Representing Word as Vectors

Guanlin Li

Dec. 2 2016

Outline

PART I

- Traditional Language Model Review
  - Modelling Language? Generative Model
  - Perplexity with Intuition
  - Sparsity with Intuition *(I hope)*
- Intro to Neural Probabilistic Language Model
  - Symbol-in Symbol-out
  - Self Supervision
  - By Product

Outline

PART II

- Word Embedding – Basic Models
  - Word2vec
  - GloVe
- Representation Learning of Lexical Meaning
  - Sentiment Embedding
  - Topical Embedding
  - Discourse Relation-aware Embedding

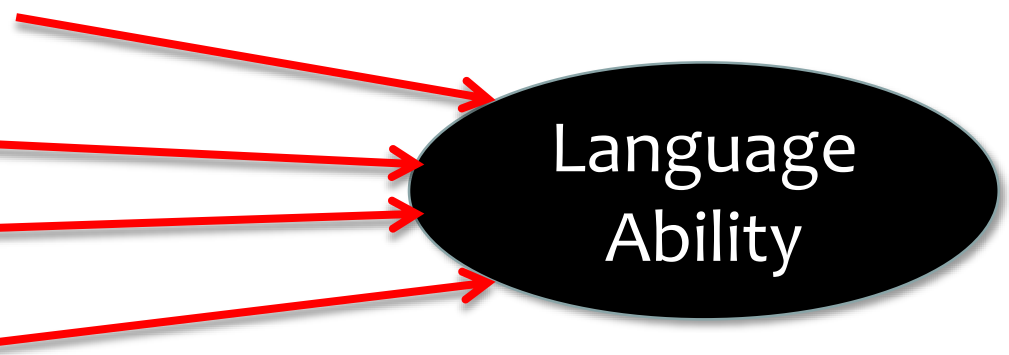
Outline

PART I

- Traditional Language Model Review
  - Modelling Language? Generative Model
  - Perplexity with Intuition
  - Sparsity with Intuition *(I hope)*
- Intro to Neural Probabilistic Language Model
  - Symbol-in Symbol-out
  - Self Supervision
  - By Product

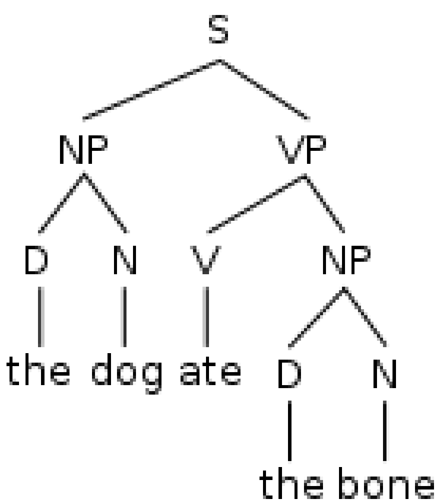
Human Language & Linguistics: in a few words

- Human Language is the ability to acquire and use complex systems of communication.
- Modern Linguistics: analytic study of language
  - Morphology
  - Syntax
  - Semantics
  - Pragmatics



Syntax & Chomsky

- Generative Grammar



- Kind of modelling the language production process of human being
- In a very formal, abstract way

Noam Chomsky

Cognitive Science

Transformation Grammar

Non-Transformation Grammar

Parameter Theory

X-Bar Theory

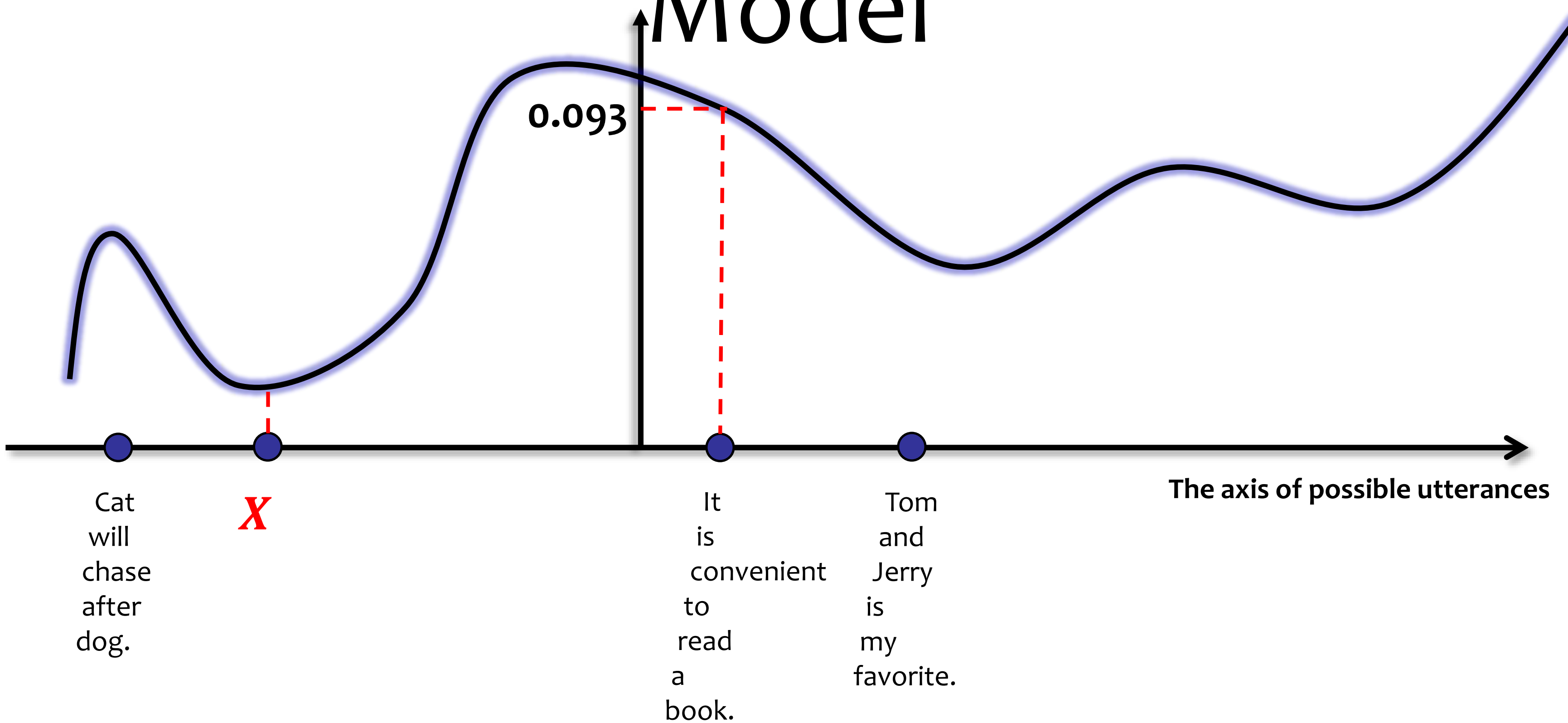
Lexical Functional Grammar

GB Theory

Minimalist Program

Categorial Grammar

Traditional (Statistical) Language Model



- A Probability distribution over a whole sequence
  - $X \sim P(\cdot)$

Traditional Language Model

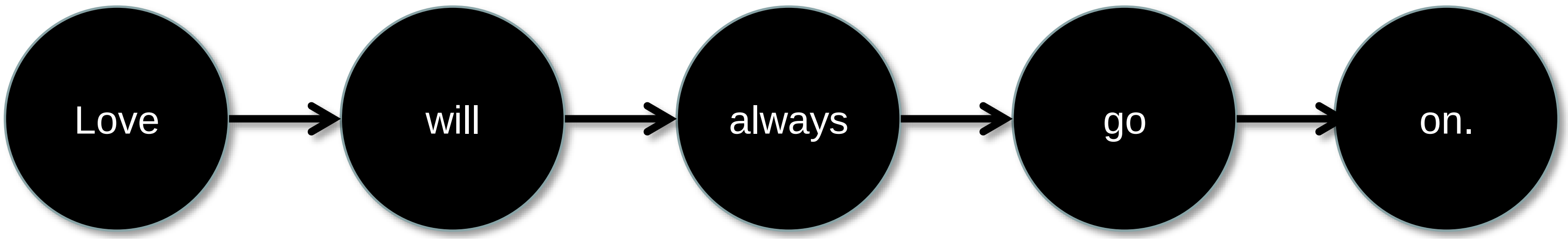
- The property of natural language utterance
  - Segment after segment.
  - A linear structure: from left to right, word by word
- Chain rule can be used without extra assumption
  - $P(w_1, w_2, w_3) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)$

Parameter of the model

$$P(dog|walk) = \frac{count(walk, dog)}{count(walk)}$$

$$P(w_n|w_1, \dots, w_{n-1})$$

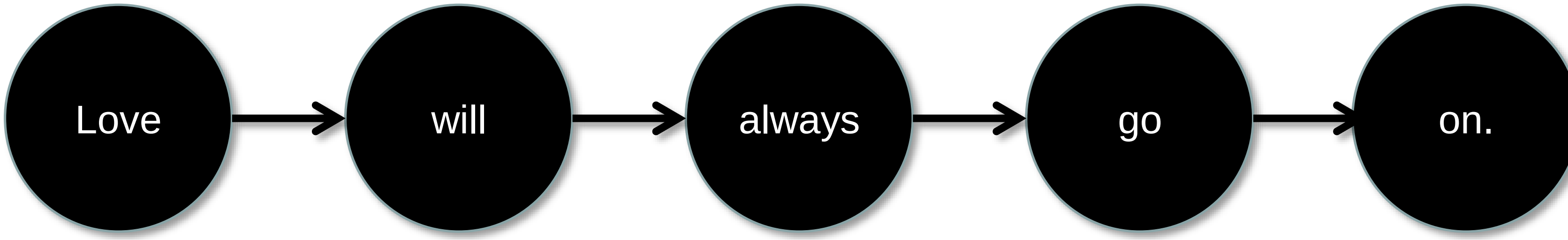
Traditional Language Model



- Markov Chain is a natural way of modelling distribution over sequence
  - $P(w_1, \ldots, w_n) = P(w_1|start)P(w_2|w_1) \ldots P(w_n|w_{n-1})P(stop|w_n)$

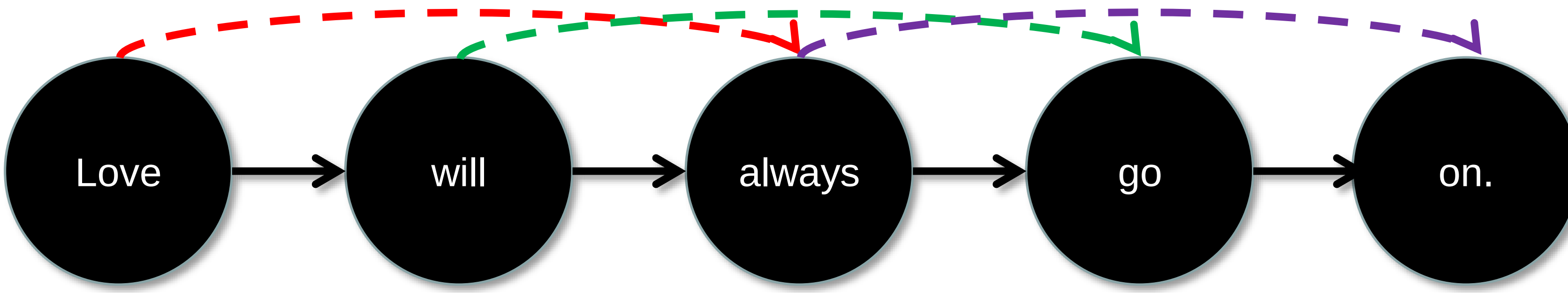
First-order Markov Assumption:
Given the previous state, the probability of the current state is independent of the history, $P(w_n|w_1, \ldots, w_{n-1}) = P(w_n|w_{n-1})$.

Traditional Language Model



- If we work with 1st order Markov assumption to model language, we get a bigram model
  - $P(w_i|w_{i-1})$ are the parameters
  - $w_i \in \{stop\} \cup V, w_{i-1} \in \{start\} \cup V$
  - So if we have a vocabulary size of 10000, we are going to have 10001^2 parameters, $|V|^2$

Traditional Language Model



- If we work with 2st order Markov assumption to model language, we get a 3-gram model
  - $P(w_i | w_{i-1}, w_{i-2})$ are the parameters
  - $w_i \in \{stop\} \cup V$, $w_{i-1} \in \{start\} \cup V$, $w_{i-2} \in \{start\} \cup V$
  - Approximately 10000^3 parameters
  - Longer context ☺

Traditional Language Model

n-gram language model:

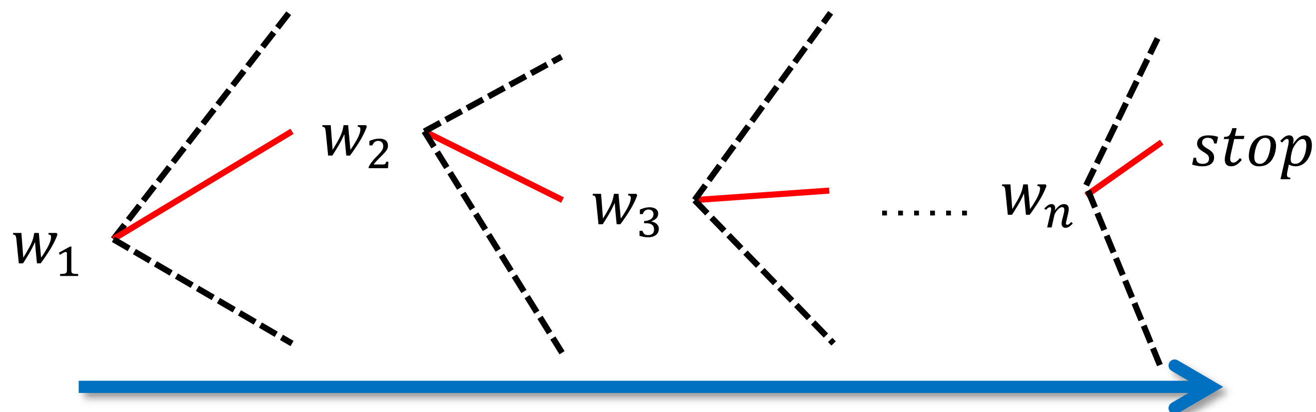
An n-gram language model consists of a finite set V, and a parameter $q(w_t|w_{t-1}, \dots w_{t-n+1})$ for each n-gram $w_{t-n+1}, \dots, w_t$ such that $w_{t-i} \in V \cup \{stop\}$, and $u, v \in V \cup \{*\}$.

An example - trigram

- $q(w|u, v)$ can be seen as the prob. of seeing w after seeing bigram u, v
- For any sentence, $x_1 \dots x_n$ where $x_i \in V$ for $i = 1 \dots (n-1)$, and $x_n = stop$, the prob. of the sentence under the trigram model is
  - $P(x_1, \dots, x_n) = \prod_{i=1}^{n} q(x_1|x_0, x_{-1})$, where $x_{0,-1} =*$

Generative Power

- Once we use maximum likelihood estimation to estimate $q(w|u,v)$ with count ratios over a corpus, we can get a generative model
- That is: start from arbitrary word w_1, run simulation



$q(w_2|*,w_1)$ $q(w_3|w_1,w_2)$ $q(stop|w_{n-1},w_n)$

A conditional multinoulli over V

Don’t be perplexed with Perplexity

- Evaluation of a learned language model
  - Suppose we have two language model $\mathcal{M}_1, \mathcal{M}_2$, estimated on $\mathcal{L}_{train}$
  - Given $\mathcal{L}_{test}$, we can compute the probability P_i , $i = \{1,2\}$, of generating $\mathcal{L}_{test}$ under $\mathcal{M}_1, \mathcal{M}_2$
  - According to Maximum Likelihood Principle, the better one, the bigger P_i
- $P_i = \prod_{k=1}^{N} P_{\mathcal{M}_i}(x^k) = \prod_{k=1}^{N} \prod_{j=1}^{l^k} q(x_j | x_{j-1}, x_{j-2})$
  - where $x^k \in \{x^1, \dots, x^N\} = \mathcal{L}^{test}$
  - Too small

Don’t be perplexed with Perplexity

- Perplexity over a word
  - Defined as 2^{-l}, l is average log-likelihood of a word
  - $l = \frac{1}{M}\sum_k^N logP_{\mathcal{M}_i}(x^k)$, word number $M = \Sigma l^k$
- Information Theory
  - $logP(X = x)$ is the surprisal of observing an event x
  - The code length we encode this event using 0-1 code , according to frequency

Don't be perplexed with Perplexity

- Since $logP_{\mathcal{M}_i}(x^k) = \Sigma \log q(w_i|w_{i-2}, w_{i-1})$
  - $-logq(w|\cdot)$ means surprisal of observing a w, given context, it is the bits need to encode the random event
  - Sum them all and average! We got average bit length to encode any word.
- Since binary coding, we can compute the total word number it could represent
  - 2^{-l}, this is like the actual vocabulary size after compression

The Problem of Sparsity

- If everyday usage, vocabulary size of 5000, we have 5000^3 parameters, it is 1.25×10^{11}
- Sentence average length 20, 20 triples per sentence
- We need at least 6.25×10^9 sentence with no repeatable triples, can we have every parameter $q(.|..)$ not being zero

$$\underset{0}{\underset{\Downarrow}{P(x^i)}} = ..\, \underset{0}{\underset{\Downarrow}{q\left(x_t^i \middle| x_{t-2}^i, x_{t-1}^i\right)}}..$$

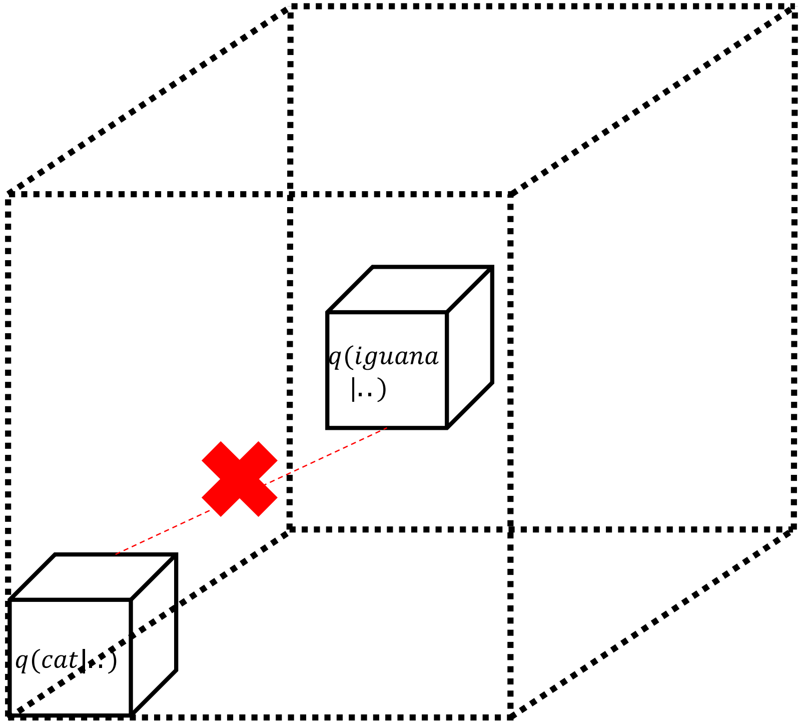
The Problem of Sparsity

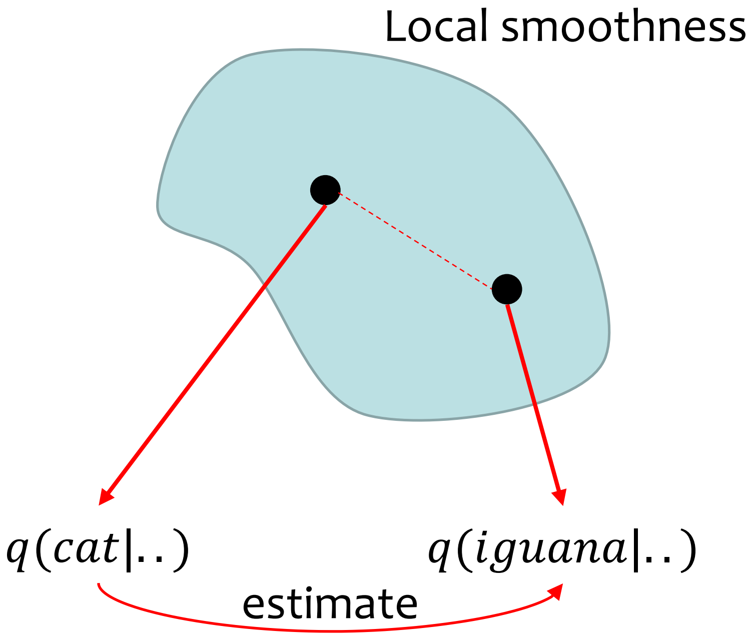
- Discrete makes interpolation difficult

美洲鬣蜥蜴

  - *The dog is chasing the cat.*
    - $q(cat|chasing, the)$
  - *The dog is chasing the iguana.*
    - $q(iguana|chasing, the)$
- If the corpus has *iguana* and *cat* cooccur with *animal*. We can use the knowledge to smooth the prediction.

The Problem of Sparsity





The word *cat iguana* and *animal* need to **share** their information of cooccurrence

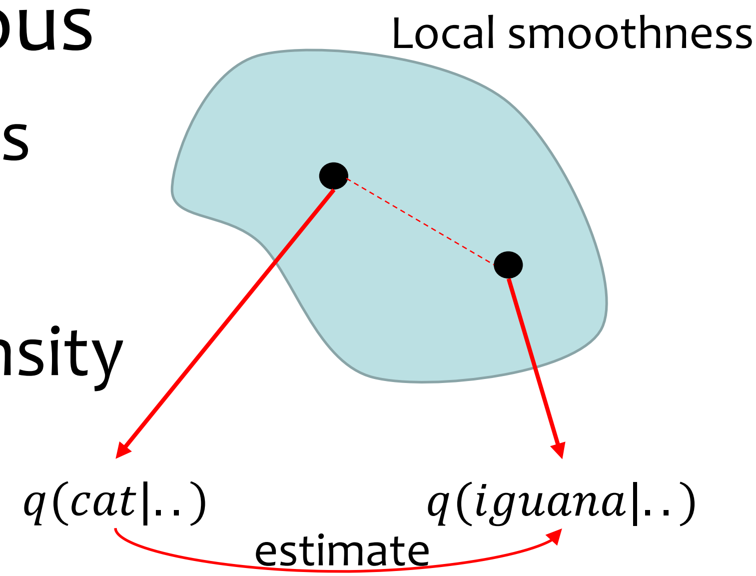
Outline

PART I

- Traditional Language Model Review
  - Modelling Language? Generative Model
  - Perplexity with Intuition
  - Sparsity with Intuition *(I hope)*
- Intro to Neural Probabilistic Language Model
  - Symbol-in Symbol-out
  - Self Supervision
  - By Product

Neural Probabilistic Language Model

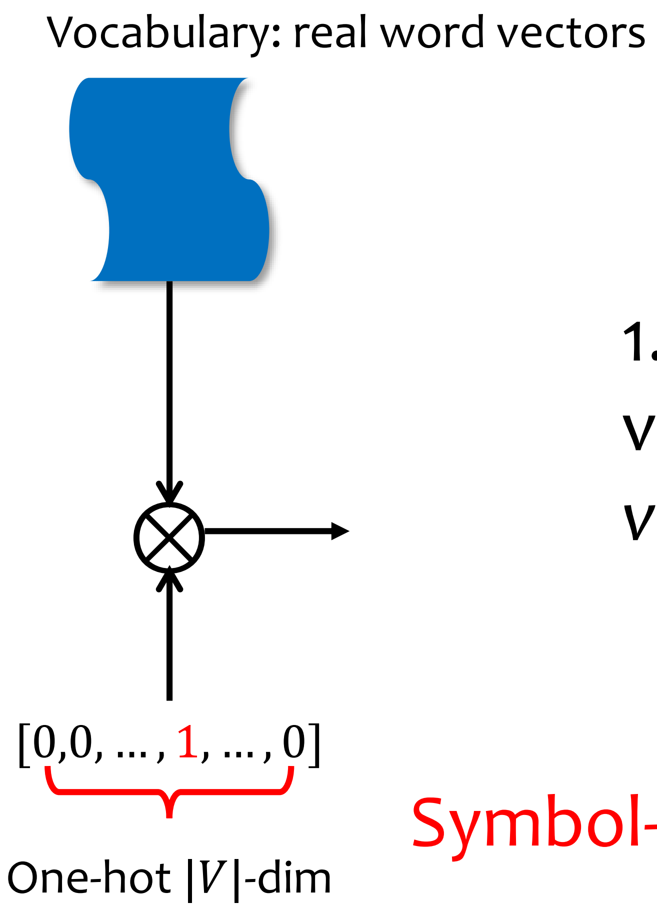
- Statistical Language Model
  - $P(w_i|context)$
  - Count, count, count; too young, too naïve!
- Change model of $P(w_i|context)$
- Make discrete → continuous
  - Continuousness guarantees neighborhood smooth
  - Model P as continuous density
  - Conditional distribution
    - Ah Hah! Neural NetWORKS!



Neural Probabilistic Language Model

1. Associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in $\mathbb{R}^m$)
2. Express the joint probability function of word sequences in terms of the feature vector of these words in the sequence
3. Learn simultaneously the word feature vectors and the parameters of that probability function

Neural Probabilistic Language Model

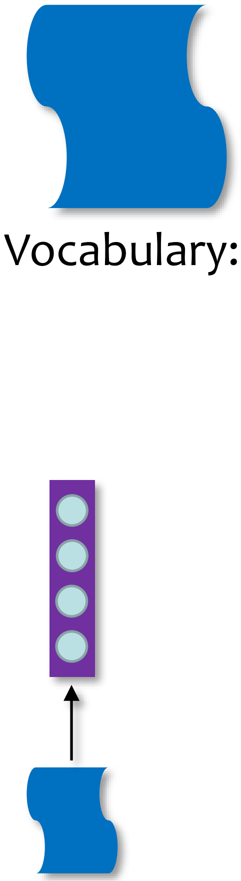


1. Associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in $\mathbb{R}^m$)

Symbol-in

The good life is one inspired by love and guided by knowledge .

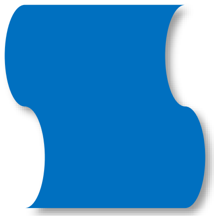
Neural Probabilistic Language Model



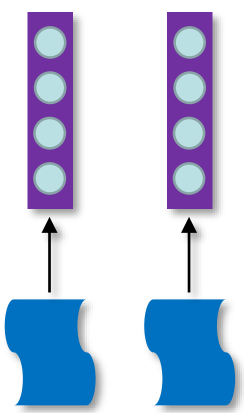
Vocabulary: real word vectors

The good life is one inspired by love and guided by knowledge .

Neural Probabilistic Language Model



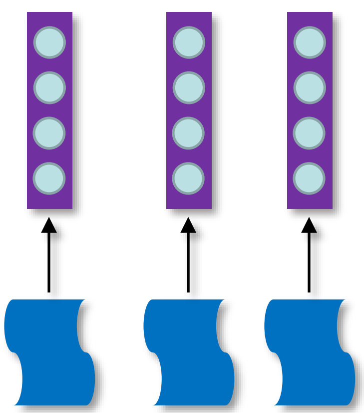
Vocabulary: real word vectors



The good life is one inspired by love and guided by knowledge .

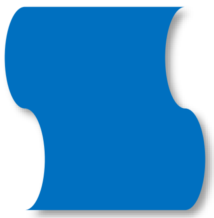
Neural Probabilistic Language Model

Vocabulary: real word vectors

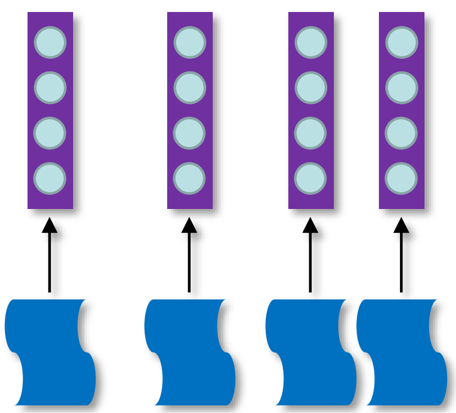


The good life is one inspired by love and guided by knowledge .

Neural Probabilistic Language Model

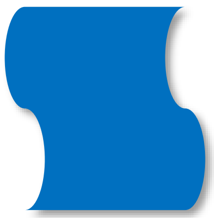


Vocabulary: real word vectors

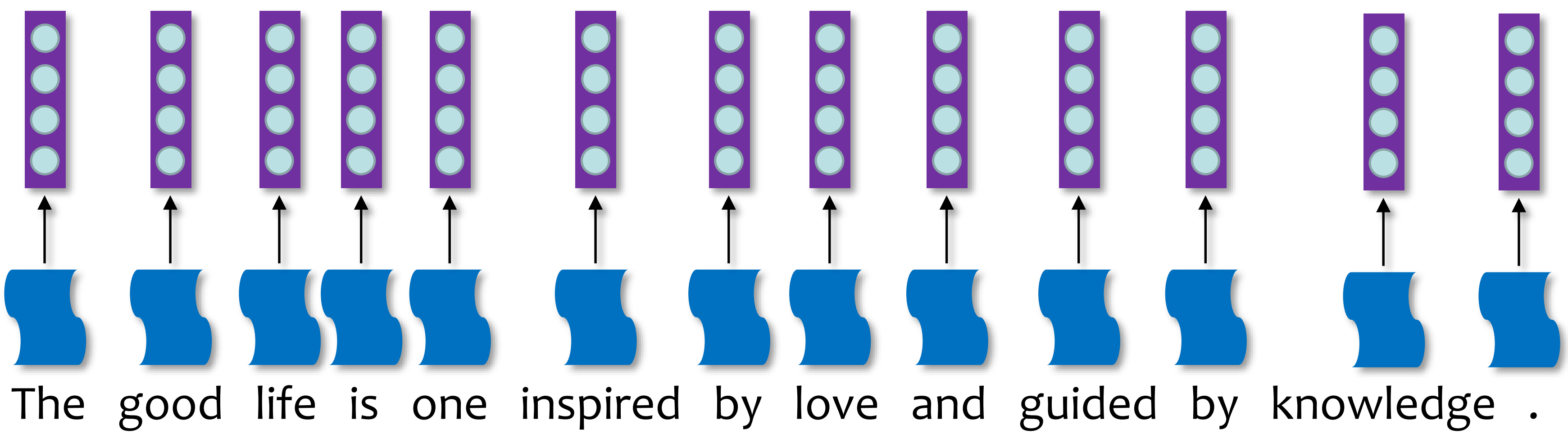


The good life is one inspired by love and guided by knowledge .

Neural Probabilistic Language Model

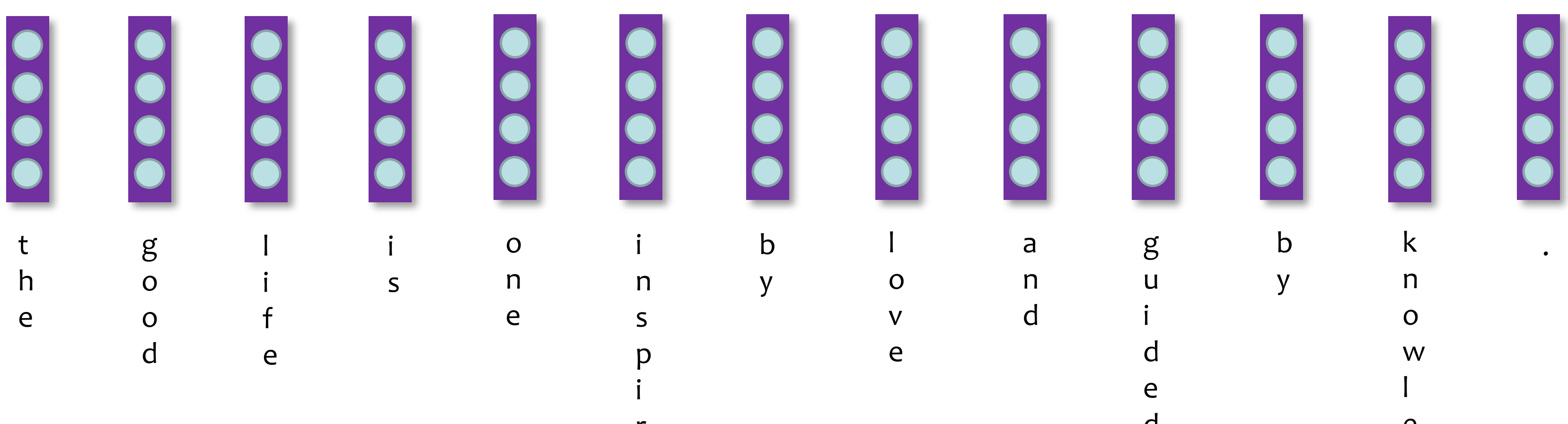


Vocabulary: real word vectors

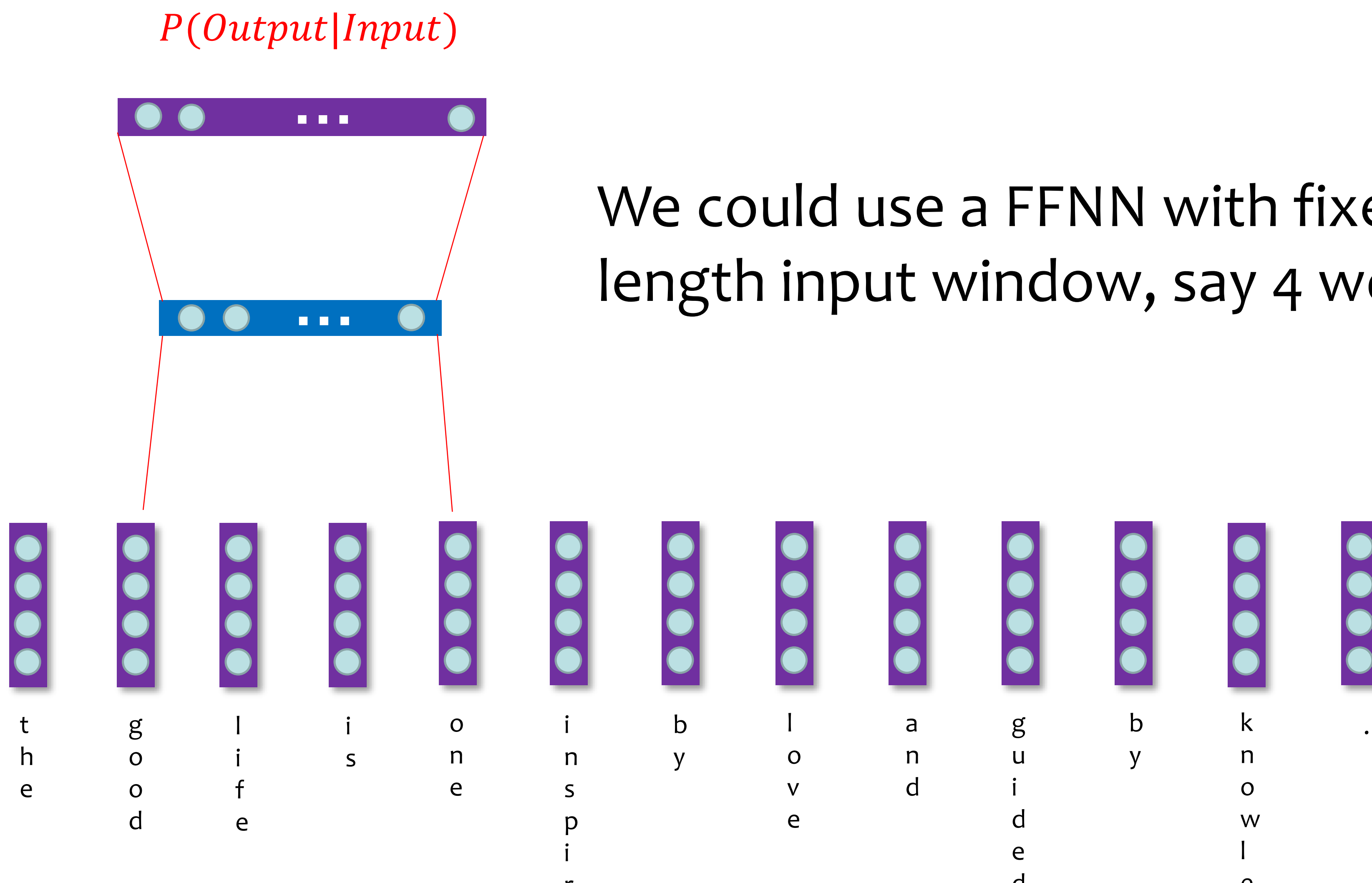


Neural Probabilistic Language Model

2. Express the joint probability function of word sequences in terms of the feature vector of these words in the sequence

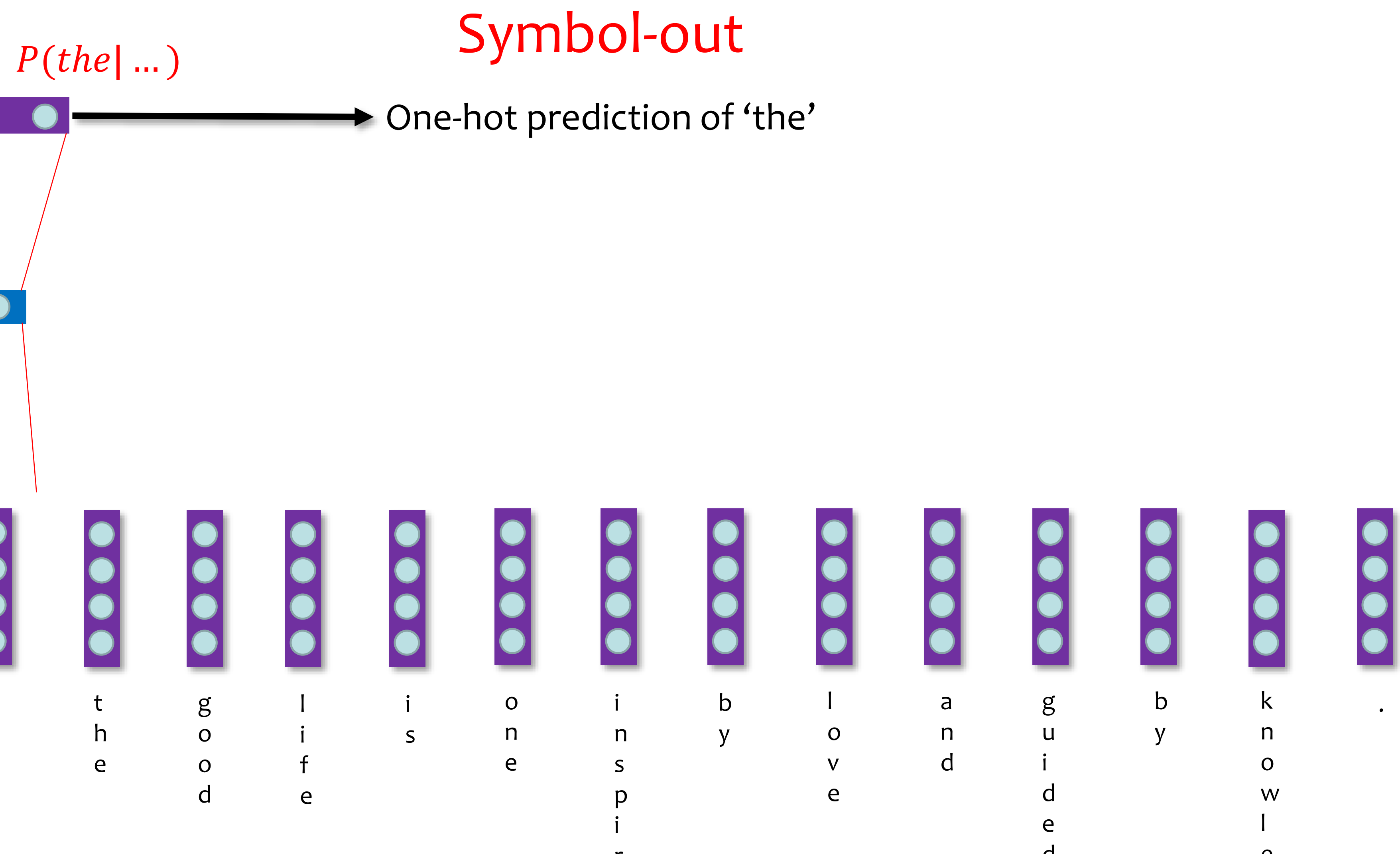


Neural Probabilistic Language Model

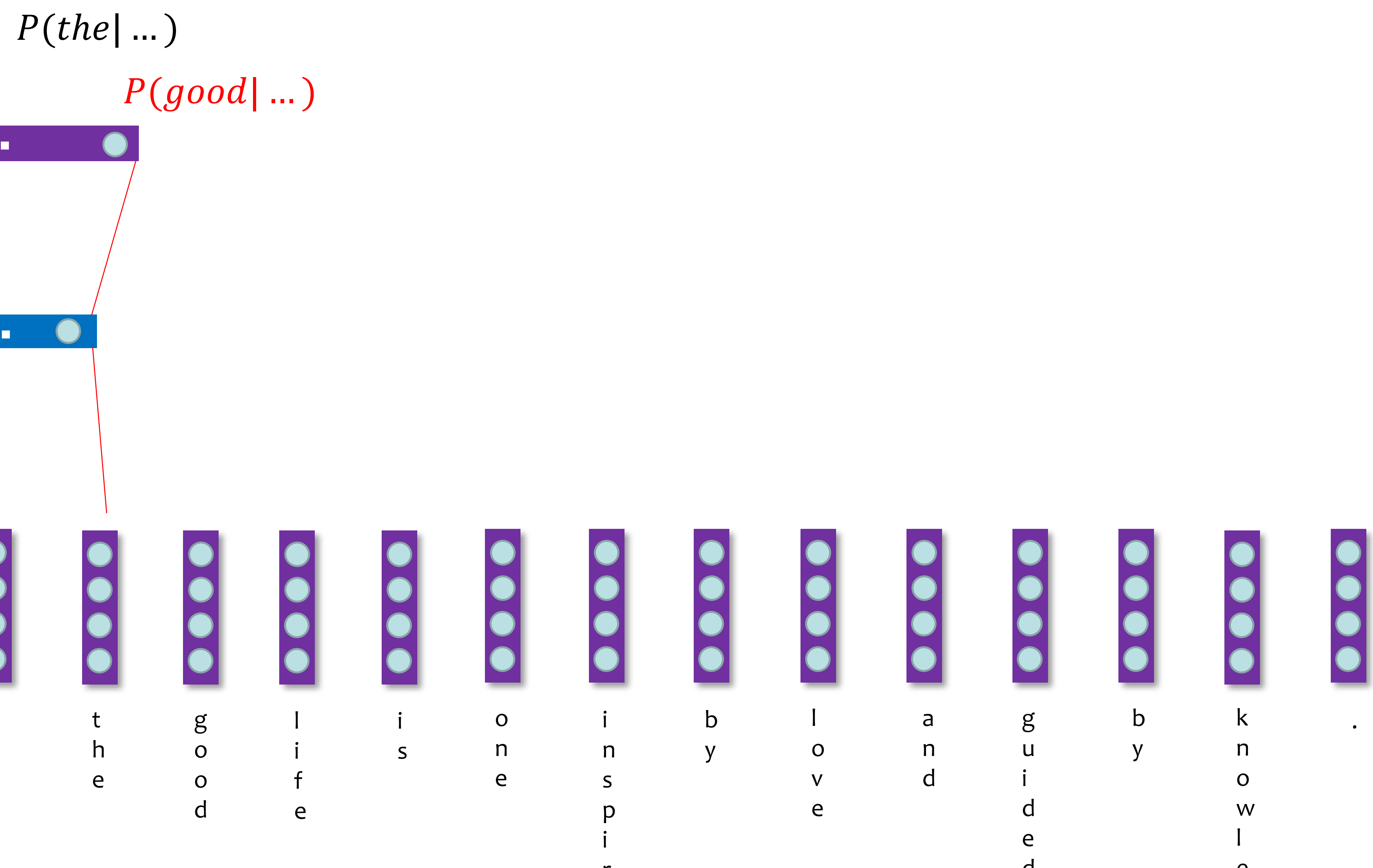


We could use a FFNN with fixed length input window, say 4 words.

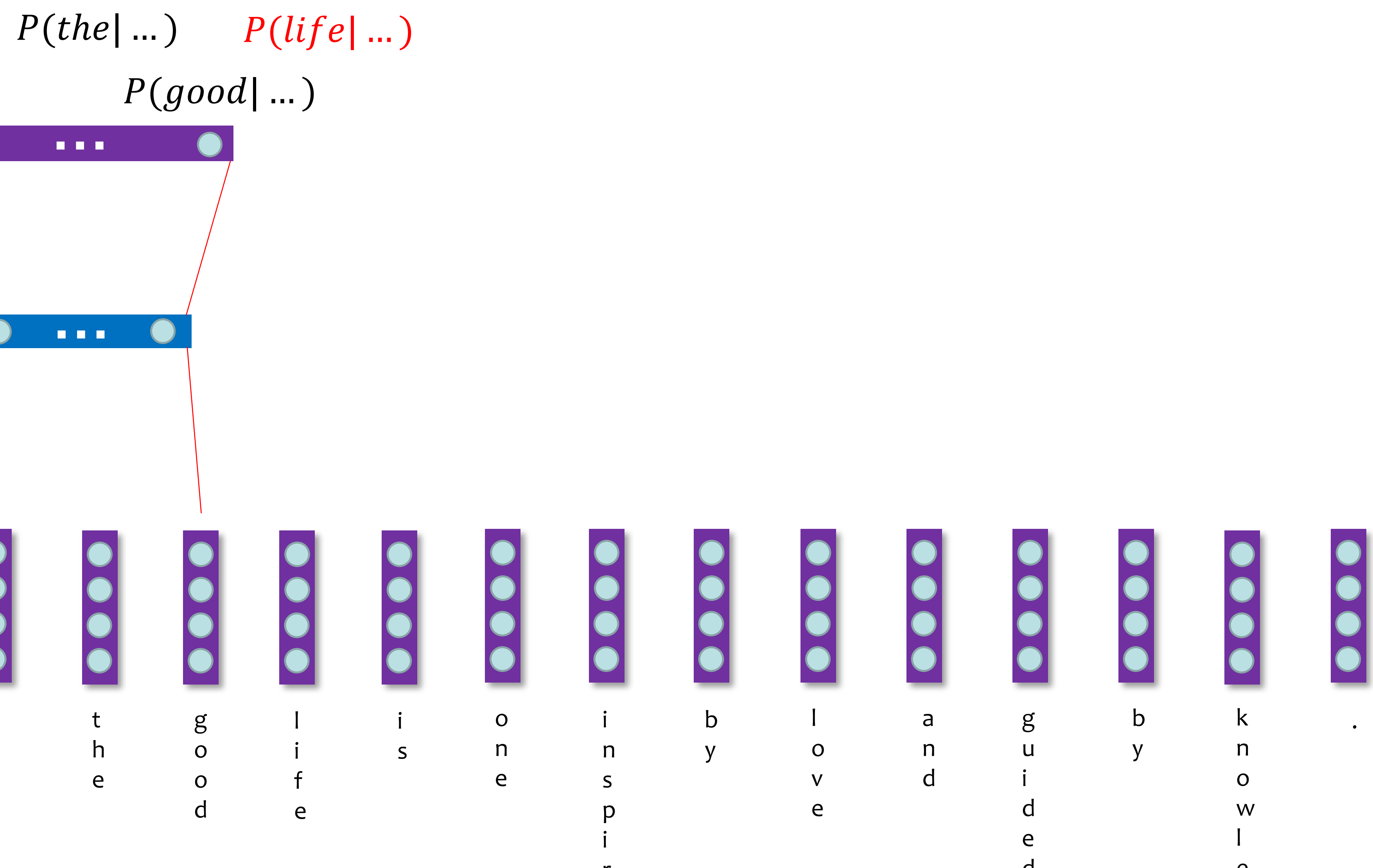
Neural Probabilistic Language Model



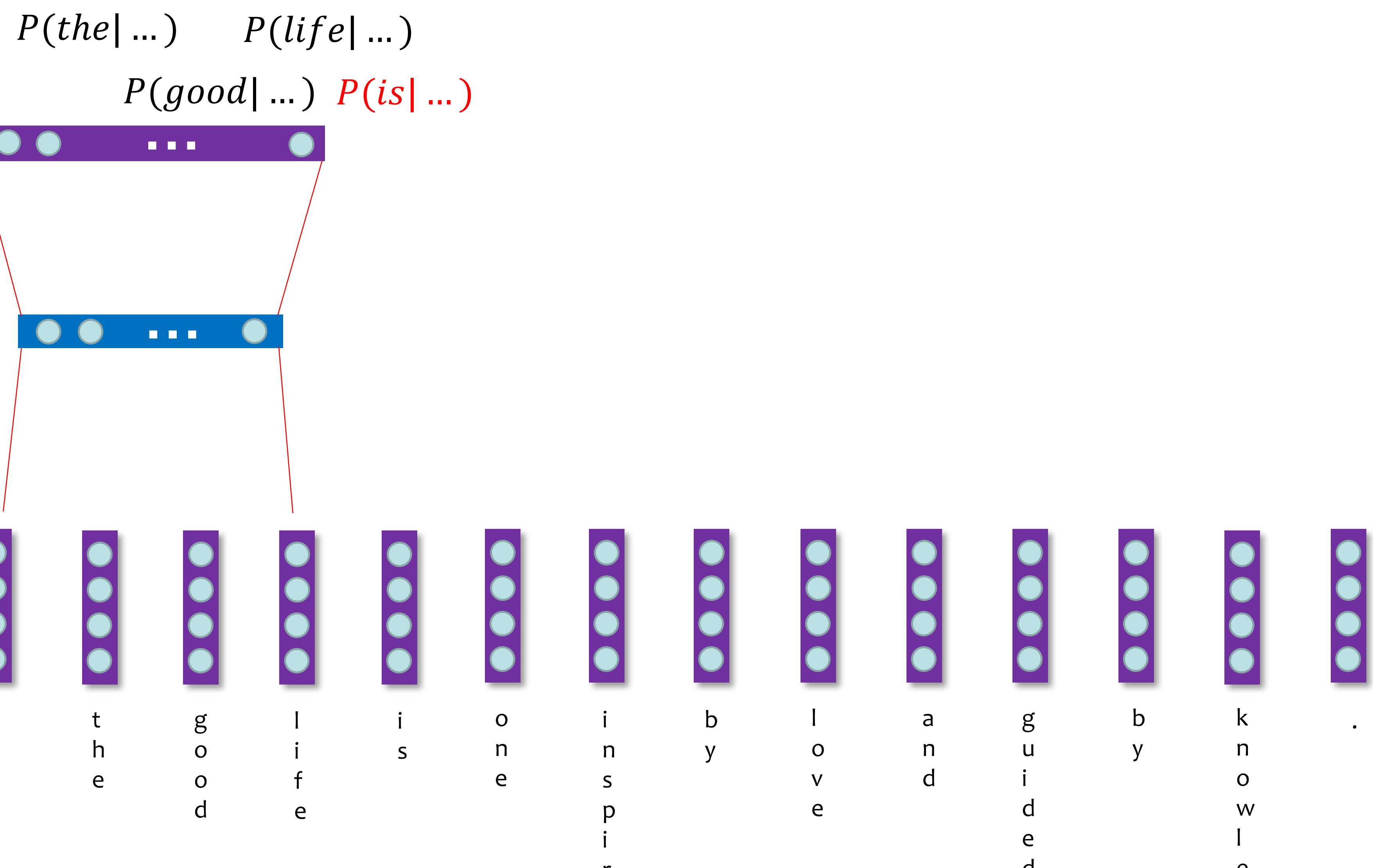
Neural Probabilistic Language Model



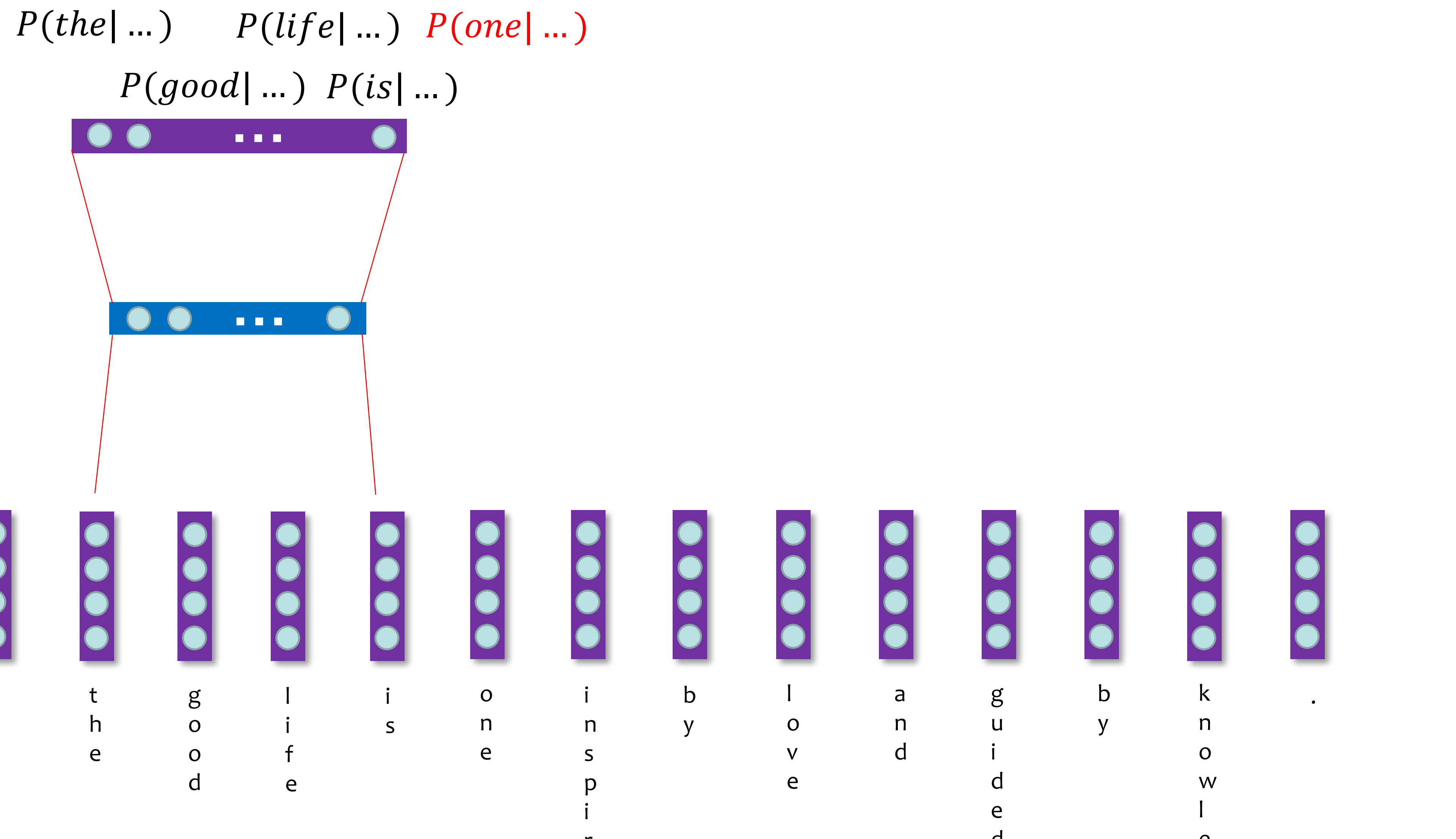
Neural Probabilistic Language Model



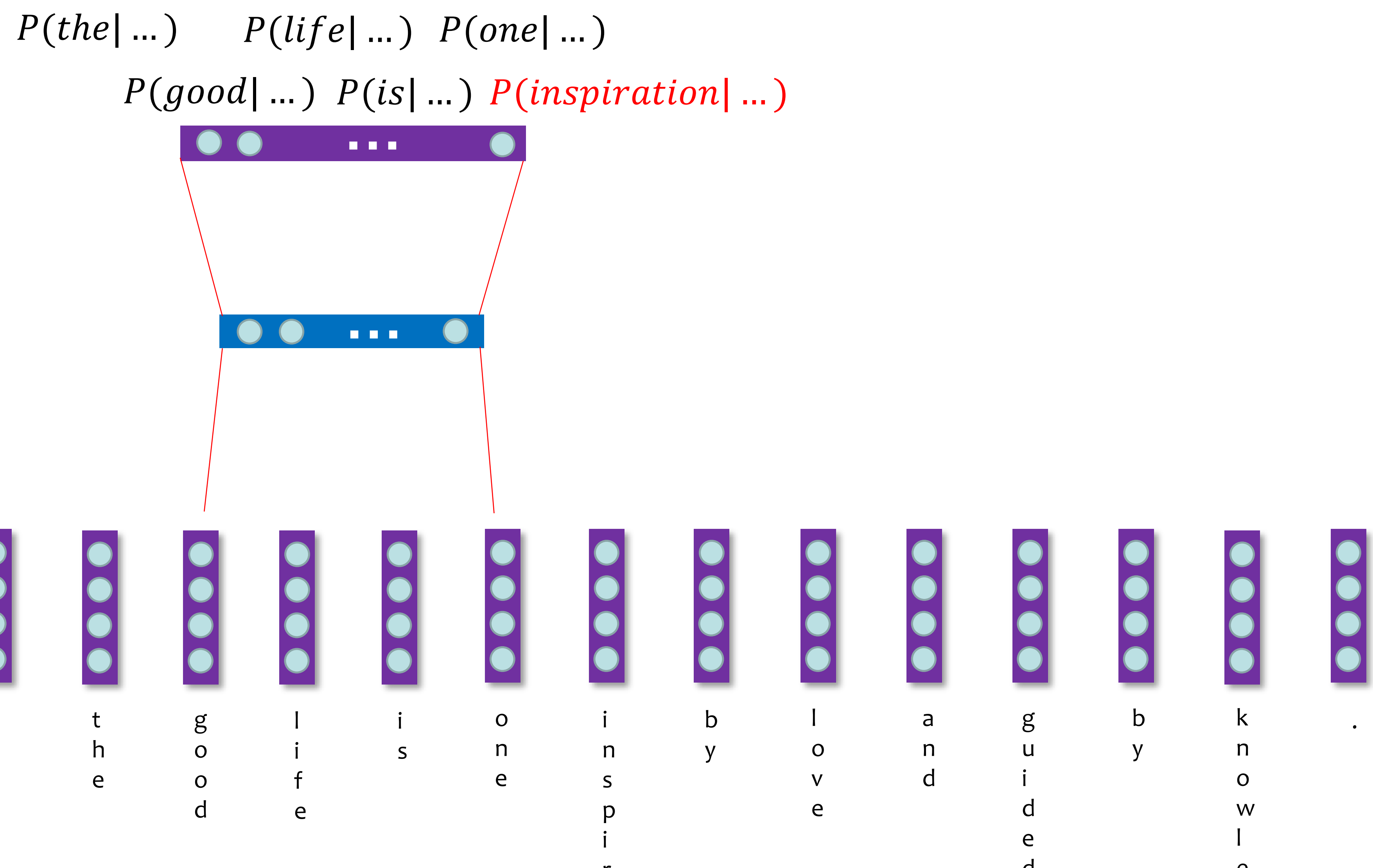
Neural Probabilistic Language Model



Neural Probabilistic Language Model



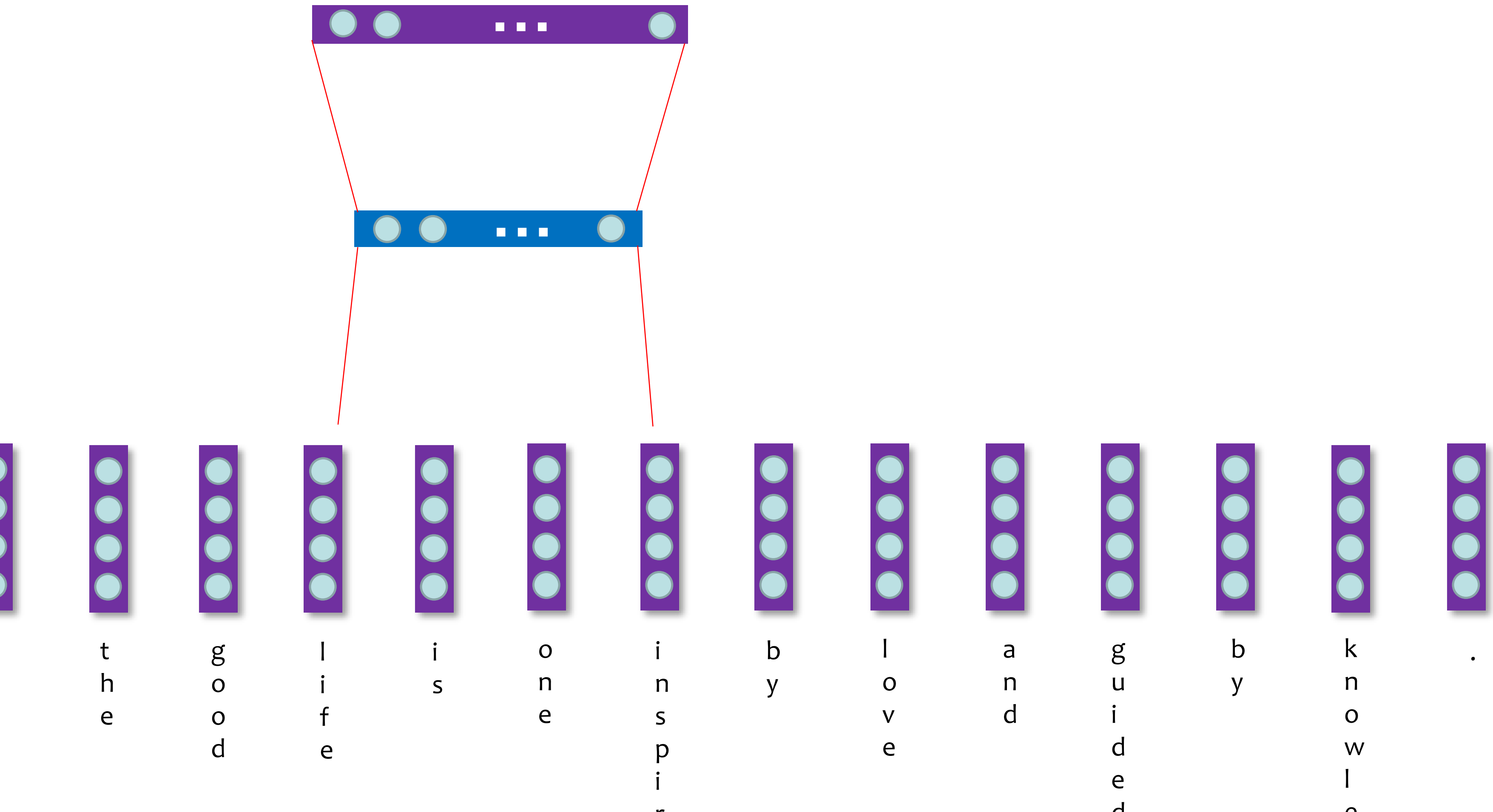
Neural Probabilistic Language Model



Neural Probabilistic Language Model

$P(the|\dots)$ $P(life|\dots)$ $P(one|\dots)$ $P(by|\dots)$

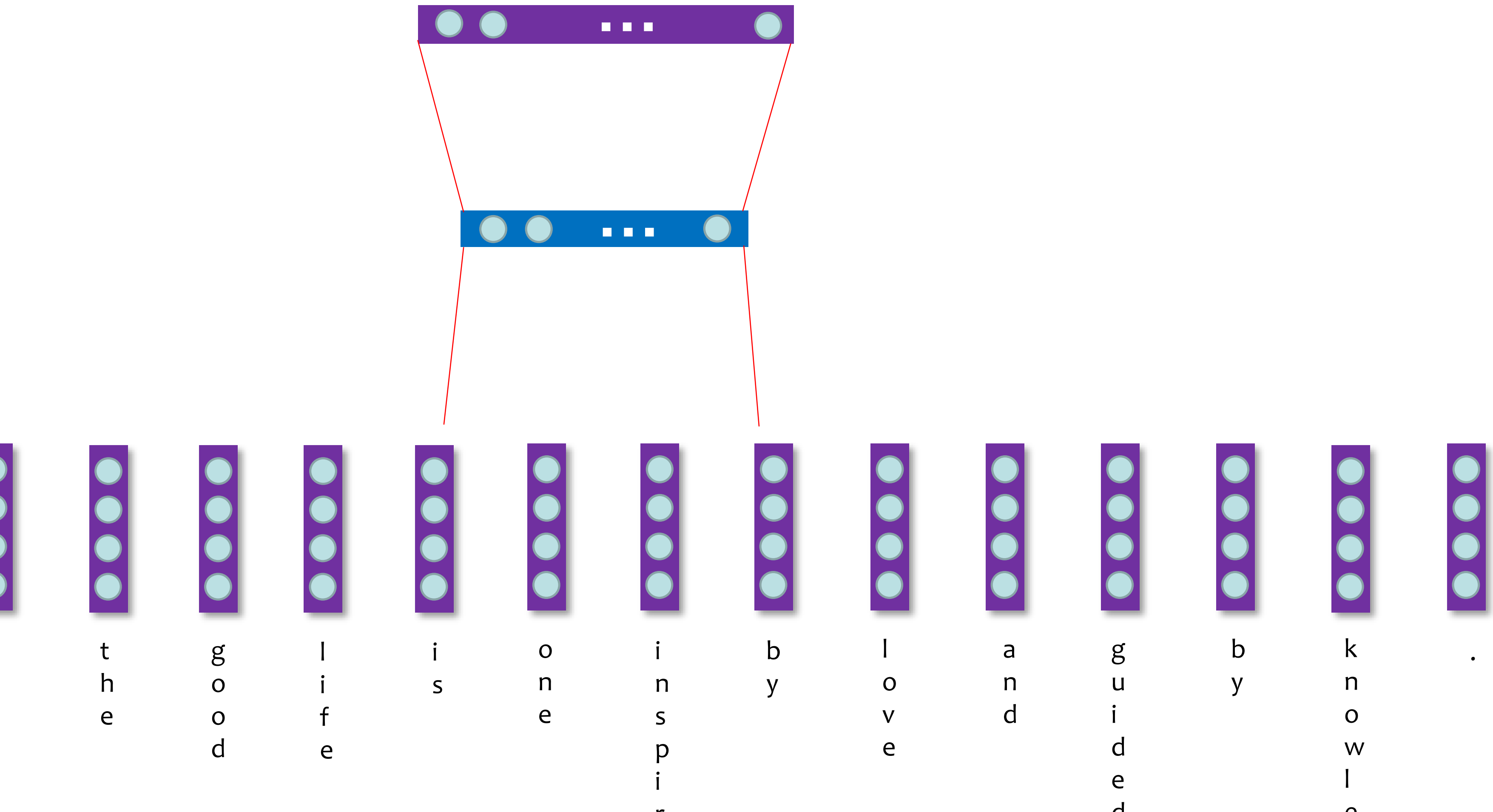
$P(good|\dots)$ $P(is|\dots)$ $P(inspiration|\dots)$



Neural Probabilistic Language Model

$P(the|\ldots)$ $P(life|\ldots)$ $P(one|\ldots)$ $P(by|\ldots)$

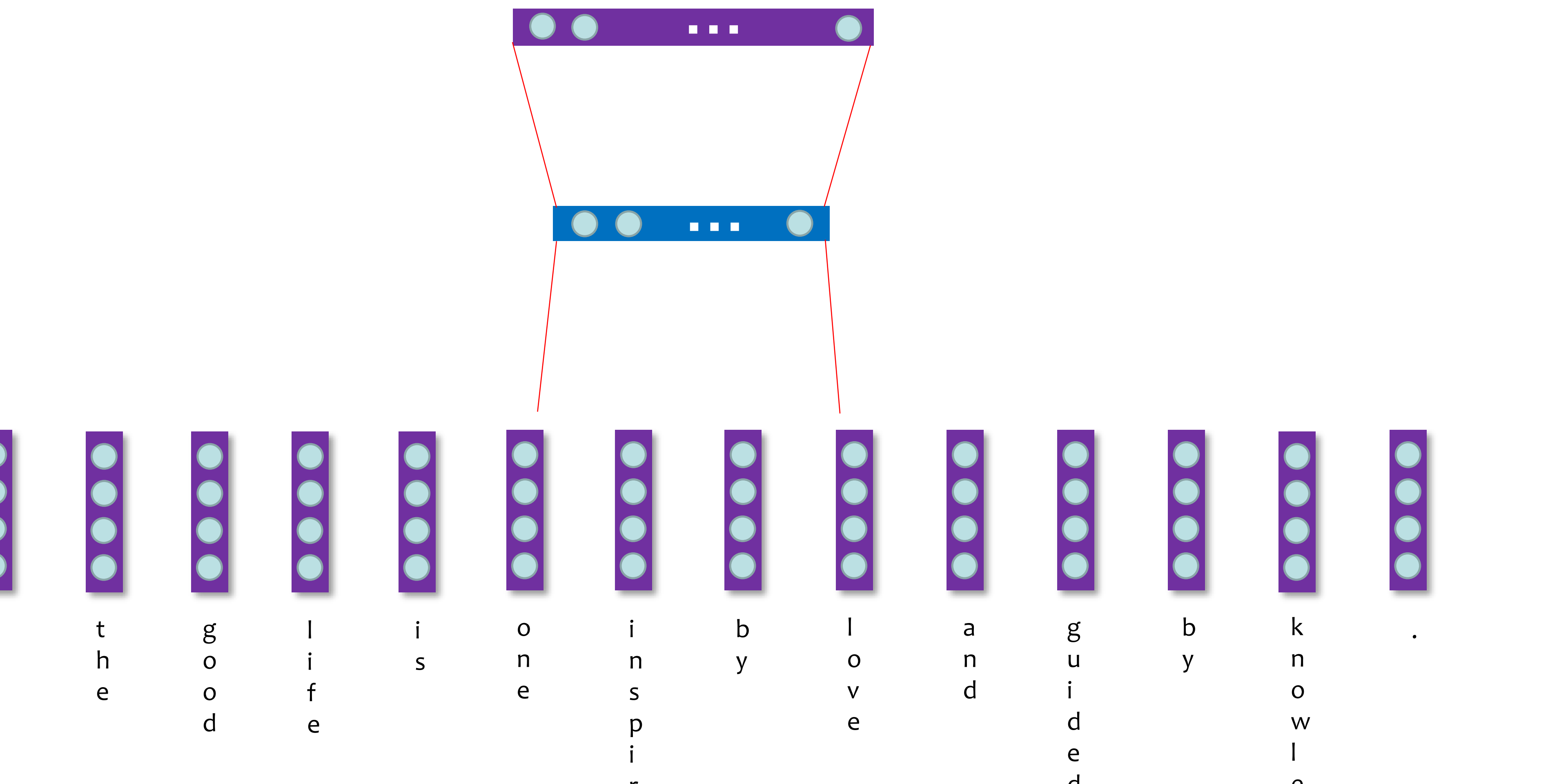
$P(good|\ldots)$ $P(is|\ldots)$ $P(inspiration|\ldots)$ $P(love|\ldots)$



Neural Probabilistic Language Model

$P(the|\ldots)$ $P(life|\ldots)$ $P(one|\ldots)$ $P(by|\ldots)$ $P(and|\ldots)$

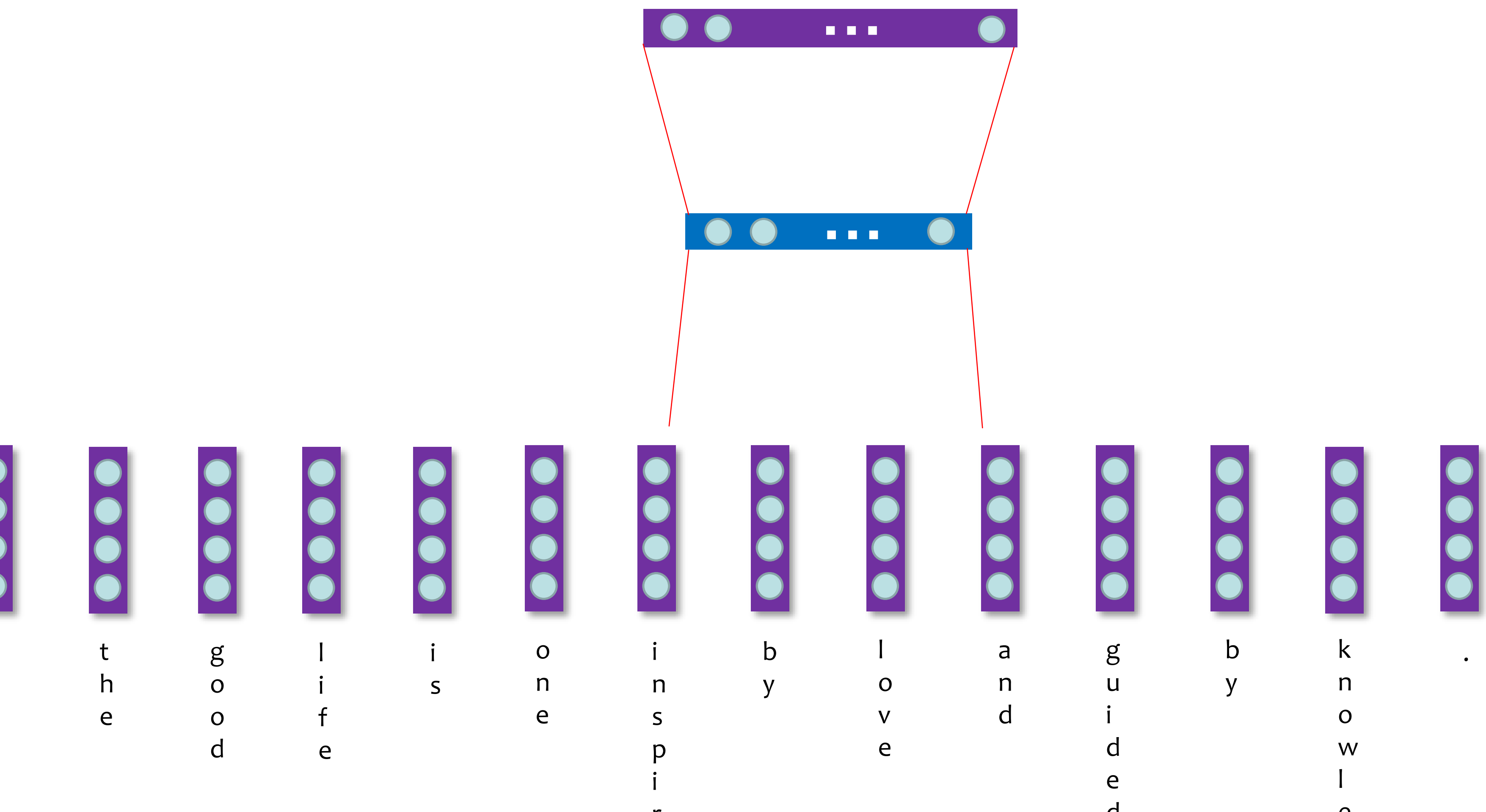
$P(good|\ldots)$ $P(is|\ldots)$ $P(inspiration|\ldots)$ $P(love|\ldots)$



Neural Probabilistic Language Model

$P(the|\ldots)$ $P(life|\ldots)$ $P(one|\ldots)$ $P(by|\ldots)$ $P(and|\ldots)$

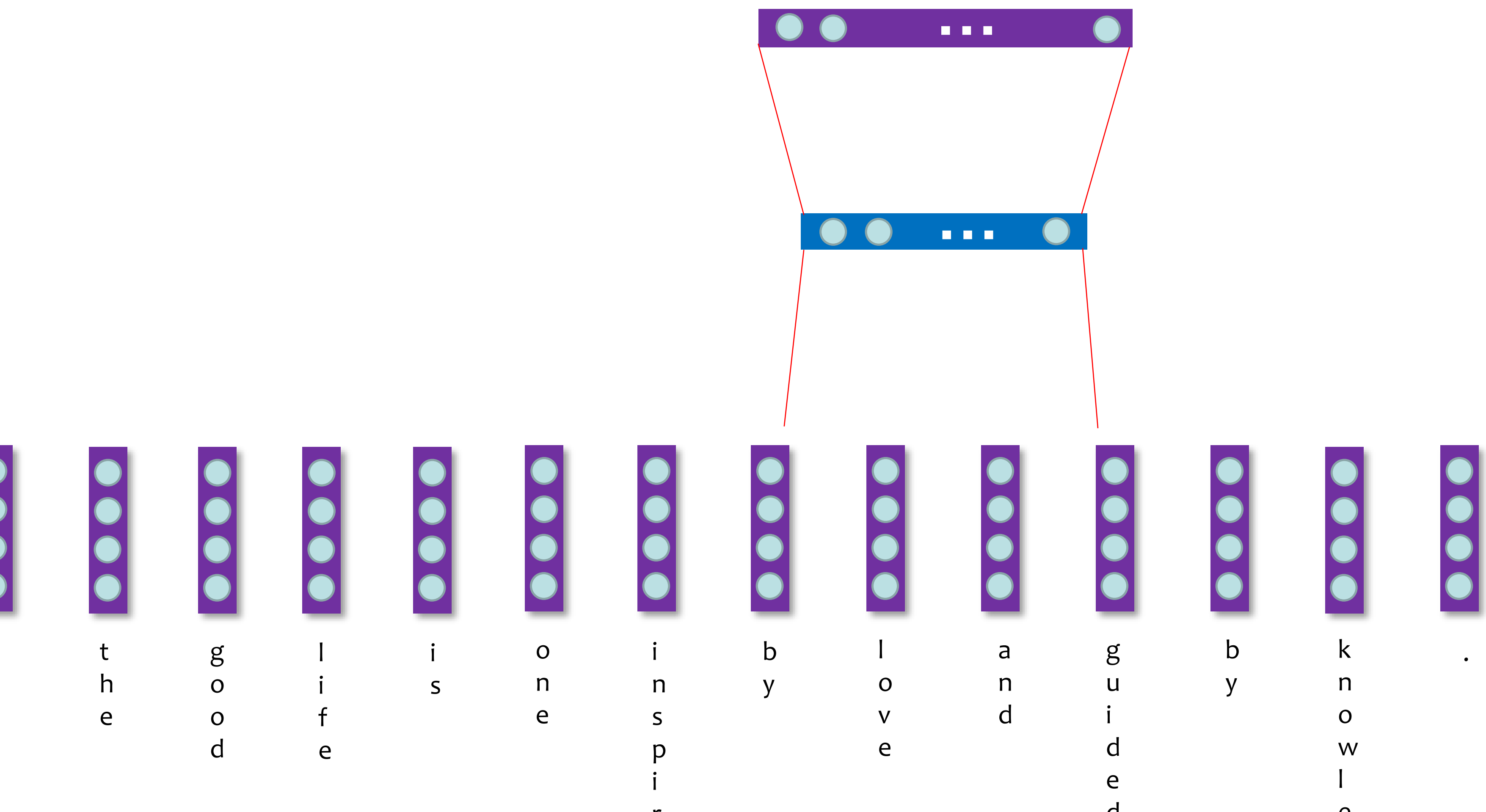
$P(good|\ldots)$ $P(is|\ldots)$ $P(inspiration|\ldots)$ $P(love|\ldots)$ $P(guided|\ldots)$



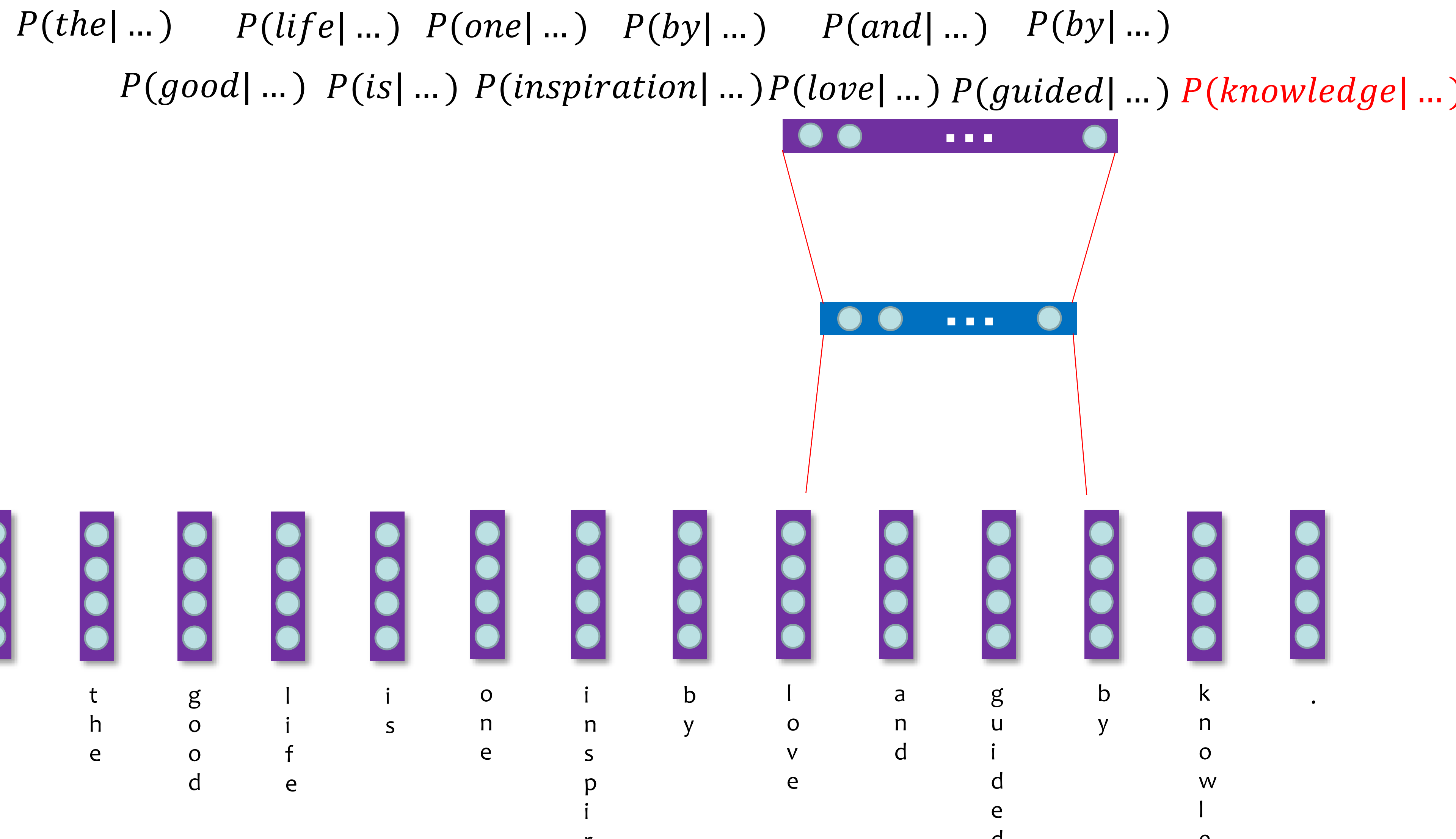
Neural Probabilistic Language Model

$P(the|\ldots)$ $P(life|\ldots)$ $P(one|\ldots)$ $P(by|\ldots)$ $P(and|\ldots)$ $P(by|\ldots)$

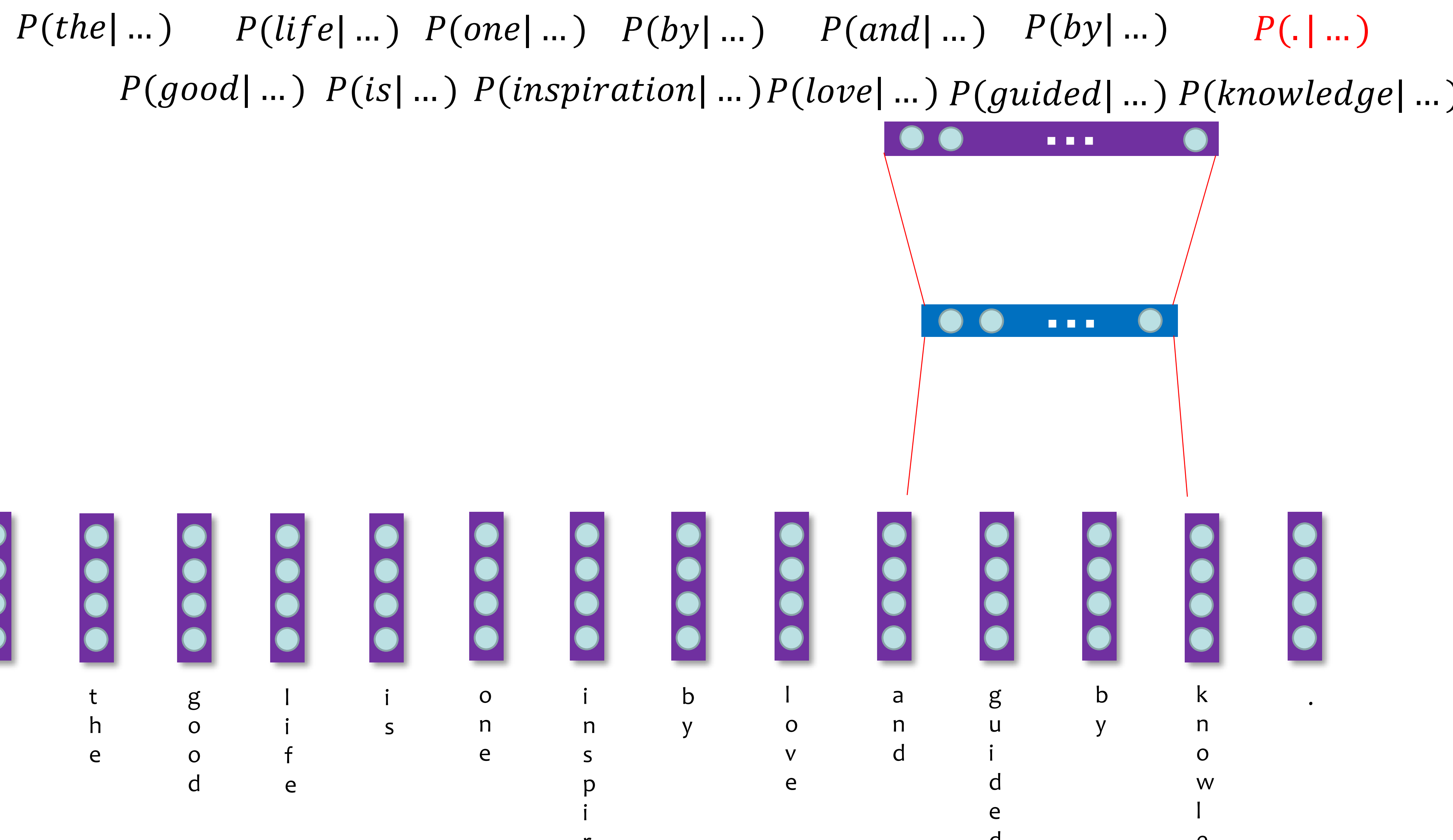
$P(good|\ldots)$ $P(is|\ldots)$ $P(inspiration|\ldots)$ $P(love|\ldots)$ $P(guided|\ldots)$



Neural Probabilistic Language Model



Neural Probabilistic Language Model

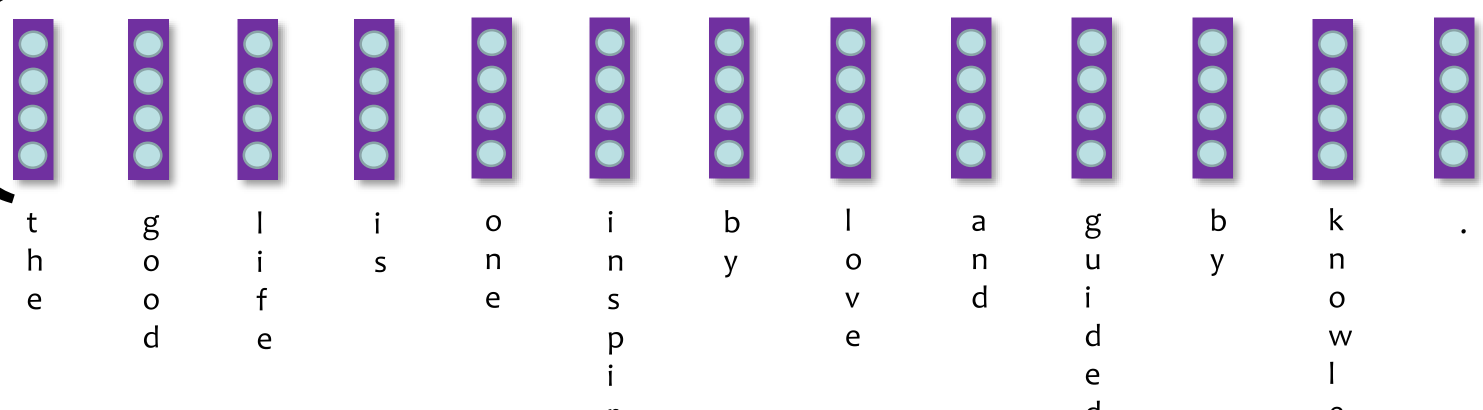


Neural Probabilistic Language Model

$$\prod \left(P(the|\ldots)\; P(good|\ldots)\; P(life|\ldots)\; P(is|\ldots)\; P(one|\ldots)\; P(inspiration|\ldots)\; P(by|\ldots)\; P(love|\ldots)\; P(and|\ldots)\; P(guided|\ldots)\; P(by|\ldots)\; P(knowledge|\ldots)\; P(.|\ldots) \right)$$

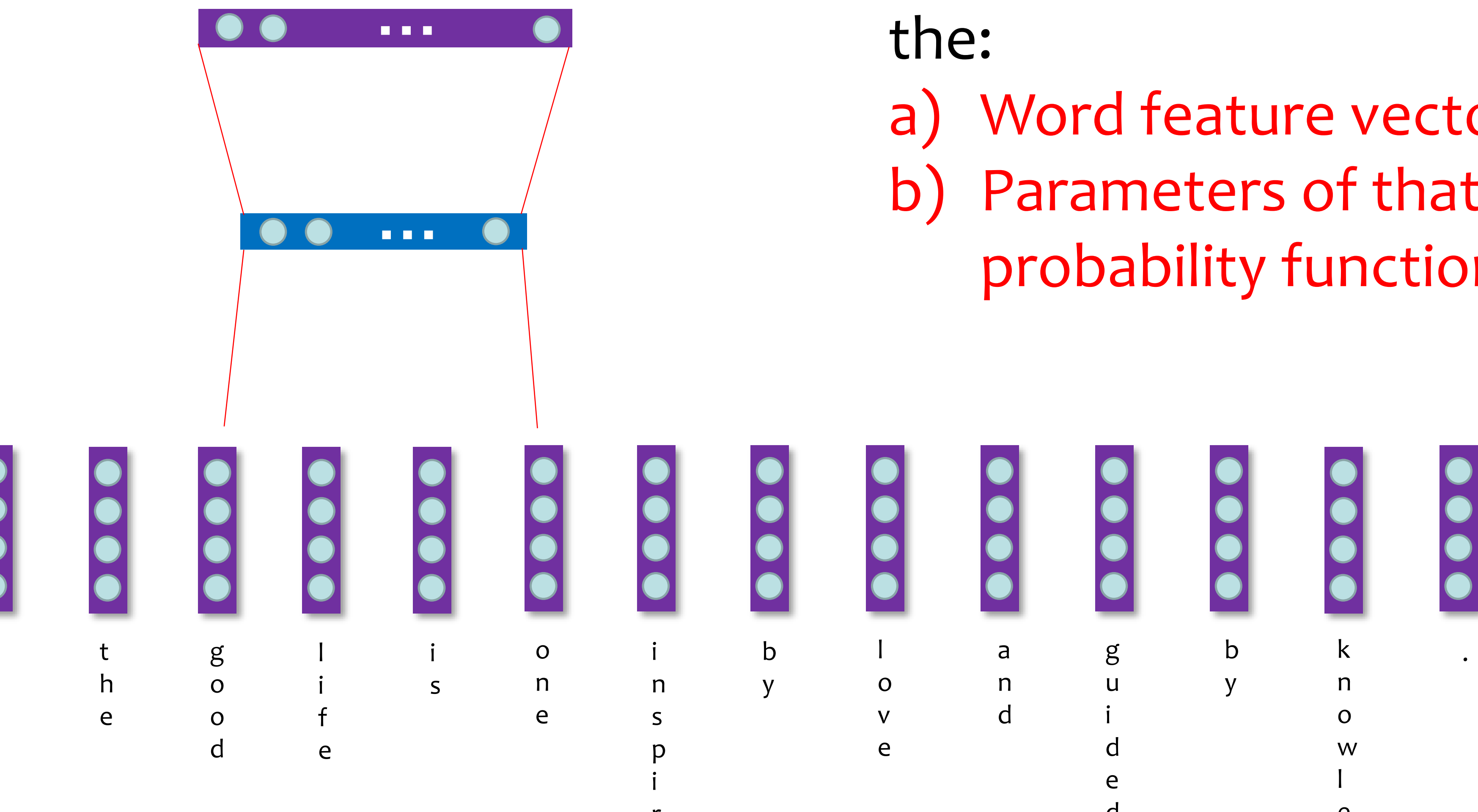
$$=$$

$$P\Bigg($$



$$\Bigg)$$

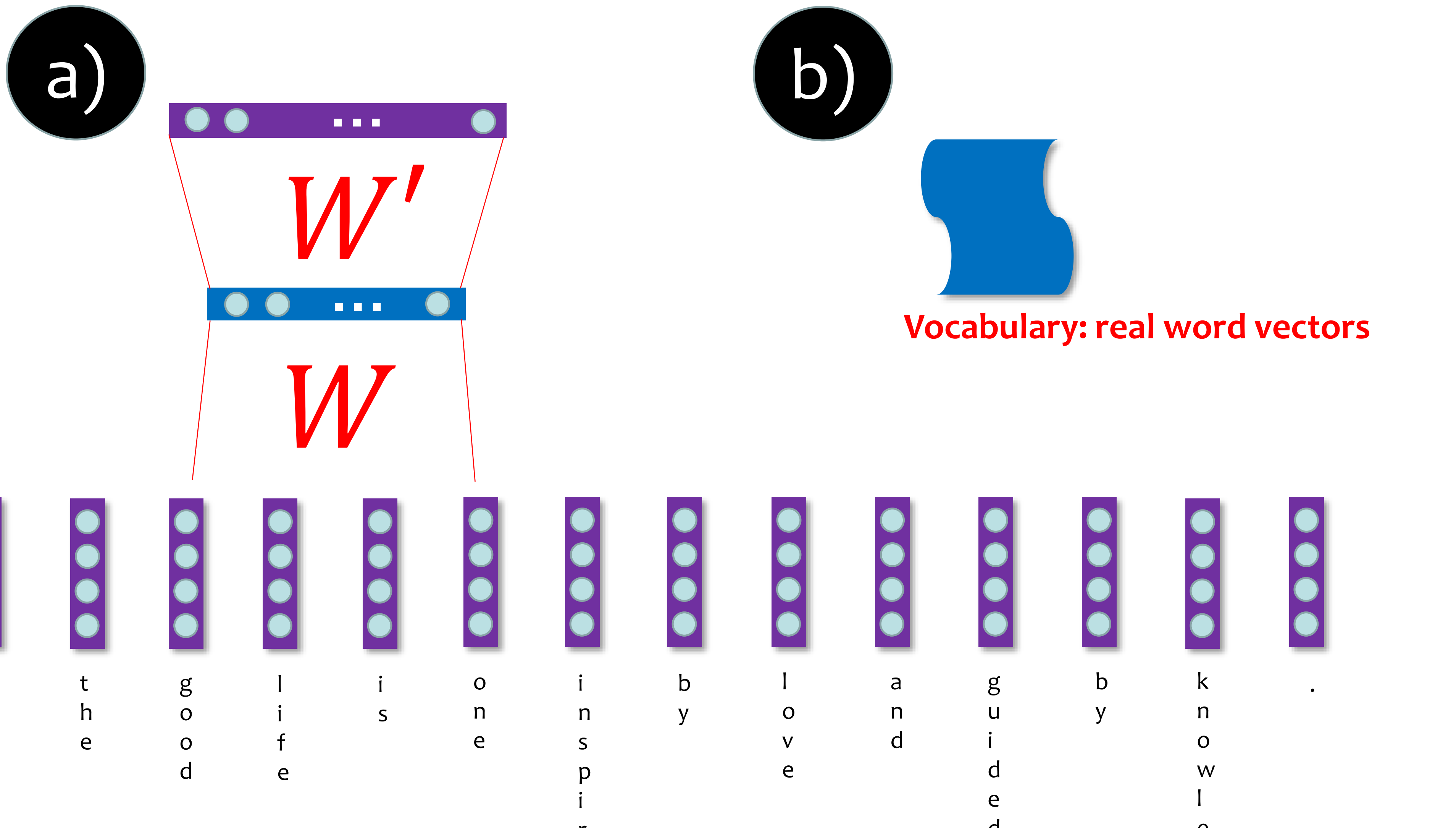
Neural Probabilistic Language Model



3. Learn simultaneously the:

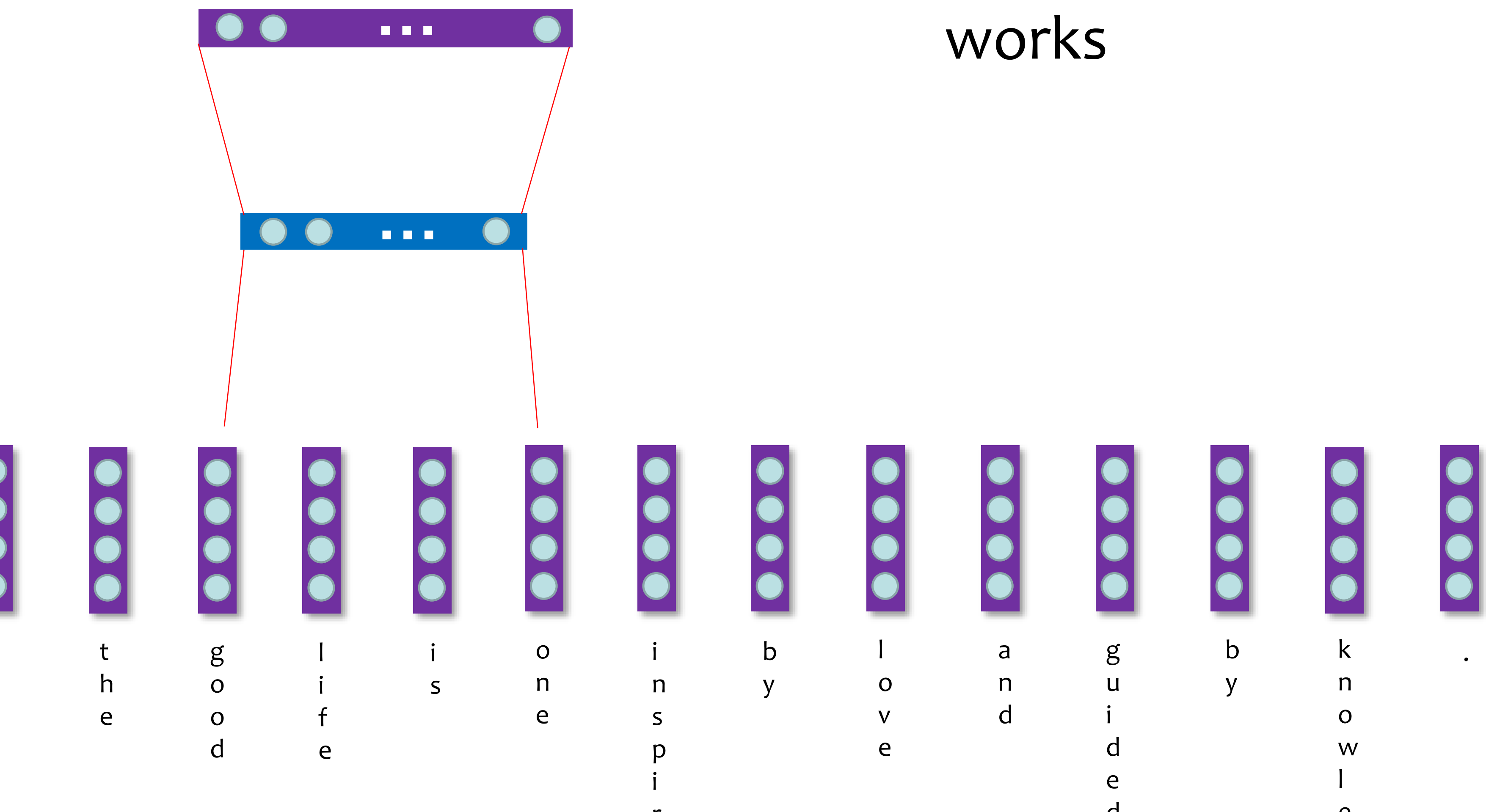
a) Word feature vectors
b) Parameters of that probability function

Neural Probabilistic Language Model



Neural Probabilistic Language Model

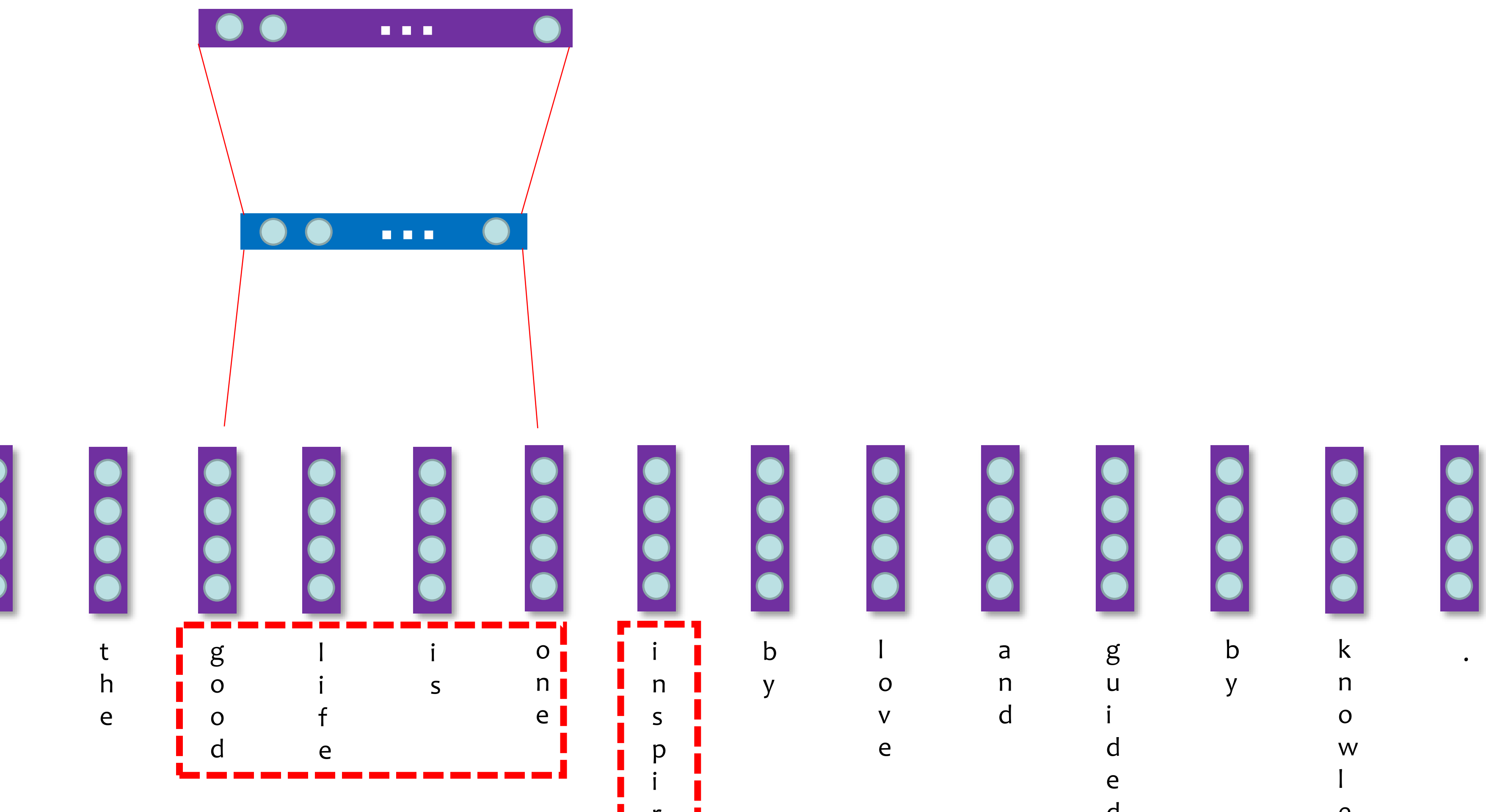
Backpropagation works



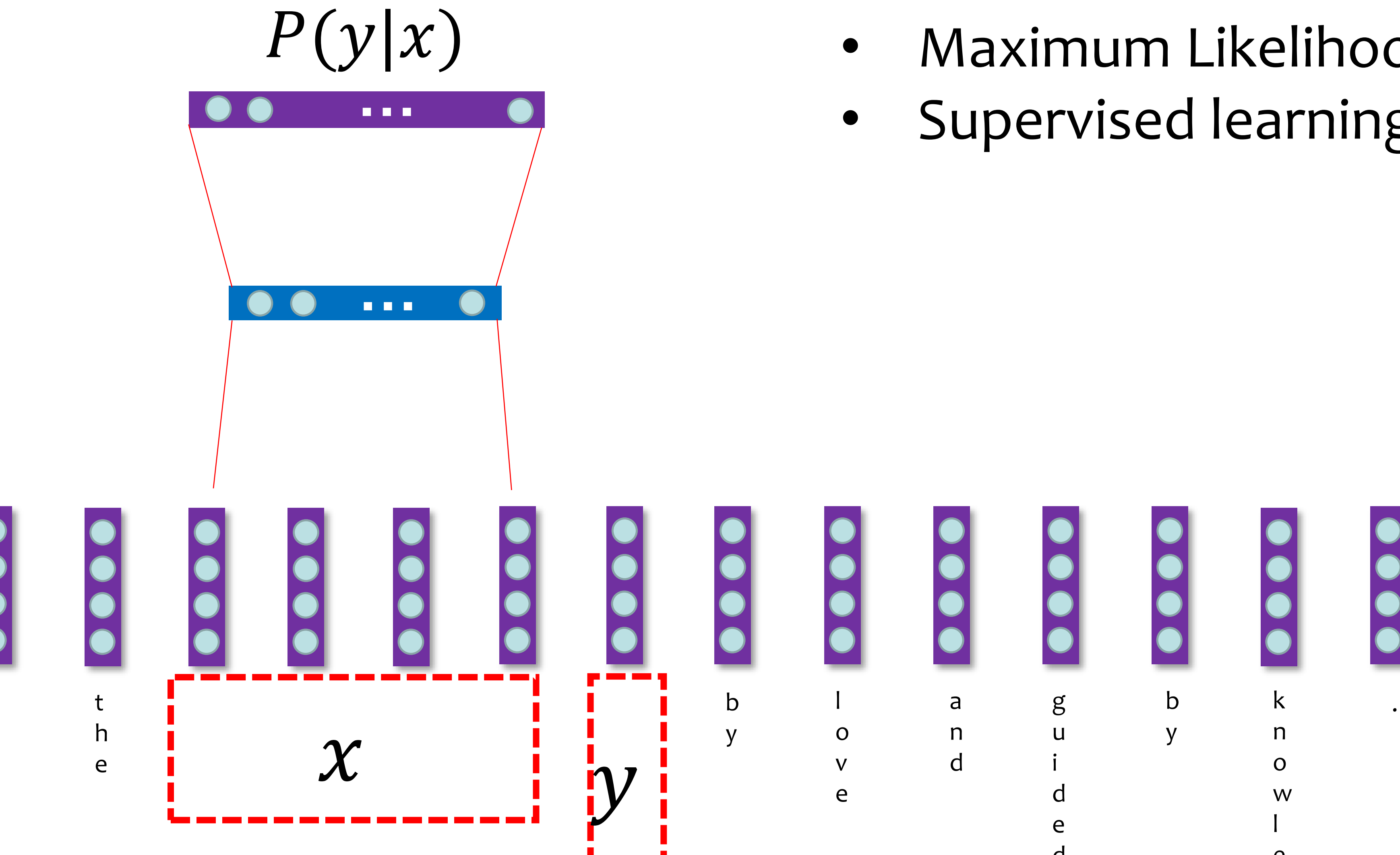
Neural Probabilistic Language Model

$P(inspiration|good, life, is, one)$

Self Supervision

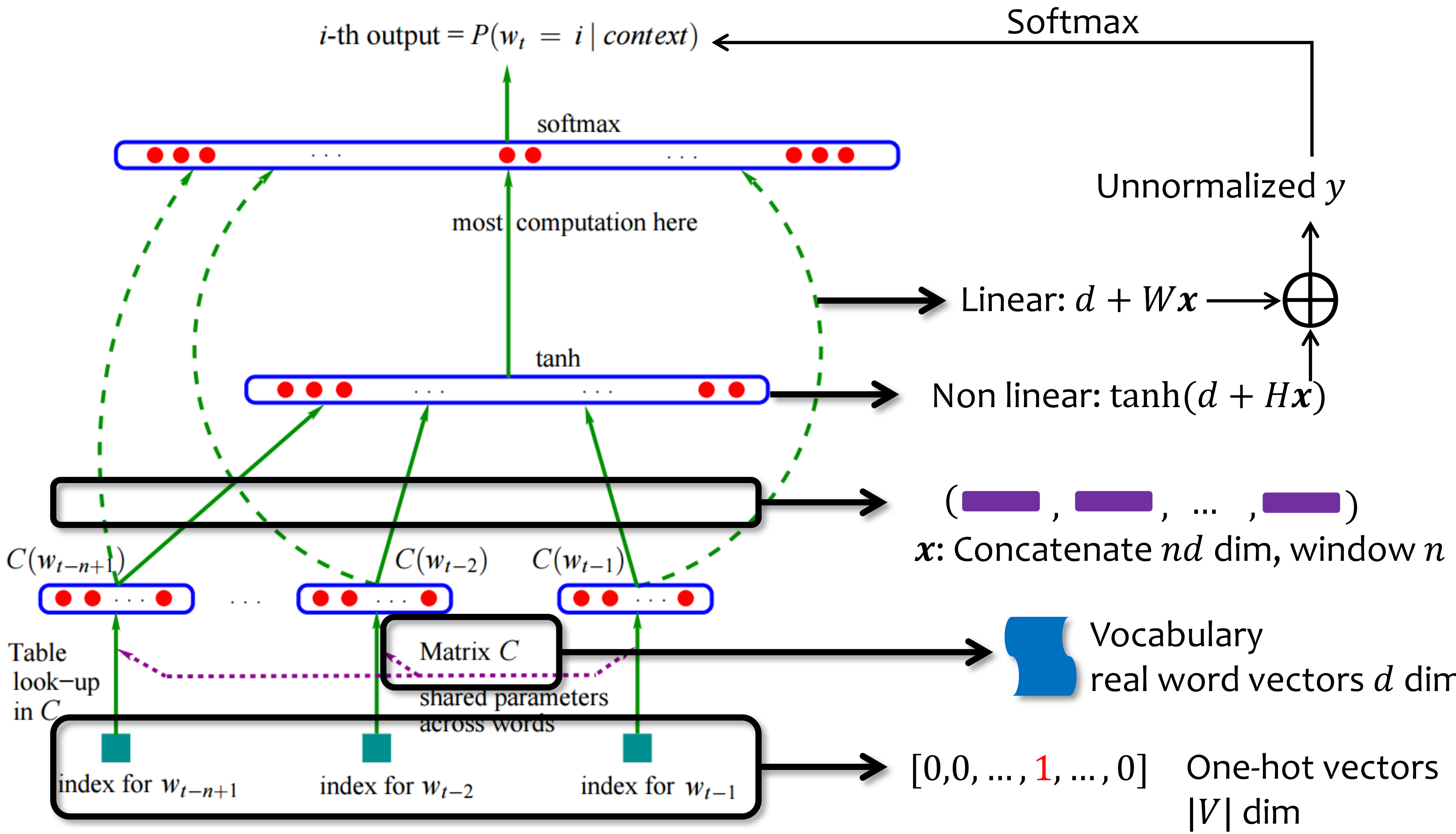


Neural Probabilistic Language Model



- Maximum Likelihood
- Supervised learning

Neural Probabilistic Language Model



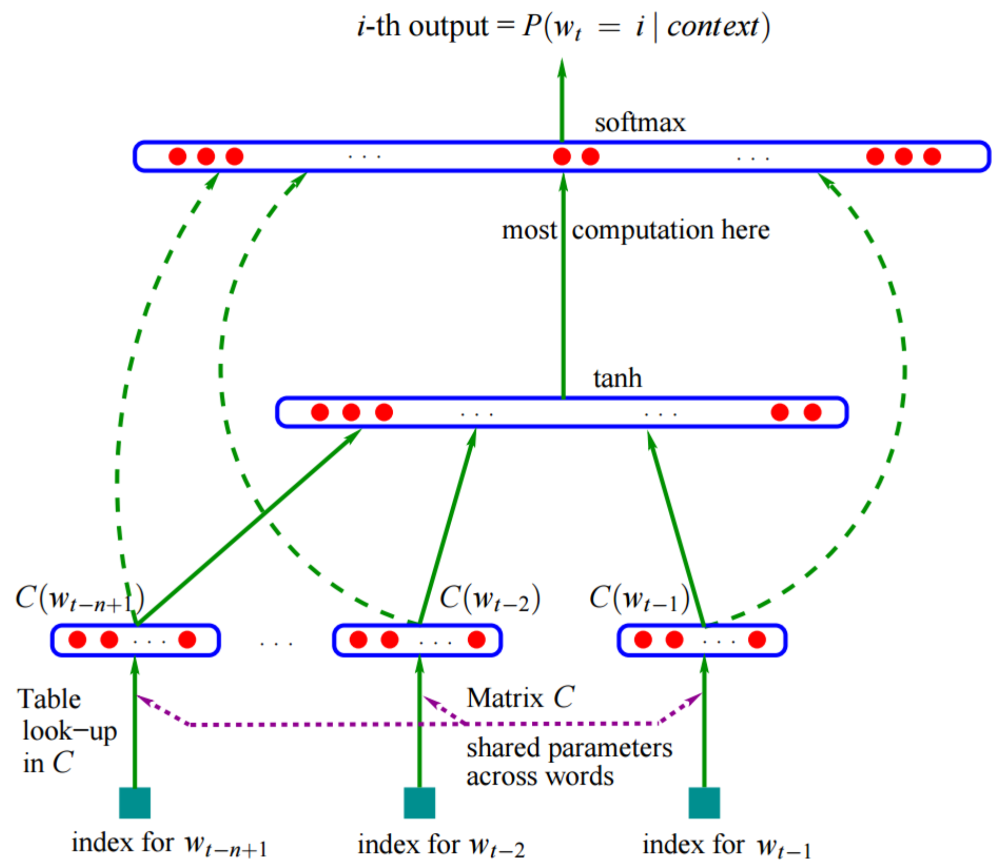
Neural Probabilistic Language Model

- Result

	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	**312**
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

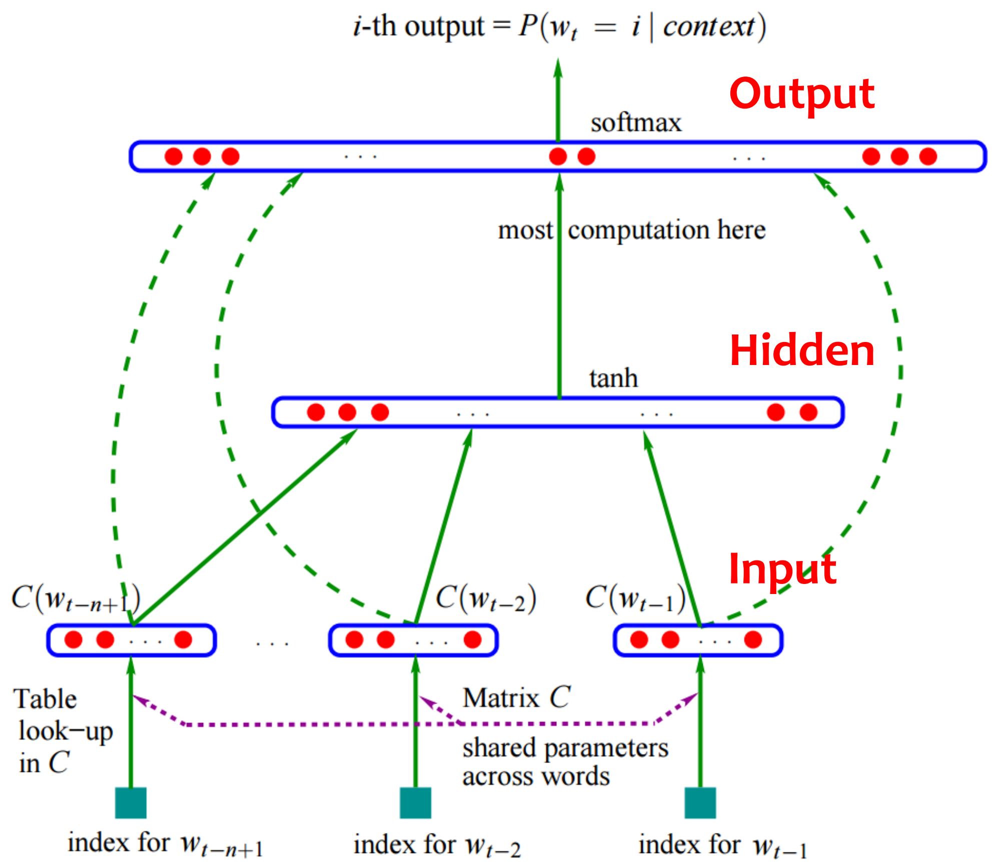
Neural Probabilistic Language Model

- Speed up
  - Why?
- Layer-wise computation
- Where is the bottleneck?



Neural Probabilistic Language Model

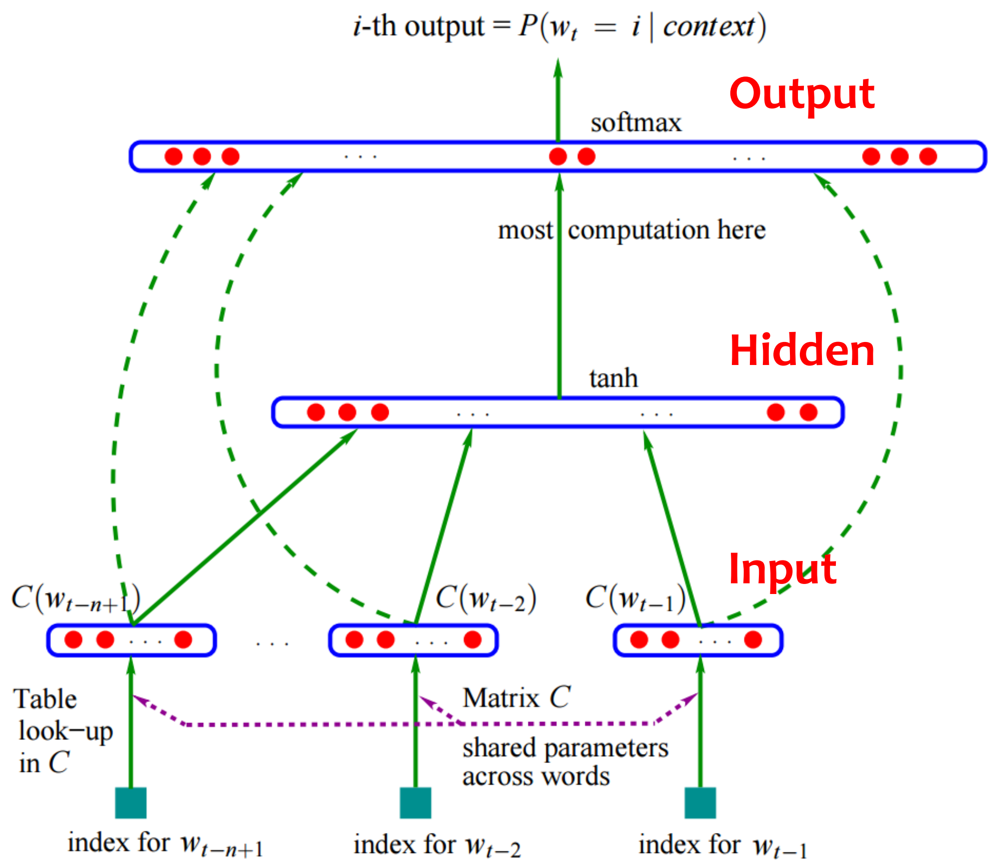
- Input
  - $C: m \times |V|$
- Input2Hid
  - $H: h \times (n-1)m$
- Hid2Out
  - $U: |V| \times h$
- Input2Out
  - $W: |V| \times (n-1)m$



Neural Probabilistic Language Model

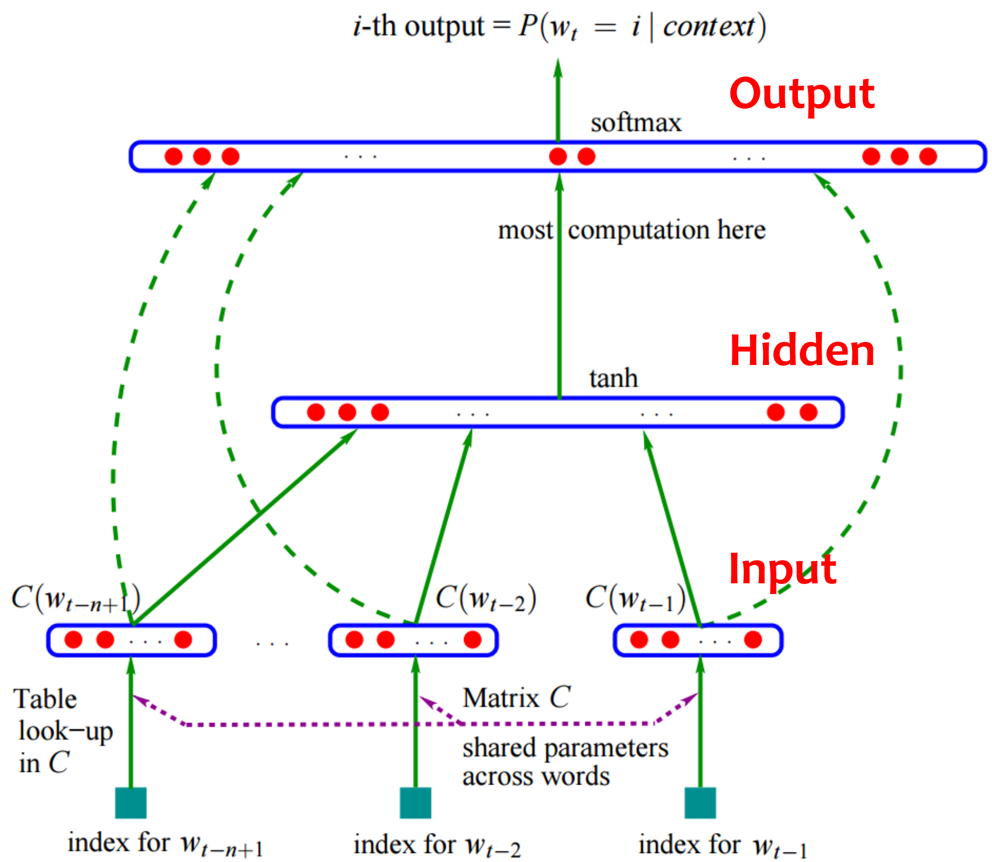
Basic computations

- Look-up
- Add
- Multiply
- Tanh
- Concat
- Divide



Neural Probabilistic Language Model

- Input2Out: $2|V|(n-1)m$
  - $W \cdot \boldsymbol{x}$
- Hid2Out: $2|V|h$
  - $U \cdot \tanh(\ldots)$
- Input2Hid: $2h \times (n-1)m$
  - $H \cdot \boldsymbol{x}$
- $n-1$ Look-ups



Neural Probabilistic Language Model

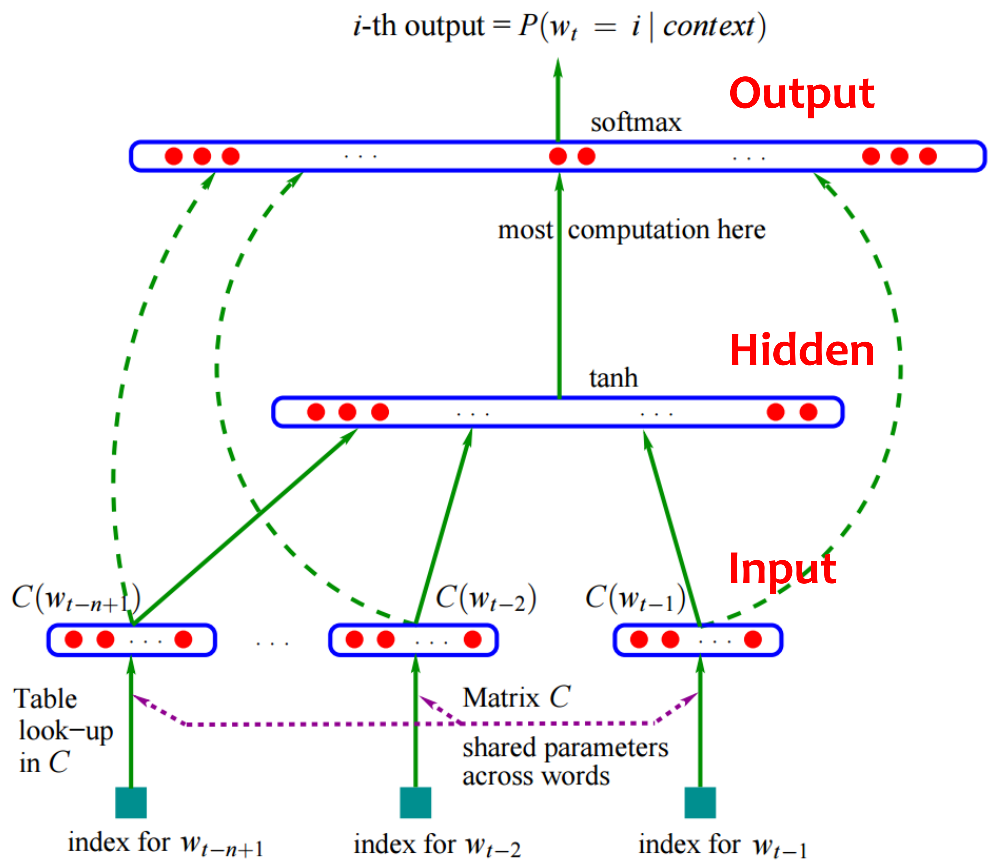
- $|V| \gg m, h, n$

- Input2Out: $2|V|(n-1)m$
  - $W \cdot \boldsymbol{x}$

- Hid2Out: $2|V|h$
  - $U \cdot \tanh(\ldots)$

- Input2Hid: $2h \times (n-1)m$
  - $H \cdot \boldsymbol{x}$

- $n-1$ Look-ups



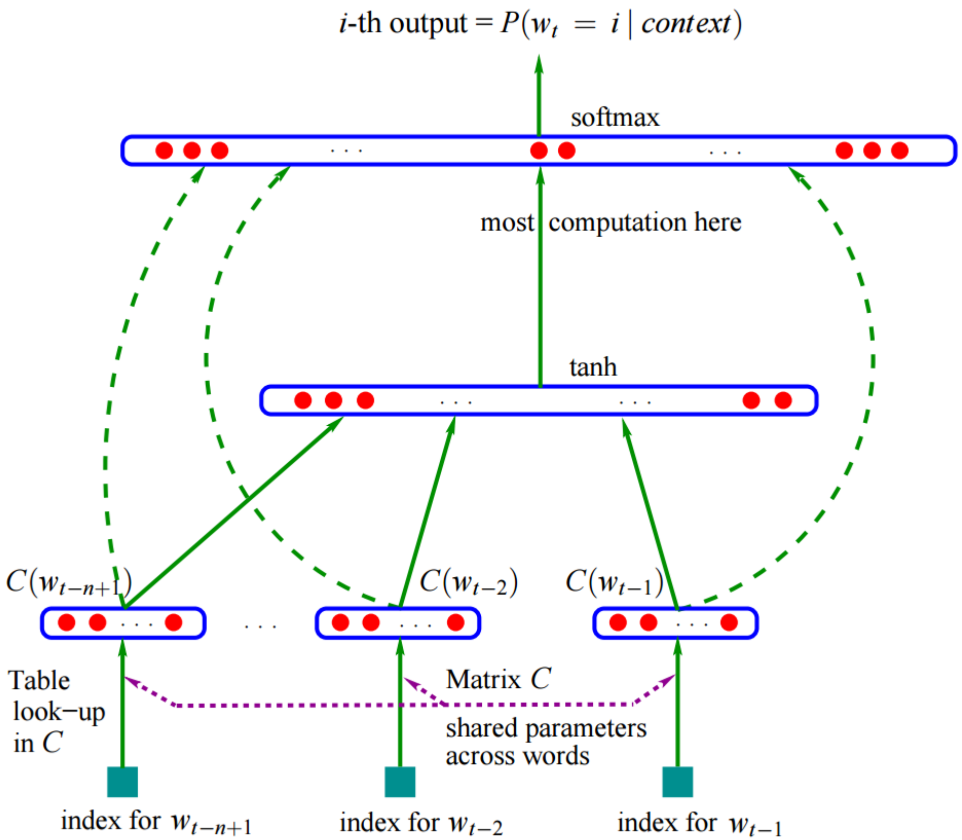
Neural Probabilistic Language Model

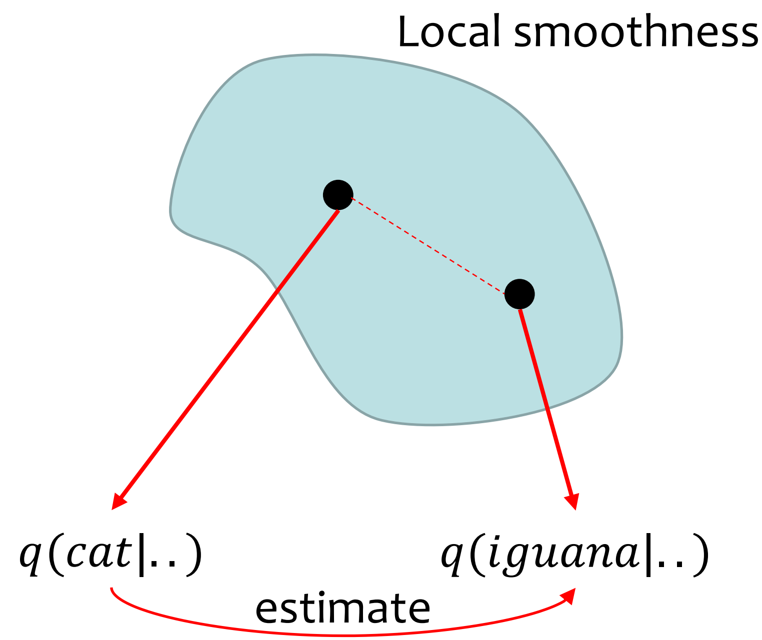
Future work:

- Representing conditional probability with a tree structure
- Propagating gradients only from a subset of the output words

Bonus!

- Think, think, imagine, imagine!
  - Where do them first share information?





The word *cat iguana* and *animal* need to **share** their information of cooccurrence

Outline

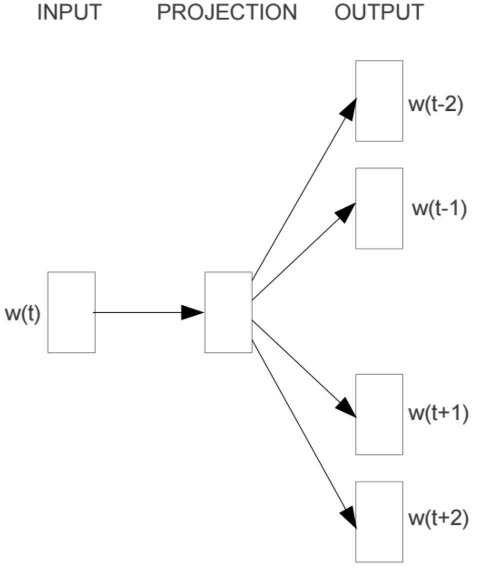
PART II

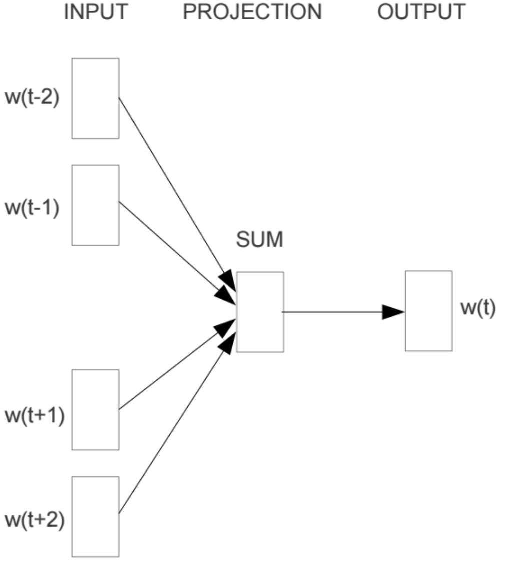
- Word Embedding – Basic Models
  - Word2vec
  - GloVe
- Representation Learning of Lexical Meaning
  - Sentiment Embedding
  - Topical Embedding
  - Discourse Relation-aware Embedding

Word2vec

- CBOW
  - Continuous Bag of Words
- Skip-gram

Tomas Mikolov





Word2vec

Tomas Mikolov

Tomas Mikolov

✉ 追蹤 ▾

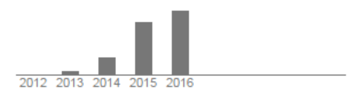
Research scientist, Facebook
Artificial Intelligence, Machine Learning, Language Modeling
在 fb.com 的電子郵件地址已通過驗證

標題 1–20	引用次數	年份
Distributed representations of words and phrases and their compositionality T Mikolov, I Sutskever, K Chen, GS Corrado, J Dean Advances in neural information processing systems, 3111-3119	2480	2013
Efficient estimation of word representations in vector space T Mikolov, K Chen, G Corrado, J Dean arXiv preprint arXiv:1301.3781	2308	2013
Recurrent neural network based language model. T Mikolov, M Karafiát, L Burget, J Cernocký, S Khudanpur Interspeech 2, 3	897	2010
Linguistic Regularities in Continuous Space Word Representations. T Mikolov, W Yih, G Zweig HLT-NAACL 13, 746-751	705	2013
Distributed Representations of Sentences and Documents. QV Le, T Mikolov ICML 14, 1188-1196	674	2014
Extensions of recurrent neural network language model T Mikolov, S Kombrink, L Burget, J Černocký, S Khudanpur 2011 IEEE International Conference on Acoustics, Speech and Signal ...	361	2011
On the difficulty of training recurrent neural networks. R Pascanu, T Mikolov, Y Bengio ICML (3) 28, 1310-1318	308	2013
Devise: A deep visual-semantic embedding model A Frome, GS Corrado, J Shlens, S Bengio, J Dean, T Mikolov Advances in neural information processing systems, 2121-2129	282	2013
Statistical Language Models Based on Neural Networks T Mikolov Ph. D. thesis, Brno University of Technology	238	2012
Exploiting similarities among languages for machine translation T Mikolov, QV Le, I Sutskever arXiv preprint arXiv:1309.4168	214	2013
Strategies for training large scale neural network language models T Mikolov, A Deoras, D Povey, L Burget, J Černocký Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on ...	153	2011

Google 學術搜尋

建立我自己的個人學術檔案

引文指數	全部	自 2011 年
引文	9877	9809
H 指數	25	25
i10 指數	36	35



共同作者 查看所有共同作者...

Lukas Burget
Jeff Dean
Jan Cernocky
Greg Corrado
Kai Chen
Stefan Kombrink
Martin Karafiat
Ilya Sutskever
Anoop Deoras
Armand Joulin
Sanjeev Khudanpur
Scott Wen-tau Yih
Jon Shlens
Samy Bengio
Andrea Frome
Quoc V. Le
Yoshua Bengio
Oldřich Plchot
Pavel Matejka
Razvan Pascanu

Word2vec

Tomas Mikolov

Who is Tomas Mikolov and why did he leave Google?

I've been blown away by Mikolov's work, starting with his doctoral work on language modeling.

a) how did a graduate of a relatively unknown university become *so* influential in ML?

b) Mikolov recently left Google Brain, notably releasing word2vec, for Facebook AI Research. Why leave such a strong team?

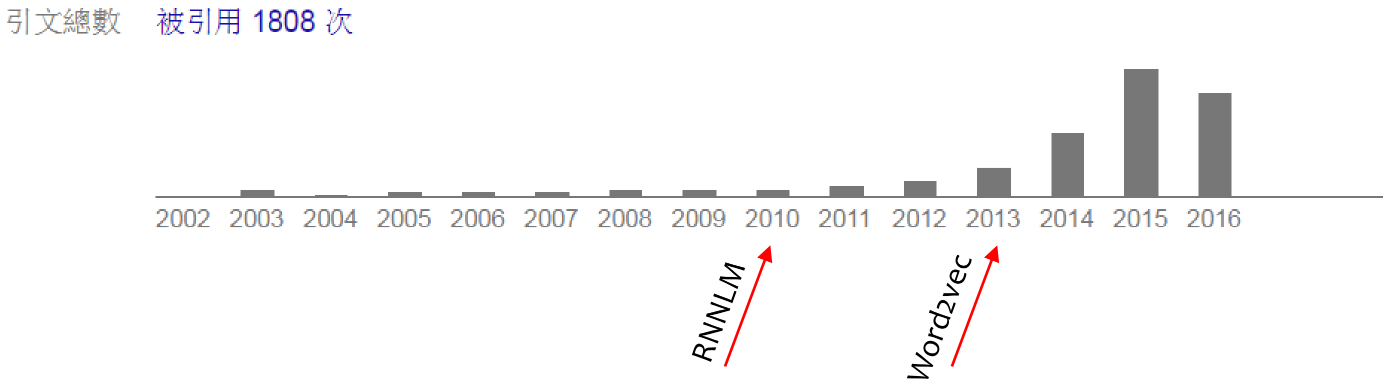
How did a graduate of a relatively unknown university became so influential in ML?

Coming from the non-Ivy League university doesn't prevent you from succeeding. Of course, being taught by top researchers helps and in many cases the equipment matters a lot. But in math and CS it's sheer determination and constructive approach that allows you to have an impact and get to the right environment. IMO, a certain psychological bias also takes place here - people are intimidated by the results produced in top universities and it sets back their passion. We talked a bit about the general type of people in ML research - universities, fields of study they come from and so on. He is opposed to any elitism in research departments and grad schools and says that more young people should be given such opportunities based on merit.

Word2vec

Tomas Mikolov

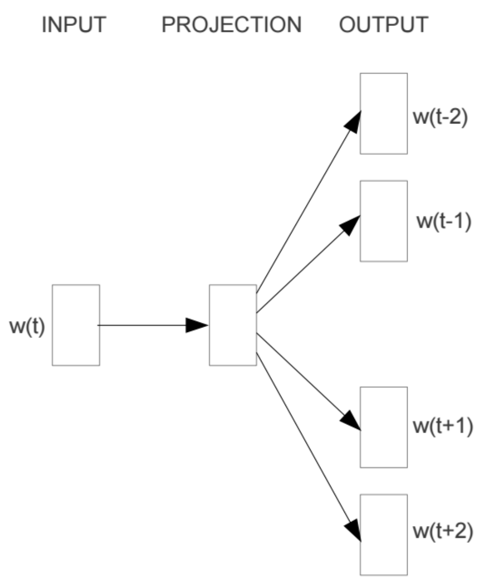
A Neural Probabilistic Language Model, Bengio et al. 2003



- Bengio's seminal work has gone through very few attentions over almost ten years. However, gold will never vanish through time.
- Mikolov found that glittering gold, and first he designed and implemented a Recurrent Neural Network Language Model in 2010
- It seems to him that computational efficiency is more important than modelling longer context, so he proposed Word2vec in 2013, and seek intuitive visualization of Word Embedding.
- Dig out gold with sharp eyesight!

Word2vec

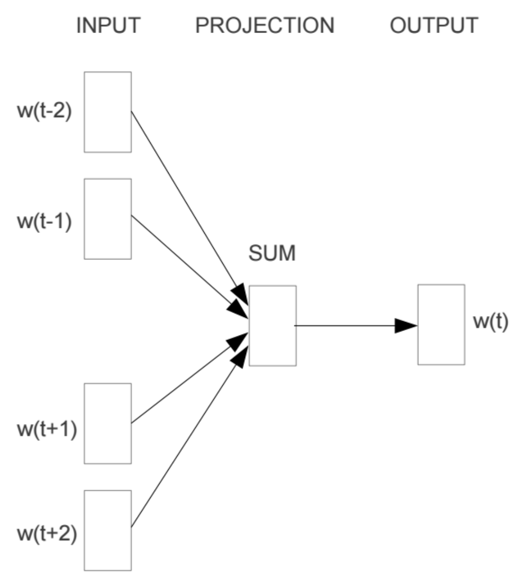
middle word → context words



Skip-gram

$P(context|w_m)$

context words → middle word

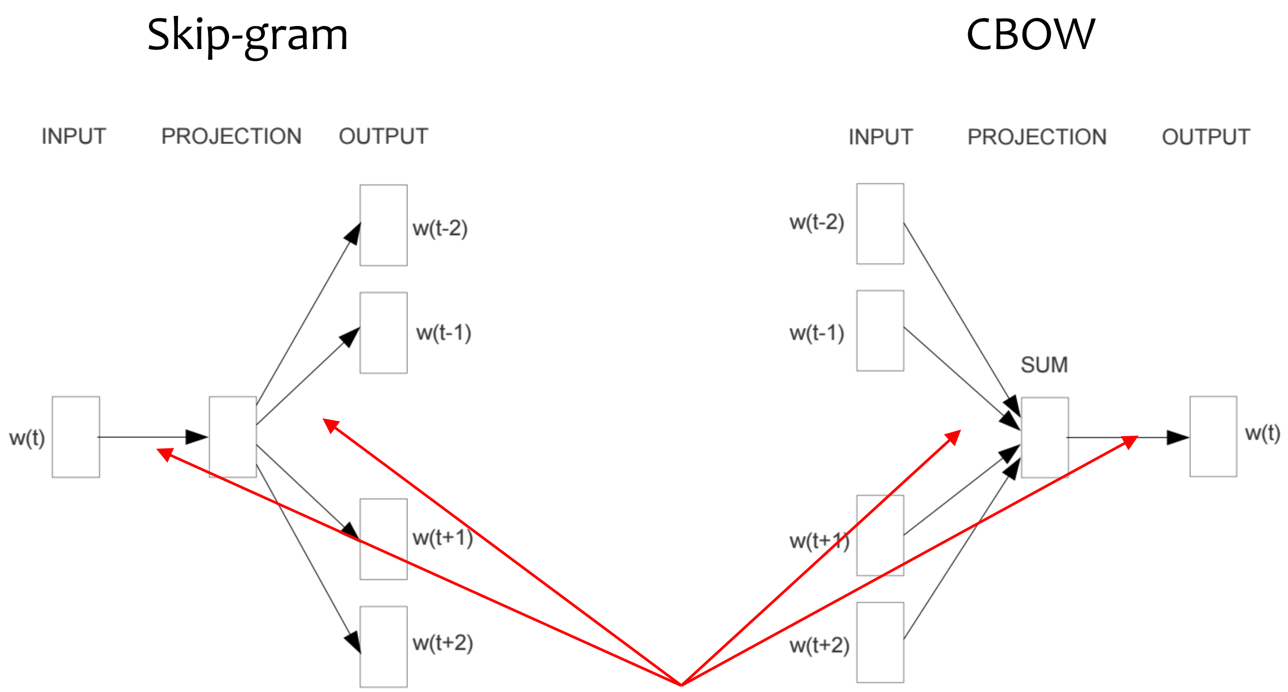


CBOW

$P(w_m|context)$

$$\underline{w_1, w_2, w_3, w_4}, w_5, \underline{w_6, w_7, w_8, w_9}$$

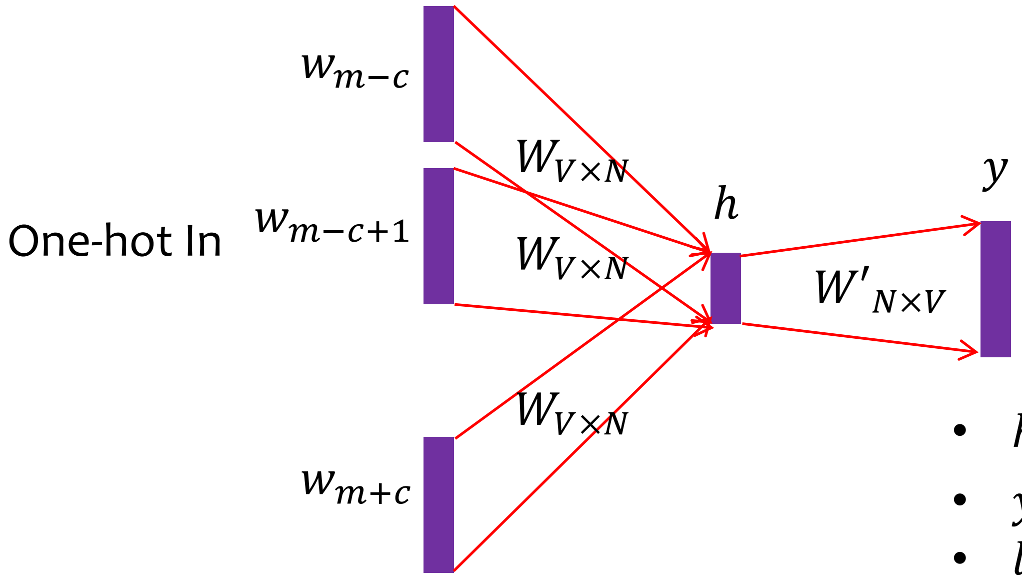
Word2vec



- Word vectors are here, weights!

Continuous Bag of Word

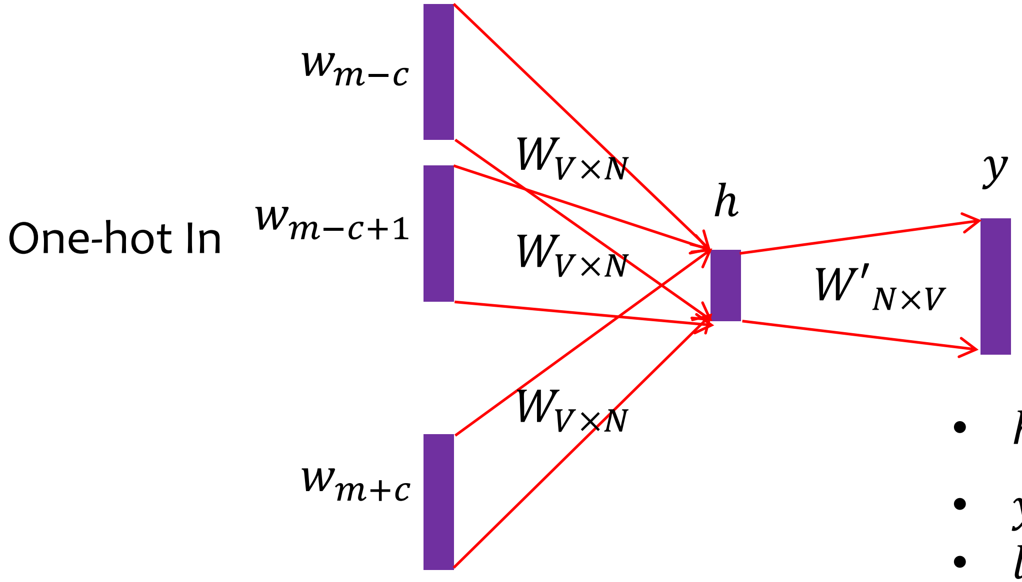
- Given supervision signal: context C, middle word w_m pairs, that is $(C, w_m)^i$
- We use a FFNN to model conditional probability, $P(w_m | w_{m-c}, \dots, w_{m+c})$



- $h = \frac{1}{2c} W^T (w_{m-c} + \cdots + w_{m+c})$
- $y = \text{softmax}(W'h)$
- $loss = -log y_{w_m}$, to be minimized

Continuous Bag of Word

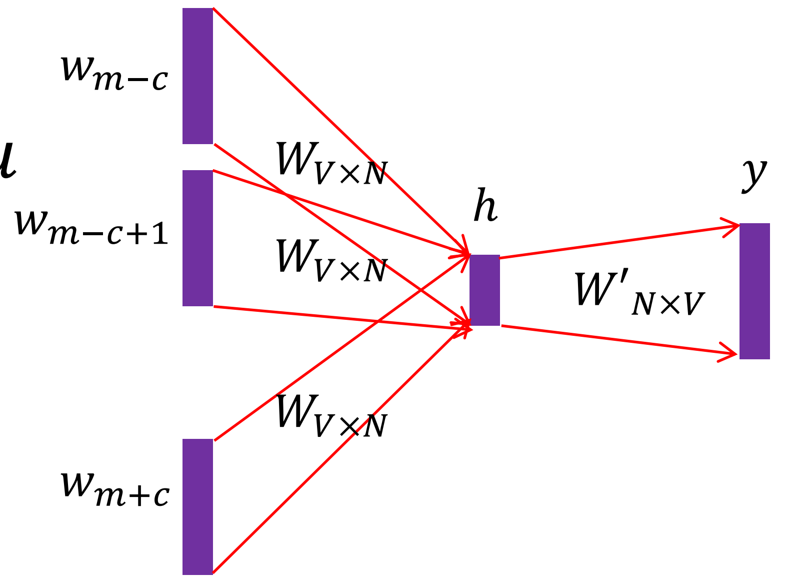
- Cost function – Again Maximum Likelihood
  - $Cost = -\sum_{(C,w_m)^i} loss^i = -\sum_{(C,w_m)^i} log y_{w_m}^i$



- $h = \frac{1}{2c} W^T(w_{m-c} + \cdots + w_{m+c})$
- $y = \mathrm{softmax}(W'^T h)$
- $loss = -log y_{w_m}$, to be Minimized

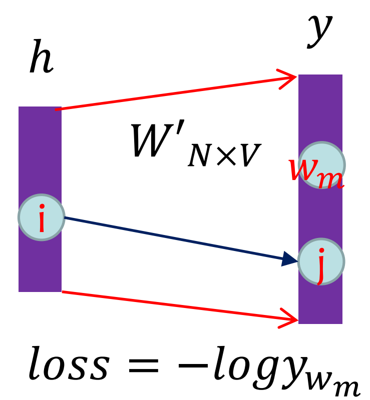
Continuous Bag of Word

- Some Notation convenience
  - v_{w_m} is the w_m row of W, input embedding of word w_m
  - v'_{w_m} is the w_m column of W', output embedding of word w_m
  - Pre-activation of output is u



Continuous Bag of Word

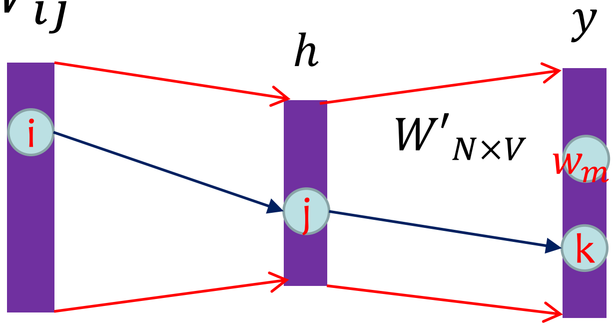
- Let's take gradient of the loss w.r.t. W'_{ij}
  - $loss = -log y_{w_m} = log \Sigma_w e^{u_w} - u_{w_m}$
  - $\frac{\partial loss}{\partial W'_{ij}} = \frac{\partial \left(log \Sigma_w e^{u_w} - u_{w_m} \right)}{\partial u_j} \frac{\partial u_j}{\partial W'_{ij}}$
  - $= \left(\frac{e^{u_j}}{\Sigma_w e^{u_w}} - \boldsymbol{I}[w_m = j] \right) h_i$
  - $= \left(y_j - \boldsymbol{I}[w_m = j] \right) h_i$



- So, $loss$ grad w.r.t. one output vector
  - $\frac{\partial loss}{\partial W'_{\cdot j}} = \left(y_j - \boldsymbol{I}[w_m = j] \right) h$, this is for every $w'_{\cdot j}$

Continuous Bag of Word

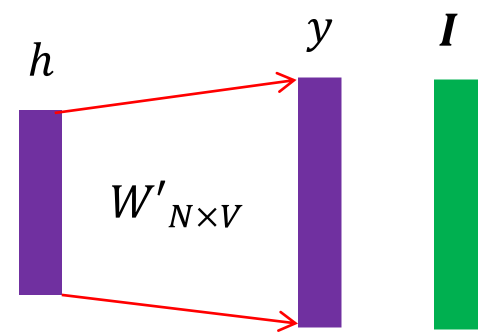
- Let's take gradient of the loss w.r.t. W_{ij}
  - $\frac{\partial loss}{\partial W_{ij}} = \Sigma_\mathrm{k} \frac{\partial \left(log \Sigma_w e^{u_w} - u_{w_m} \right)}{\partial u_k} \frac{\partial u_k}{\partial h_j} \frac{\partial h_j}{\partial W_{ij}}$
  - $= \Sigma_\mathrm{k} (y_k - \boldsymbol{I}[w_m = k]) W'_{jk} \boldsymbol{x}_i$
  - $= (y - \boldsymbol{I}) W'_{j\cdot} \boldsymbol{x}_i$, $\boldsymbol{I}$ is one hot



- So, $loss$ grad w.r.t. one input vector
  - $\frac{\partial loss}{\partial W_{i\cdot}} = (y - \boldsymbol{I}) W' \boldsymbol{x}_i$, this is for every $x_i \neq 0$

Continuous Bag of Word

- Update formula for both word vector, w.r.t. one training example (C, w_m)
  - $\frac{\partial loss}{\partial v'_{w_j}} = \left(y_{w_j} - \boldsymbol{I}[w_m = w_j]\right) h$
  - $\frac{\partial loss}{\partial v_{w_j}} = (y - \boldsymbol{I}) W' \boldsymbol{x}_j$

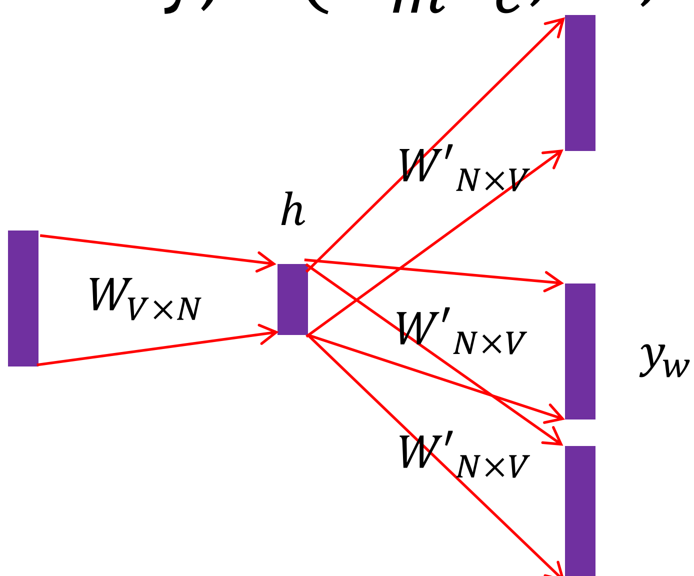


- Gradient descent
  - $v'_{w_j}{}^{new} = v'_{w_j}{}^{old} - \eta \frac{\partial loss}{\partial v'_{w_j}}$
  - $v_{w_j}^{new} = v_{w_j}^{old} - \eta \frac{\partial loss}{\partial v_{w_j}}$

$$\begin{bmatrix} | & | & & | \\ v'_1 & v'_2 & \cdots & v'_5 \\ | & | & & | \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.15 \\ 0.4 \\ 0.05 \\ 0.3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Skip-gram

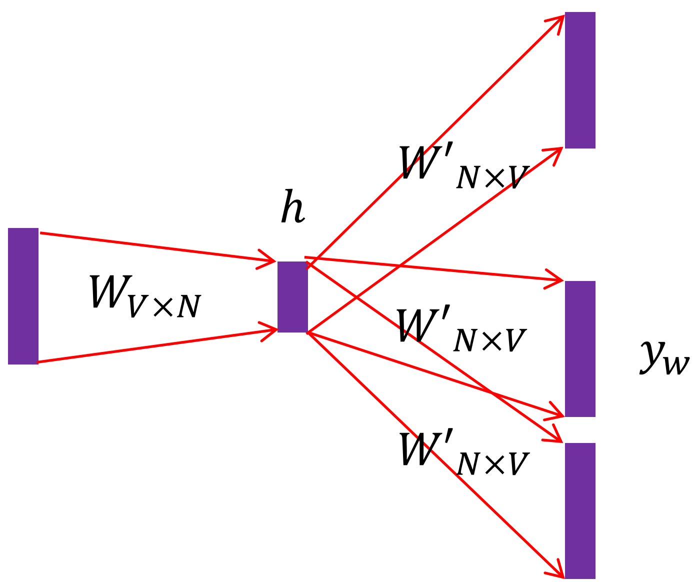
- Given supervision signal, middle word w_m and context words around C, these are pairs $(w_m, C)^i$
- Again we use FFNN to model this conditional probability, $P(w_{m-c}, \dots, w_{m+c} \mid w_m)$



- $h = W^T w_m$
- $y = \text{softmax}({W'}^T h)$
- $loss = -\Sigma_{w\in C} log y_w$, to be minimized

Skip-gram

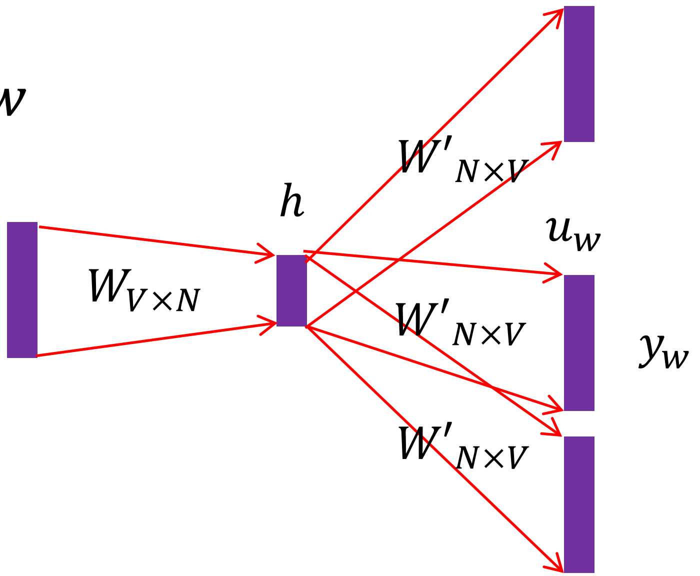
- Cost function – Maximum Likelihood
  - $Cost = -\Sigma_{(w_m,C)^i} loss^i$
  - $= -\Sigma_{(w_m,C)^i} \log \Pi_{w \in C^i} P(w|w_m^i)$



- $h = W^T w_m$
- $y = \text{softmax}(W'^T h)$
- $loss = -\Sigma_{w \in C} log y_w$, to be minimized

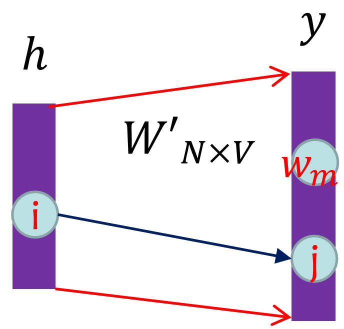
Skip-gram

- Some Notation convenience
  - v_{w_m} is the w_m row of W, input embedding of word w_m
  - v'_{w_m} is the w_m column of W', output embedding of word w_m
  - Pre-activation of output is u_w



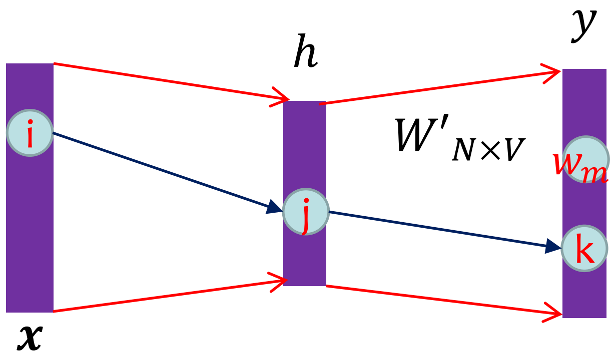
Skip-gram

- The gradient w.r.t. W'_{ij}
  - $loss = -\Sigma_{w \in C} log \frac{e^{u_w}}{\Sigma_{w' \in V} e^{u_{w'}}}$
  - $\frac{\partial loss}{\partial W'_{ij}} = -\Sigma_{w \in C} \frac{\partial (u_w - \log \Sigma)}{\partial W'_{ij}}$
  - $= -\Sigma_{w \in C} \frac{\partial u_w}{\partial W'_{ij}} - y_j . \frac{\partial u_j}{\partial W'_{ij}}$
  - $= -\Sigma_{w \in C} \frac{\partial u_j}{\partial W'_{ij}} \left(\frac{\partial u_w}{\partial u_j} - y_j \right)$
  - $= \Sigma_{w \in C} h_i (y_j - I(w = j))$



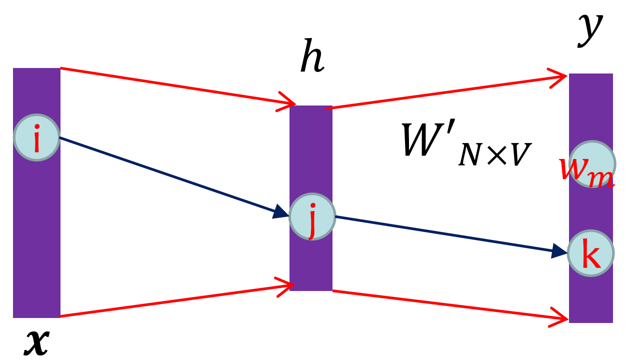
Skip-gram

- The gradient w.r.t. W_{ij}
  - $\frac{\partial loss}{\partial W_{ij}} = -\Sigma_{w\in C}\left(\frac{\partial u_w}{\partial h_j} - \frac{\partial \Sigma}{\partial h_j}\right)\frac{\partial h_j}{\partial W_{ij}}$
  - $= \Sigma_{w\in C}\left(W'_{jw} - \Sigma_t \frac{\partial \Sigma}{\partial u_t}\frac{\partial u_t}{\partial h_j}\right) x_i$
  - $= \Sigma_{w\in C}\left(W'_{jw} - \Sigma_t . \frac{e^{u_t}}{\Sigma} W_{jt}\right) x_i$
  - $= \Sigma_{w\in C}\left(W'_{jw} - \Sigma_t . y_t W_{jt}\right) x_i$



Computation Cost

- Softmax is costly for forward computation
  - $u_j = hW'_{\cdot j}$ for every j in V
  - Σe^{u_j} on $|V|$ elements
- Thus, Gradient update should loop over W'
  - h is dense
  - $\boldsymbol{x}$ is sparse

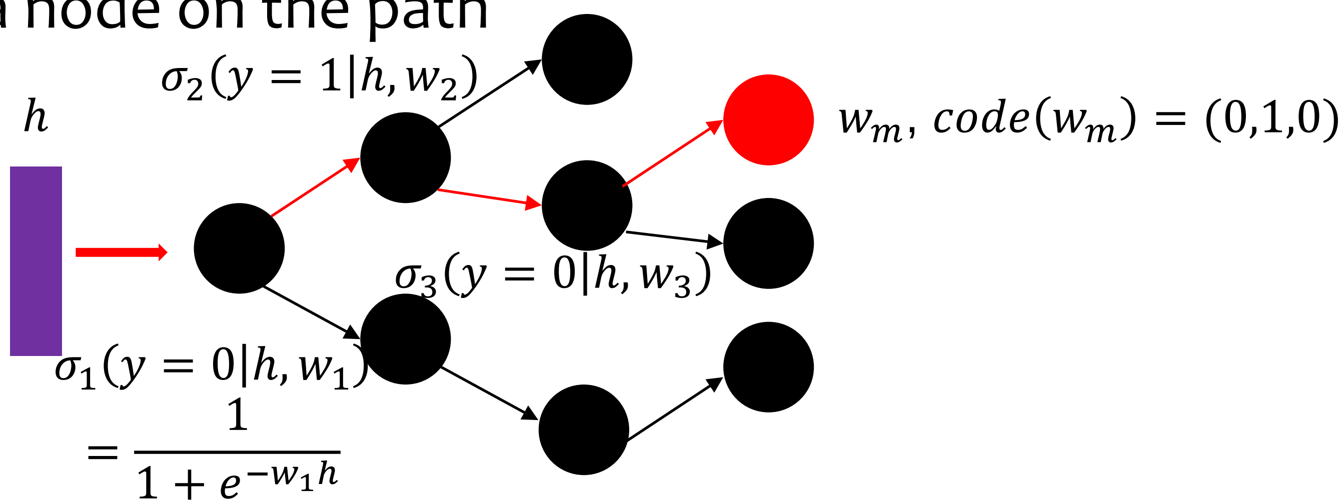


Output Layer Modification

- Hierarchical Softmax (accurate)
  - Decompose softmax as a binary tree
  - We can do iteratively binary search, so that we can use sigmoid
- Negative Sampling (approximate)
  - Monte Carlo

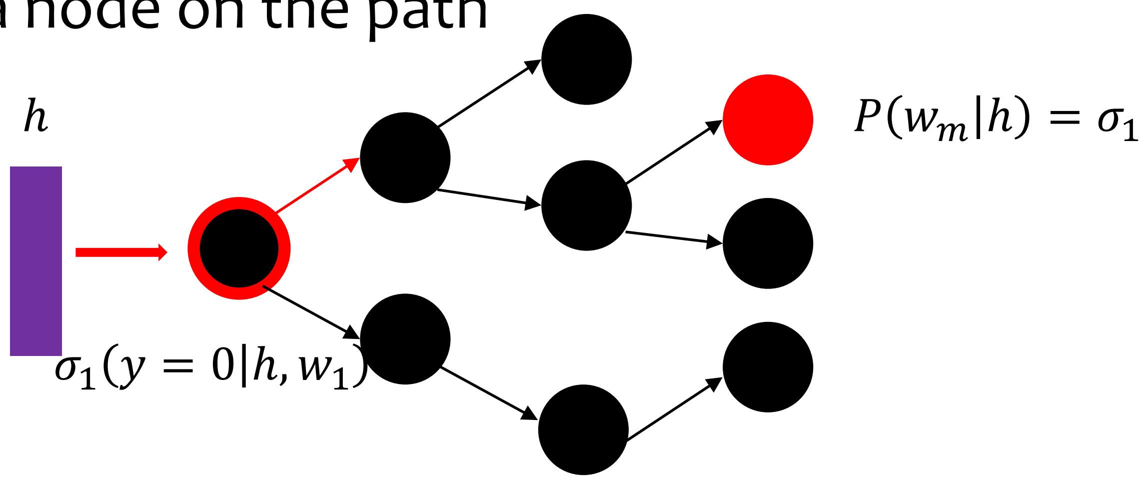
Hierarchical Softmax

- Output has a hierarchy
  - Binary tree build on Huffman code of each word in corpus
- $P(w_m|h) = \Pi_{n \in Path(w_m)} \sigma_n(code(w_m)_n | h, w_n)$
  - $code(w_m)$ is the Huffman code of the word
  - n is a node on the path



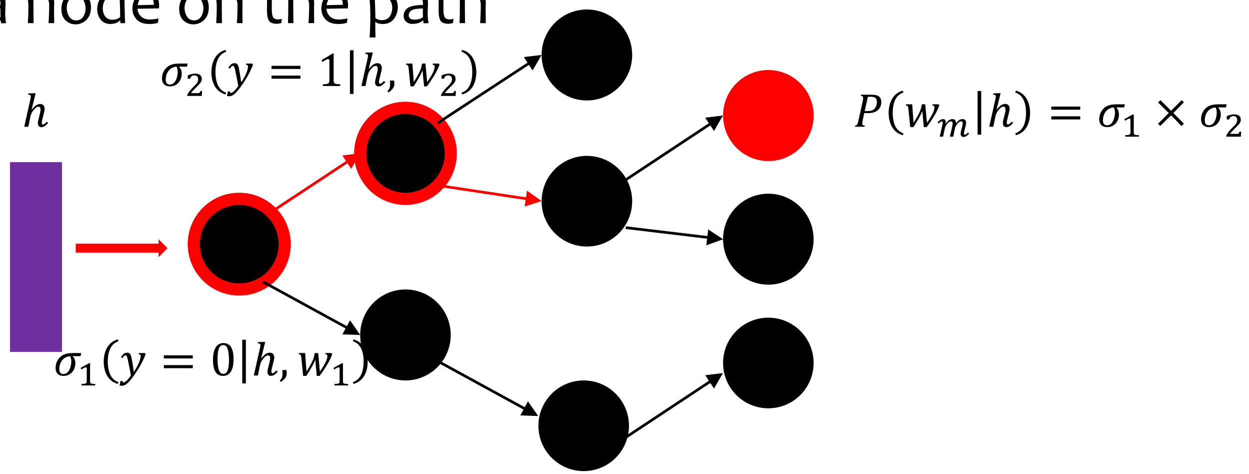
Hierarchical Softmax

- Output has a hierarchy
  - Binary tree build on Huffman code of each word in corpus
- $P(w_m|h) = \Pi_{n \in Path(w_m)} \sigma_n(code(w_m)_n | h, w_n)$
  - $code(w_m)$ is the Huffman code of the word
  - n is a node on the path



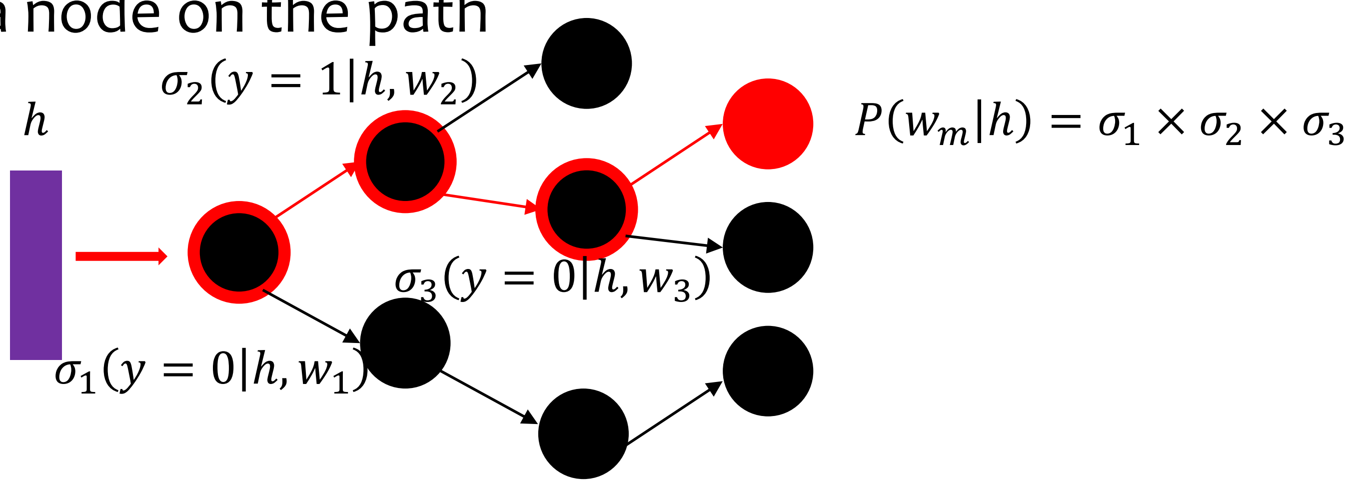
Hierarchical Softmax

- Output has a hierarchy
  - Binary tree build on Huffman code of each word in corpus
- $P(w_m|h) = \Pi_{n \in Path(w_m)} \sigma_n(code(w_m)_n|h, w_n)$
  - $code(w_m)$ is the Huffman code of the word
  - n is a node on the path



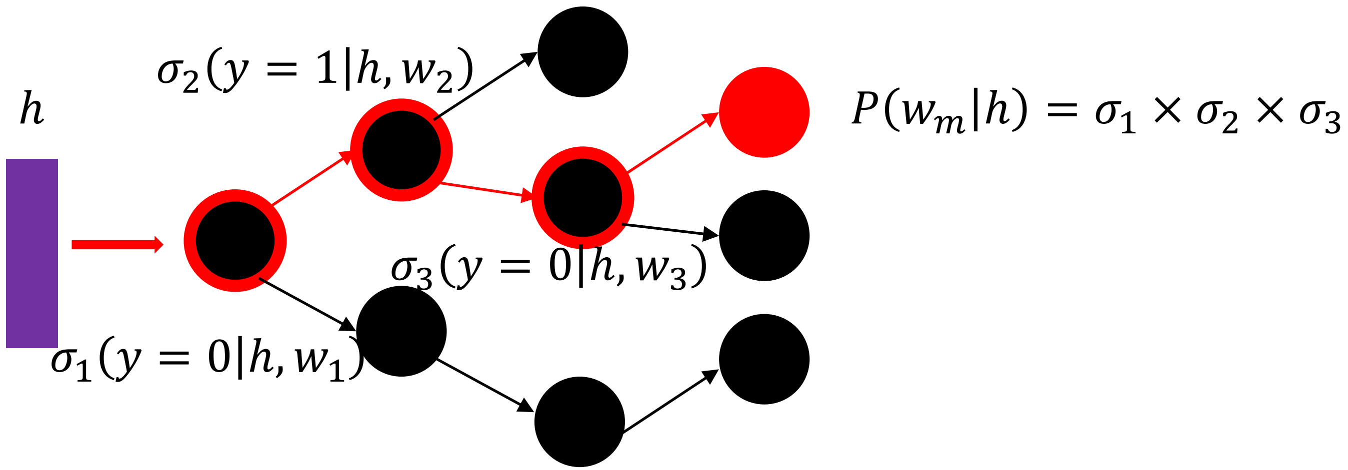
Hierarchical Softmax

- Output has a hierarchy
  - Binary tree build on Huffman code of each word in corpus
- $P(w_m|h) = \Pi_{n \in Path(w_m)} \sigma_n(code(w_m)_n|h, w_n)$
  - $code(w_m)$ is the Huffman code of the word
  - n is a node on the path



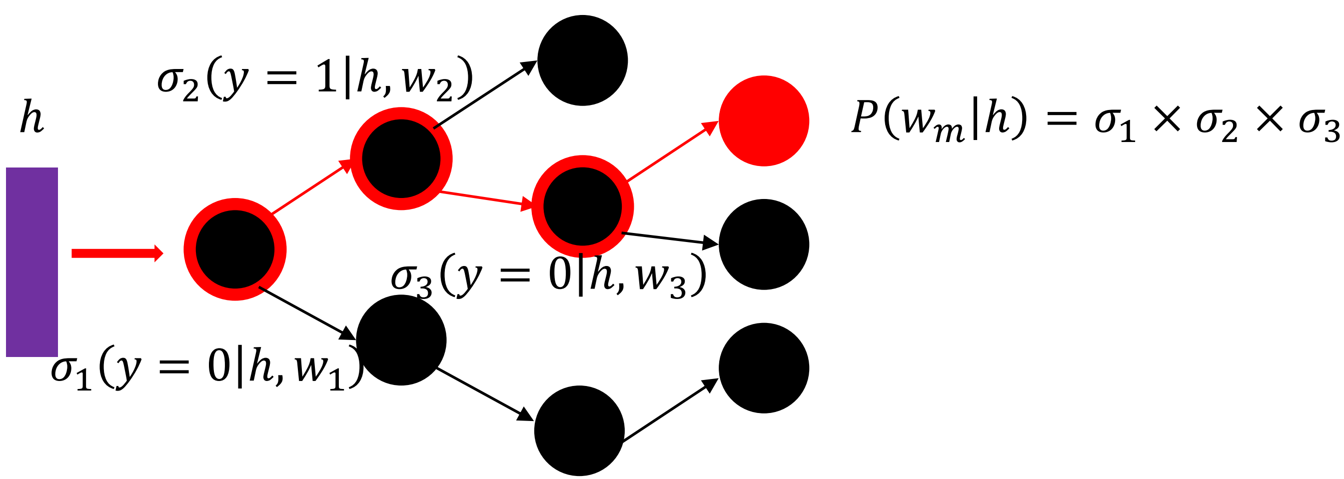
Hierarchical Softmax

- Computation cost
  - Forward compute: from $h|V|$ to $hlog|V|$
  - Backward, gradient update: from $h|V|$ to $hlog|V|$
  - Parameter number?



Hierarchical Softmax

- Accurate probability compute
  - Why? No approximate equal
  - Softmax is just one way to achieve picking one item from a class of items, here word from vocabulary, where items are flat



Negative Sampling

- Let's totally forget about softmax!
  - Just for a while: orz
- Think about an intuitive scenario
  - You are learning portrait painting, and learn to distinguish different noses given a face
- Adversarial examples

Negative Sampling

- Follow the intuition, for each training instance (C, w)
  - We have the positive instance w, the word we predict
  - Still want some negatives, to discriminate upon, so we design a simple distribution, e.g. unigram to sample a group of fake predictions
- Our objective function becomes, still maximize:
  - $P\left(D = 1 \middle| h, v'_{w_m}\right) + \Sigma_{w_i \sim p_{uni}} P(D = 0 | h, v'_{w_i})$

Besides Negative Sampling

- Importance Sampling
- Adaptive Importance Sampling
- Target Sampling
- Noise Contrastive Estimation
- Negative Sampling
- …

Sampling

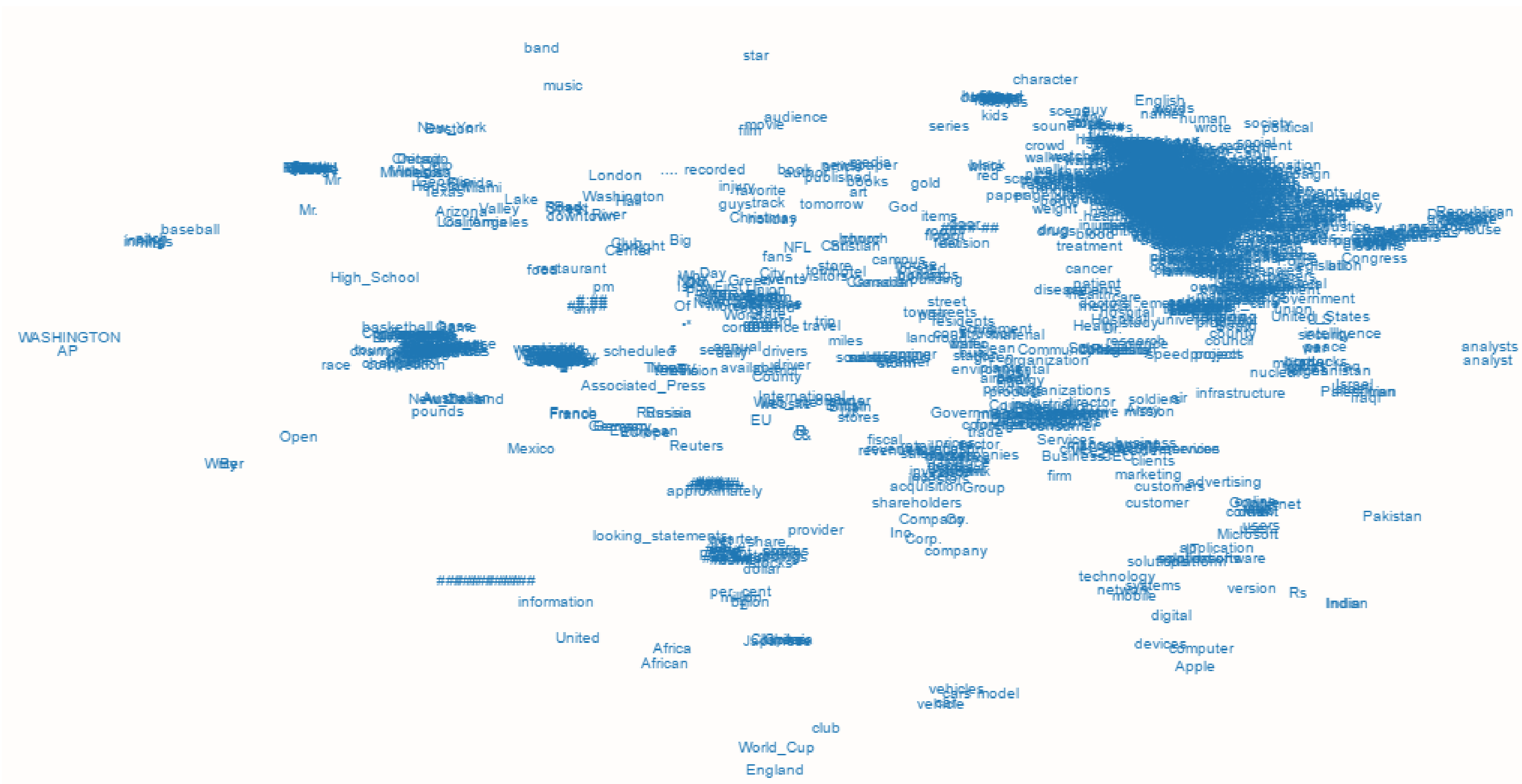
- Monte Carlo
  - We have a complex distribution $P(X)$
  - We want to compute expectation w.r.t. $f(X)$
    - $E = \int_x dx \cdot f(x)P(x)$
  - Intractable for accurate computation
  - Sample N points from $P(X)$, $\{x_1, x_2, \dots, x\_N\}$
  - Calculate empirical mean:
    - $\hat{E} = \frac{1}{N}\Sigma_i f(x_i)$

Sampling

- More formally, sampling methods is doing an approximation of the softmax
- $logP(w_m|C) = log\frac{e^{hv_{w_m}}}{\Sigma_{v_i}e^{hv_i}} = hv_w - log\Sigma e^{hv_i}$
- We made gradient of the above:
  - $\nabla_\theta hv_w - \nabla_\theta log\Sigma e^{hv_i} = \nabla_\theta hv_w - \frac{1}{\Sigma e^{hv_i}}\Sigma\nabla_\theta e^{hv_i}$
  - $= \nabla_\theta hv_w - \Sigma\frac{1}{\Sigma e^{hv_i}}e^{hv_j}\nabla_\theta hv_j = \nabla_\theta hv_w - \Sigma\frac{e^{hv_j}}{\Sigma e^{hv_i}}\nabla_\theta hv_j$
  - $= \nabla_\theta hv_w - \Sigma P(w_j|C)\nabla_\theta hv_j = \nabla_\theta hv_w - \boxed{\mathrm{E}[\nabla_\theta hv_j]}$

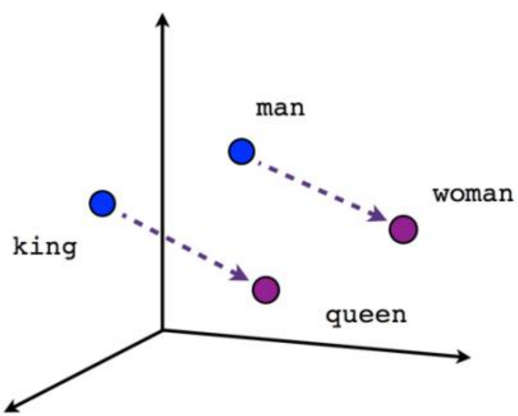
Some Insights from Word2vec

- Word regularities
  - Similarities and neighborhood

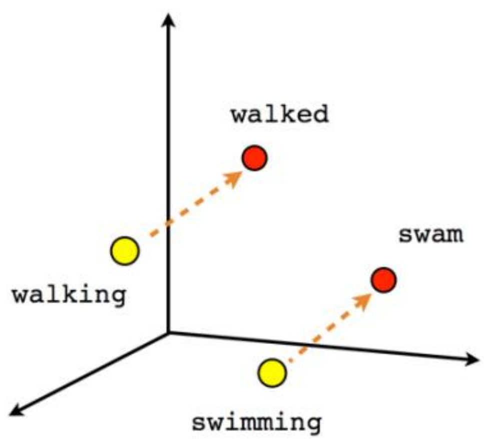


Some Insights from Word2vec

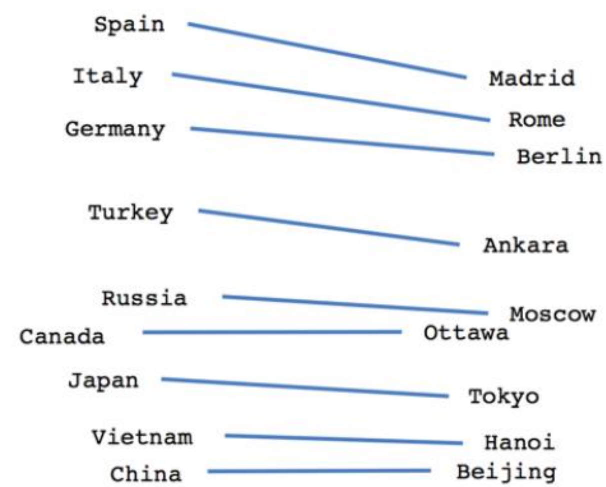
- Word regularities
  - Linear regularities are embodied



Male-Female



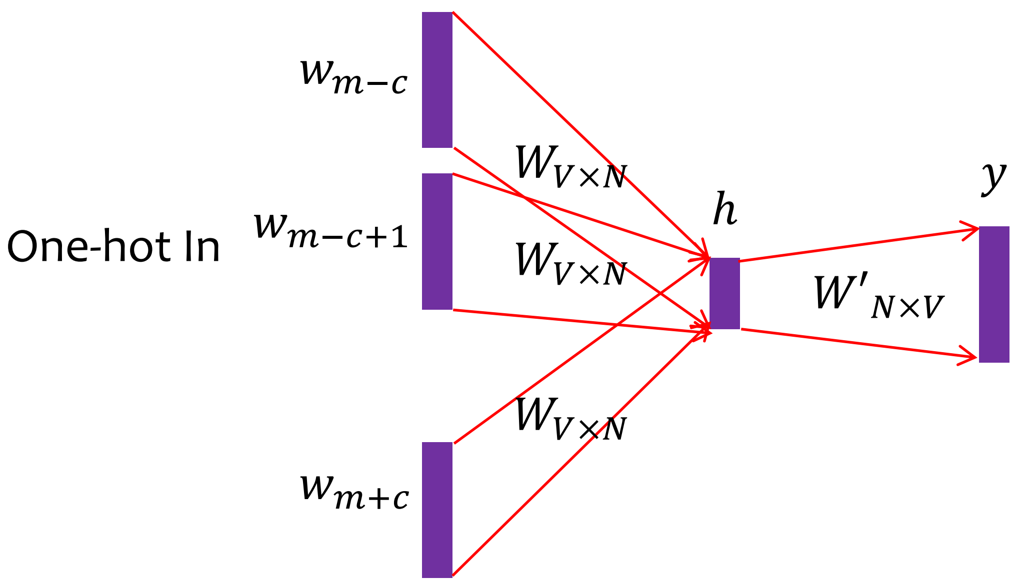
Verb tense



Country-Capital

Bonus

- Why we could use weight matrix as our real-valued word feature matrix?



Outline

PART II

- Word Embedding – Basic Models
  - Word2vec
  - GloVe
- Representation Learning of Lexical Meaning
  - Sentiment Embedding
  - Topical Embedding
  - Discourse Relation-aware Embedding

GloVe

Embedding and NLP

Is Deep Learning Really necessary for Word Embeddings?