip-gram architecture

Word2vec's Continuous Bag-of-Words (CBOW)

For each training sequence: input = (context, target) pair: $(w_{t-\frac{n}{2}} \dots w_{t-1} w_{t+1} \dots w_{t+\frac{n}{2}}, w_t)$

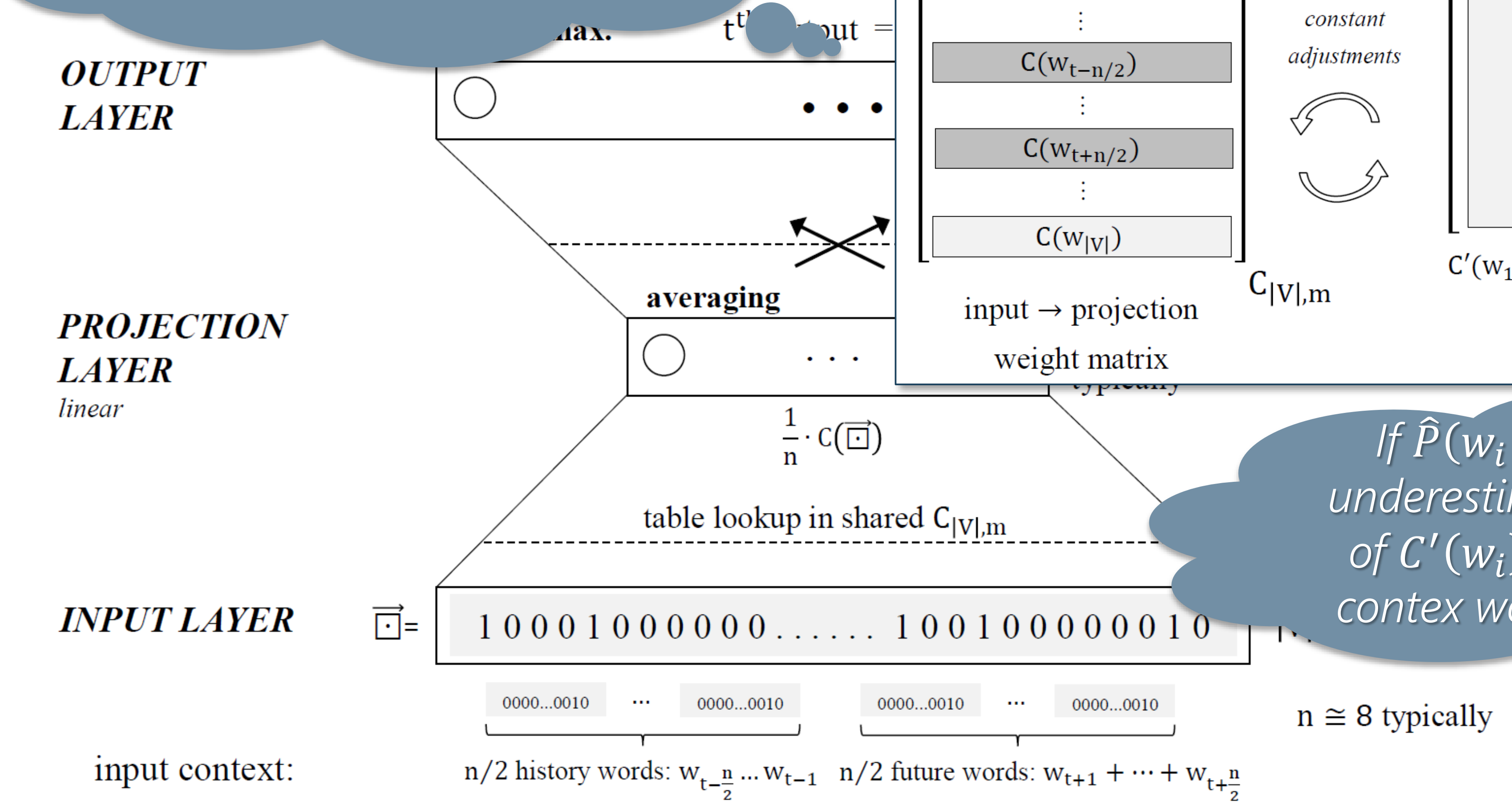
objective: minimize $E = -\log \hat{P}(w_t \mid w_{t-n/2} \dots w_{t-1} w_{t+1} \dots w_{t+n/2})$

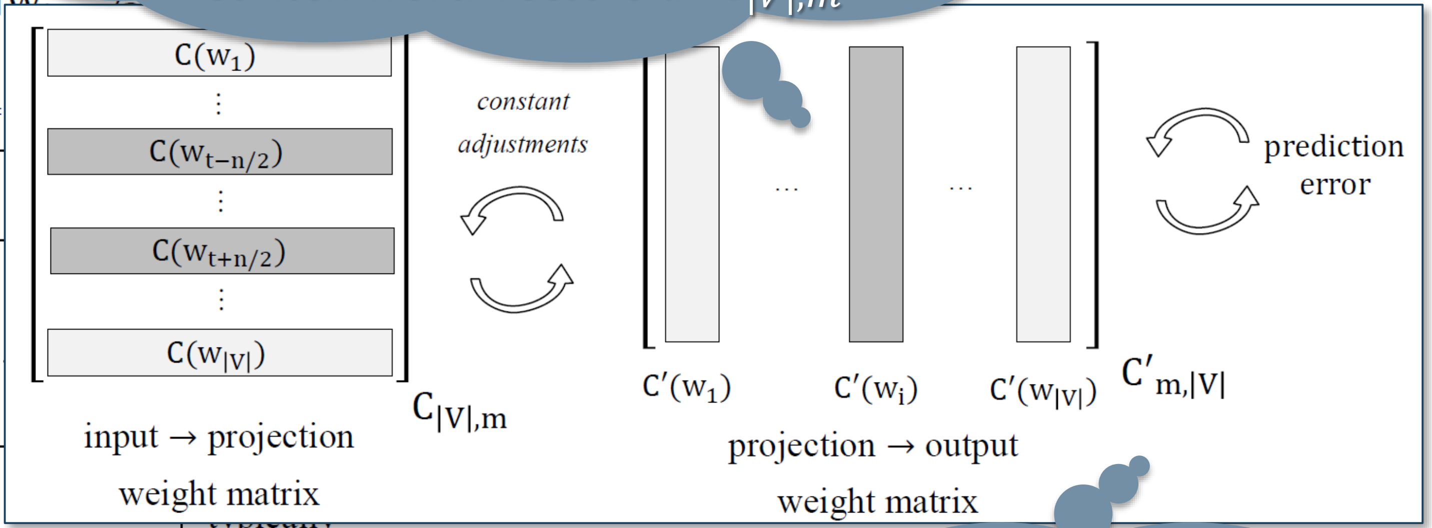


Word2vec's Continuous Bag-of-W

For each <context, target> pair only the context words are updated.

If $\hat{P}(w_i = w_t|context)$ is overestimated some portion of $C'(w_i)$ is subtracted from the contex word vectors in $C_{|V|,m}$





If $\hat{P}(w_i = w_t|context)$ is underestimated some portion of $C'(w_i)$ is added from the contex word vectors in $C_{|V|,m}$

Word2vec facts

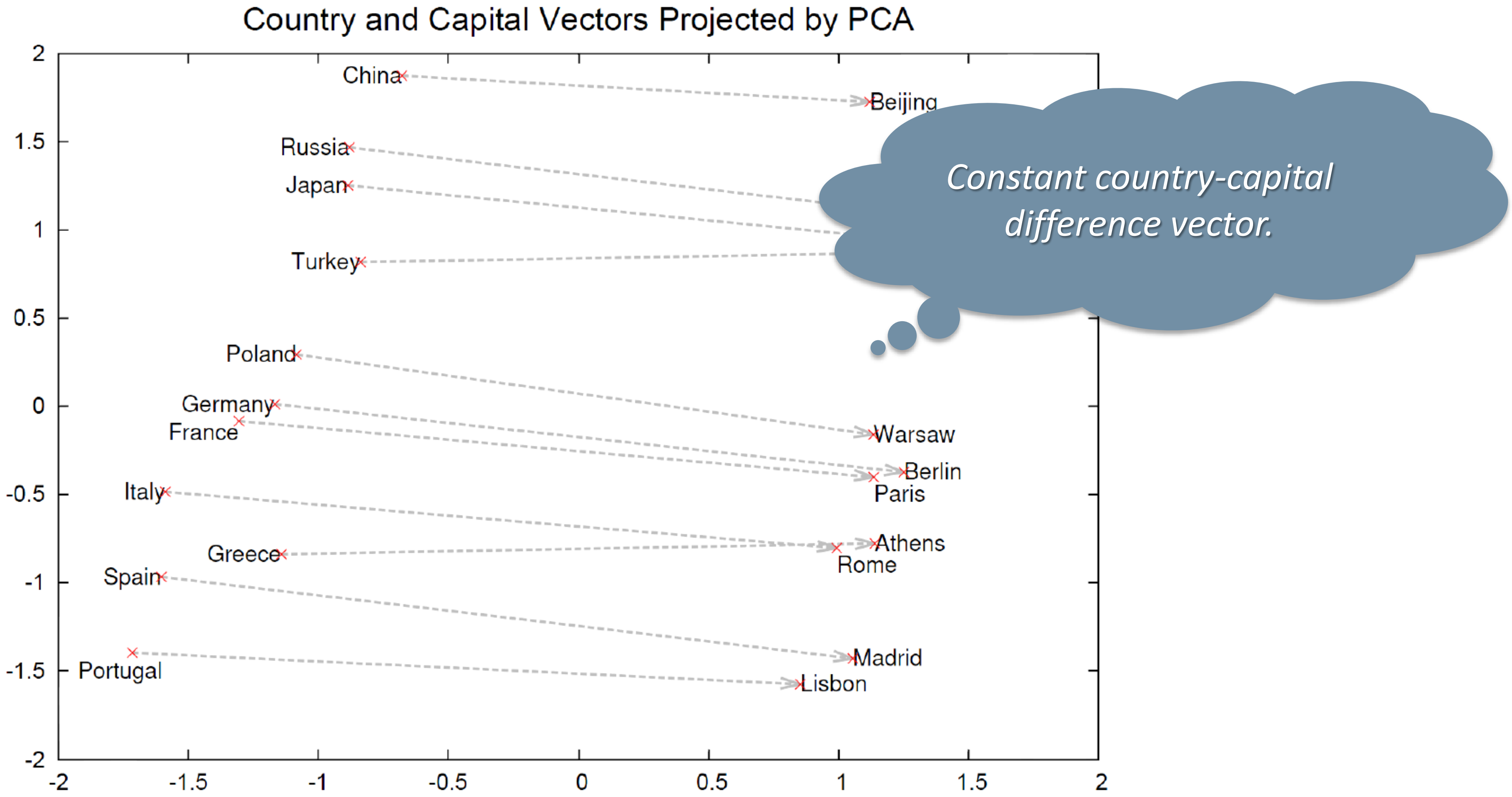
Word2vec shows significant improvements w.r.t. the NNML

- Complexity is $n \times m + m \times log|V|$ (Mikolov et al. 2013a)
- On Google news 6B words training corpus, with $|V| \sim 10^6$
  - CBOW with m=1000 took 2 days to train on 140 cores
  - Skip-gram with m=1000 took 2.5 days on 125 cores
  - NNLM (Bengio et al. 2003) took 14 days on 180 cores, for m=100 only!
- word2vec training speed ≅ 100K-5M words/s
- Best NNLM: 12.3% overall accuracy vs. Word2vec (with Skip-gram): 53.3%

Capital-Country	Past tense	Superlative	Male-Female	Opposite
Athens: **Greece**	walking: **walked**	easy: **easiest**	brother: **sister**	ethical: **unethical**

Adapted from Mikolov et al. (2013a)

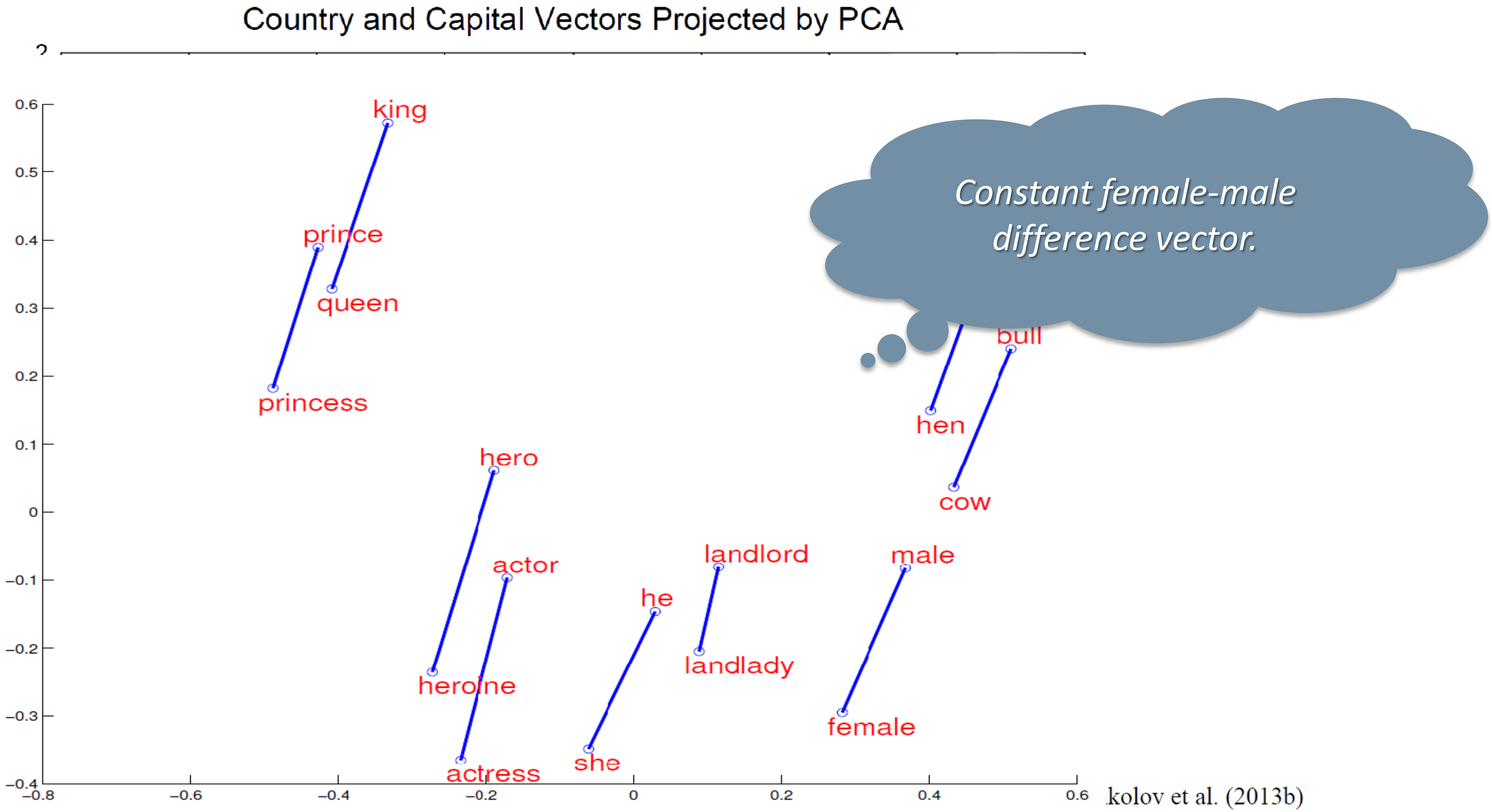
Regularities in word2vec Embedding Space



Mikolov et al. (2013b)

Picture taken from:
https://www.scribd.com/document/285890694/NIPS-DeepLearningWorkshop-NNforText

Regularities in word2vec Embedding Space

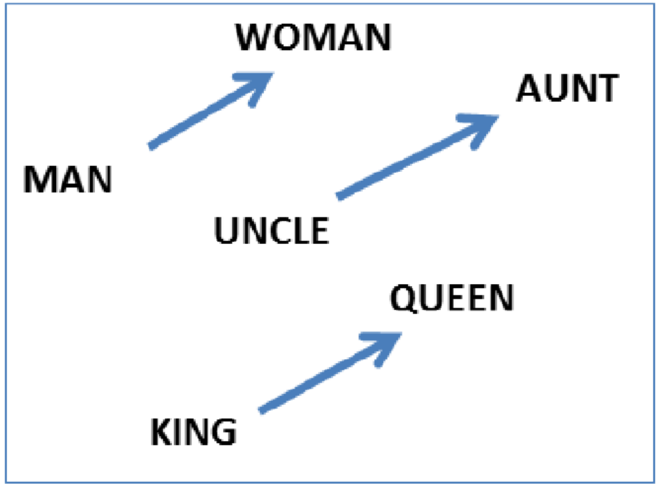


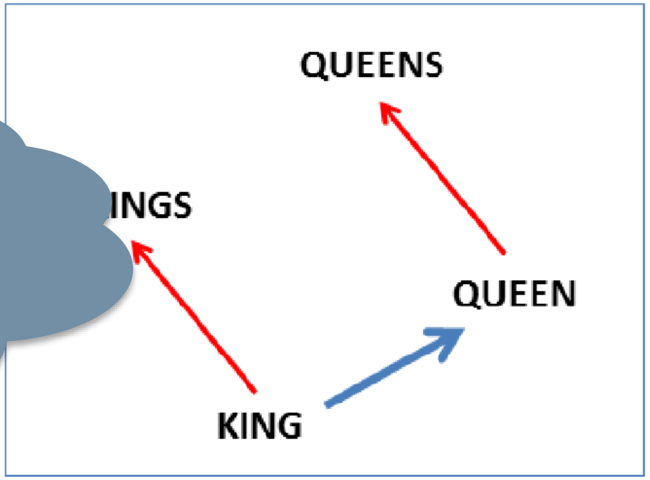
Picture taken from:
https://www.scribd.com/document/285890694/NIPS-DeepLearningWorkshop-NNforText

Regularities in word2vec Embedding Space

Vector operations are supported make «intuitive sense»:

- $w_{king} - w_{man} + w_{woman} \cong w_{queen}$
- $w_{paris} - w_{france} + w_{italy} \cong w_{rome}$
- $w_{windows} - w_{microsoft} + w_{google} \cong w_{android}$
- $w_{einstein} - w_{scientist} + w_{painter} \cong w_{picasso}$
- $w_{his} - w_{he} + w_{she} \cong w_{her}$
- $w_{cu} - w_{copper} + w_{gold} \cong w_{au}$
- …





«You shall know a word by the company it keeps»
John R. Firth, 1957:11.

Picture taken from:
https://www.scribd.com/document/285890694/NIPS-DeepLearningWorkshop-NNforTex

Applications of word2vec in Information Retrieval

Query: “restaurants in mountain view that are not very good”

Phrases: “restaurants in (mountain view) that are (not very good)”

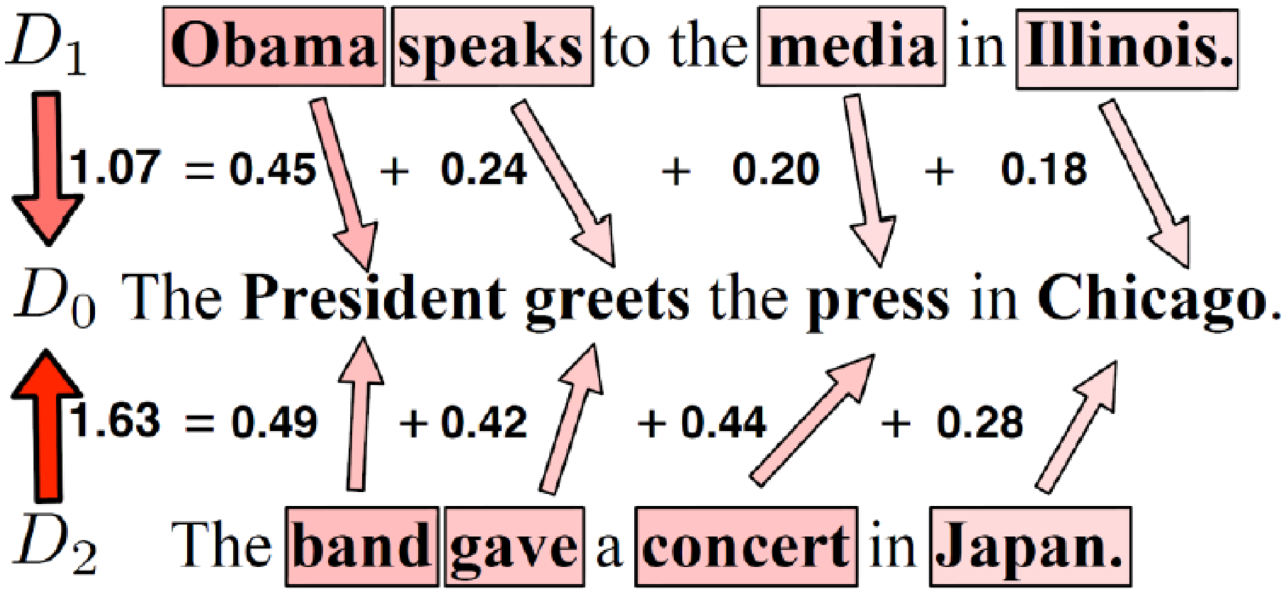
Vectors: “restaurants+in+(mountain view)+that+are+(not very good)”

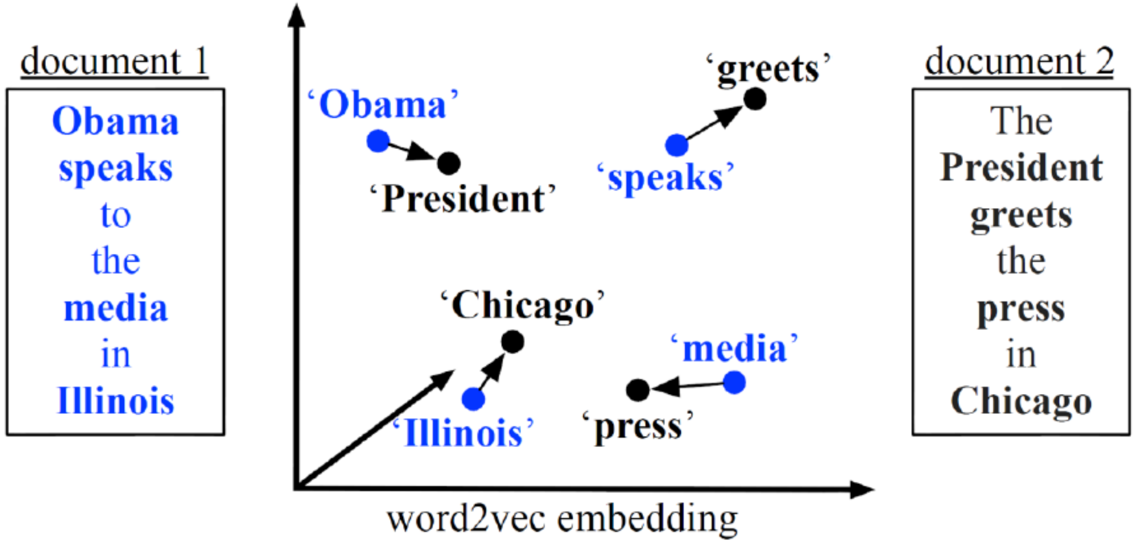
Expression	*Nearest tokens*
Czech + currency	koruna, Czech crown, Polish zloty, CTK
Vietnam + capital	Hanoi, Ho Chi Minh City, Viet Nam, Vietnamese
German + airlines	airline Lufthansa, carrier Lufthansa, flag carrier Lufthansa
Russian + river	Moscow, Volga River, upriver, Russia
French + actress	Juliette Binoche, Vanessa Paradis, Charlotte Gainsbourg

(Simple and efficient, but will not work for long sentences or documents)

Applications of word2vec in Document Classification/Similarity

With BoW D_1 and D_2 are equally similar to D_0.





Word embeddings allow to capture the «semantics» of the document ...

Applications of word2vec in Sentiment Analysis

«You shall know a word by
the company it keeps»
John R. Firth, 1957:11.

No need for classifiers, just use cosine distanc

```
Enter word or sentence (EXIT to break): sad

Word: sad  Position in vocabulary: 4067

                                             Word       Cosine distance
------------------------------------------------------------------------
                                        saddening              0.727309
                                              Sad              0.661083
                                         saddened              0.660439
                                     heartbreaking              0.657351
                                      disheartening              0.650732
                                     Meny_Friedman              0.648706
                          parishioner_Pat_Patello              0.647586
                                       saddens_me              0.640712
                                       distressing              0.639909
                                 reminders_bobbing              0.635772
                                  Turkoman_Shiites              0.635577
                                          saddest              0.634551
                                      unfortunate              0.627209
                                            sorry              0.619405
                                      bittersweet              0.617521
                                           tragic              0.611279
                                        regretful              0.603472
```

GloVe: Global Vectors for Word Representation (Pennington et al. 2014)

GloVe makes explicit what word2vec does implicitly

- Encodes meaning as vector offsets in an embedding space
- Meaning is encoded by ratios of co-occurrece probabilities

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	[illegible]
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	[illegible]	[illegible]

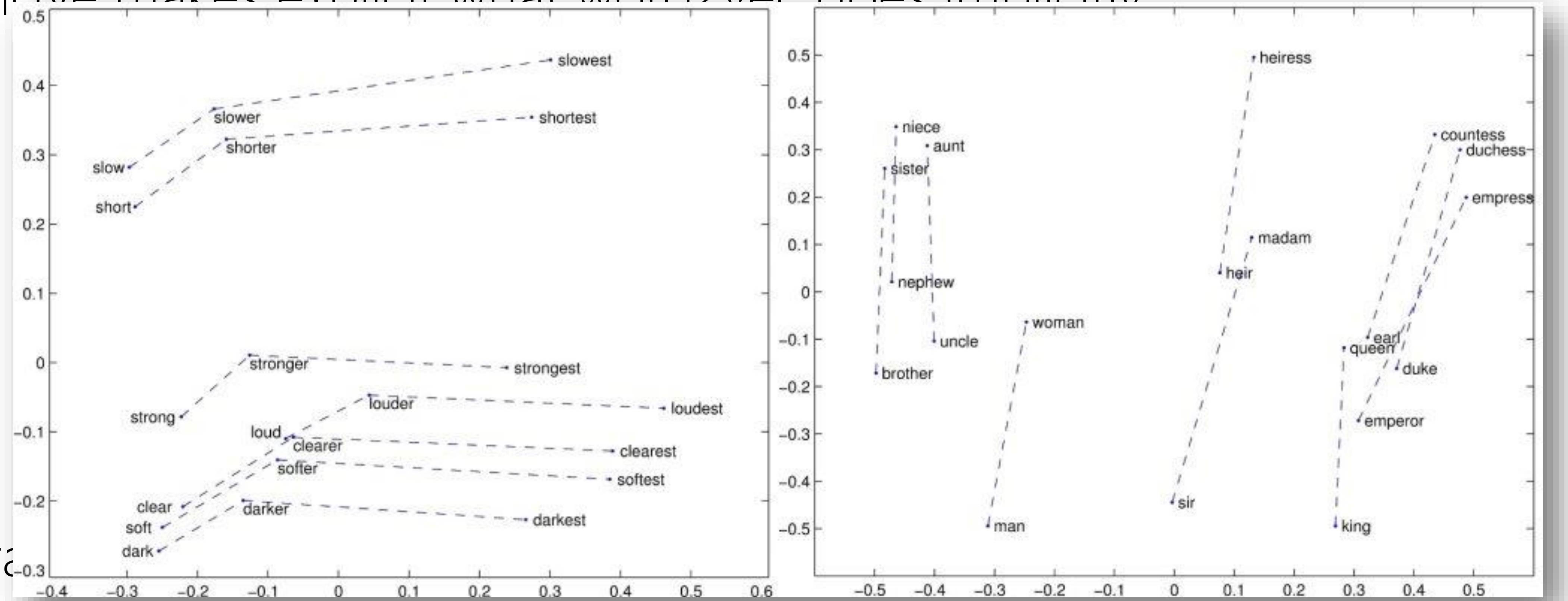
Refer to Pennington et al. paper for details on this loss function …

Trained by weighted least squares

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

GloVe: Global Vectors for Word Representation (Pennington et al. 2014)

GloVe makes explicit what word2vec does implicitly



Tra

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Nearest Neighbours with GloVe

What are the closest words to the target word *frog*:

1. *Frog*
2. Frogs
3. Toad
4. Litoria
5. Leptodactylidae
6. Rana
7. Lizard
8. Eleutherodactylus


3. litoria


4. leptodactylidae


5. rana


7. eleutherodactylus