Neural Nets, Deep-Learning and applications (NLP)

A. Allauzen

Université Paris-Sud / LIMSI-CNRS

08/01/2016

Outline

1. Advanced Architectures with NLP applications
   - n-gram language model
   - Sequence representation
   - Character based model for sequence tagging

2. Conclusion

Outline

A case study : language model

The n-gram model

Applications

Automatic Speech Recognition, Machine Translation, OCR, ...

The goal

Estimate the (non-zero) probability of a word sequence for a given vocabulary

n-gram assumption

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i | w_{i-n+1}^{i-1}), \quad \forall i, w_i \in \mathcal{V}$$

Discrete n-gram model (conventional)

A word given its context

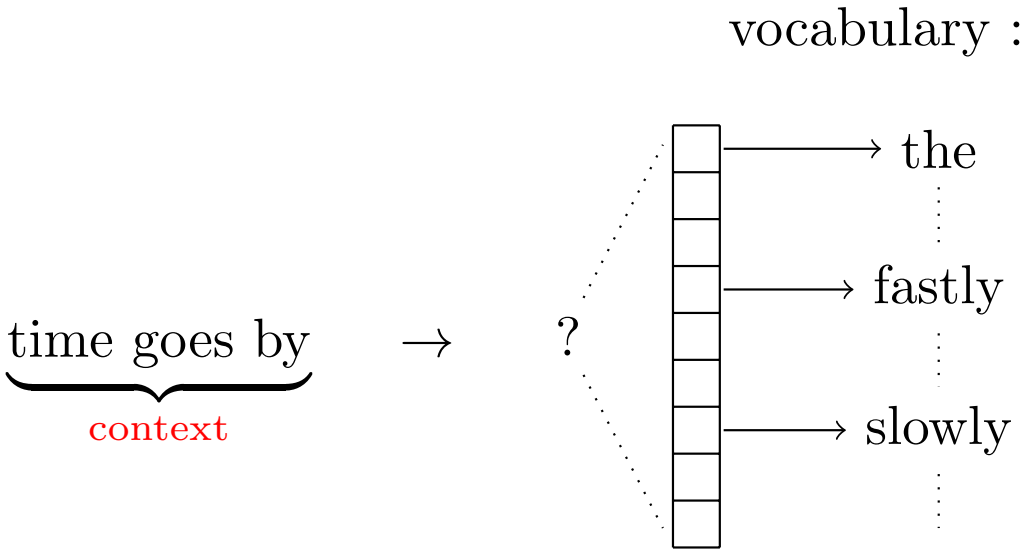
$n = 4$:

$\underbrace{\text{time goes by}}_{\text{context}} \quad \rightarrow \quad ?$

Discrete n-gram model (conventional)

A word given its context

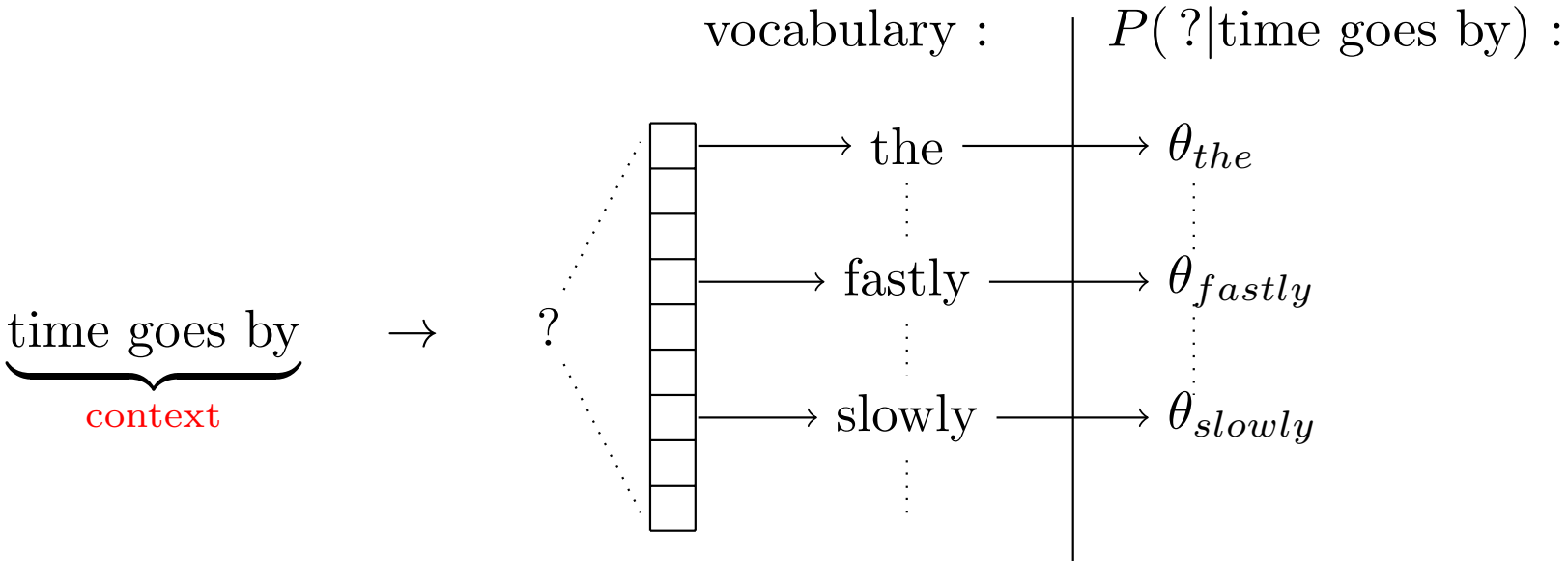
$n = 4$:



Discrete n-gram model (conventional)

A word given its context

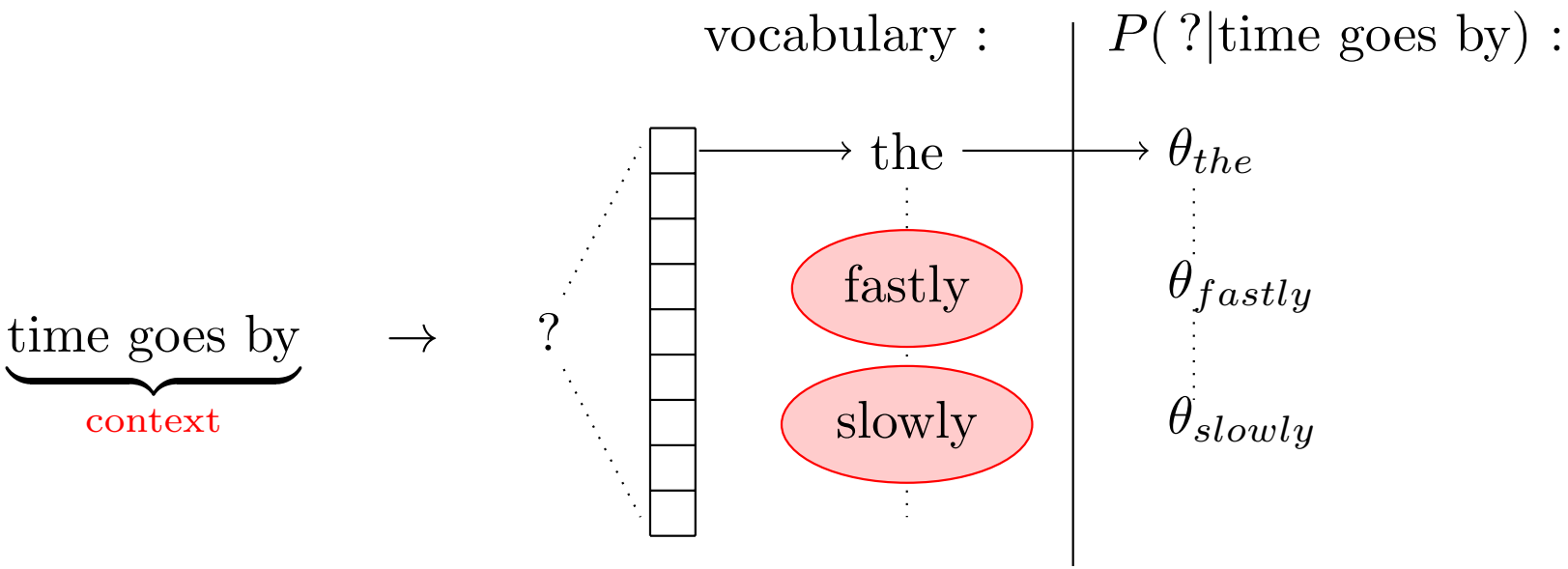
$n = 4$:



Discrete n-gram model (conventional)

A word given its context

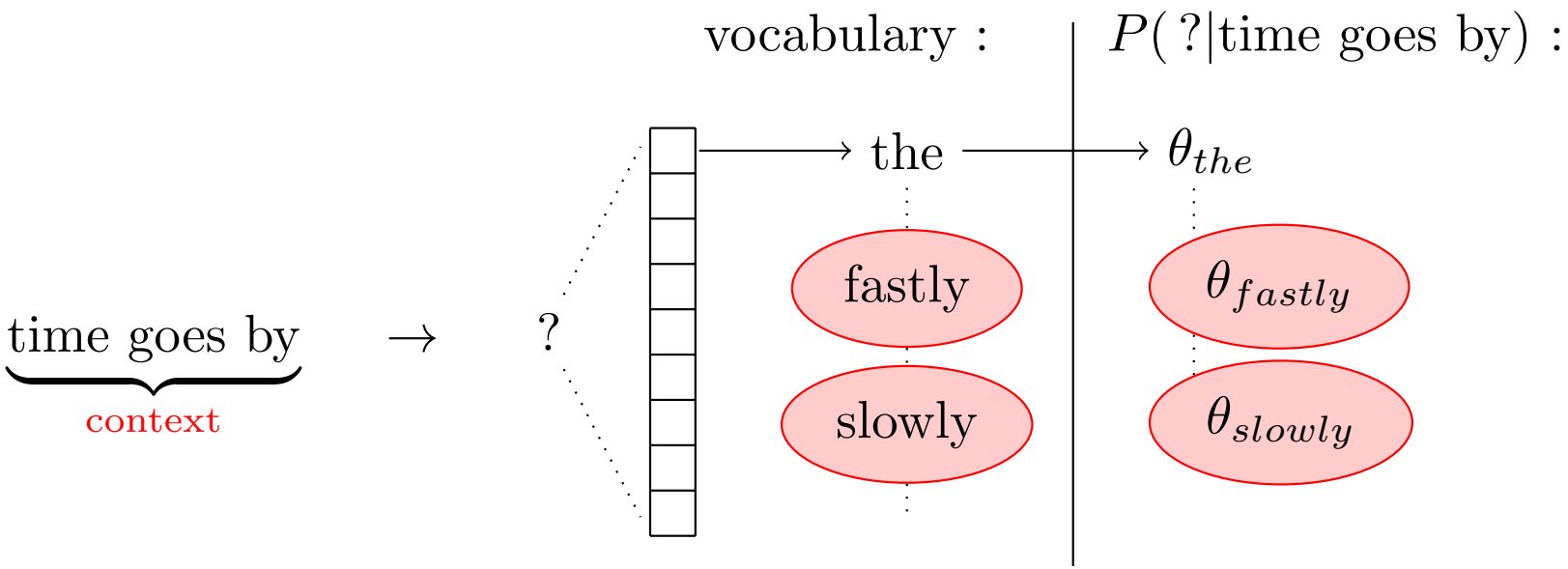
$n = 4$:



Discrete n-gram model (conventional)

A word given its context

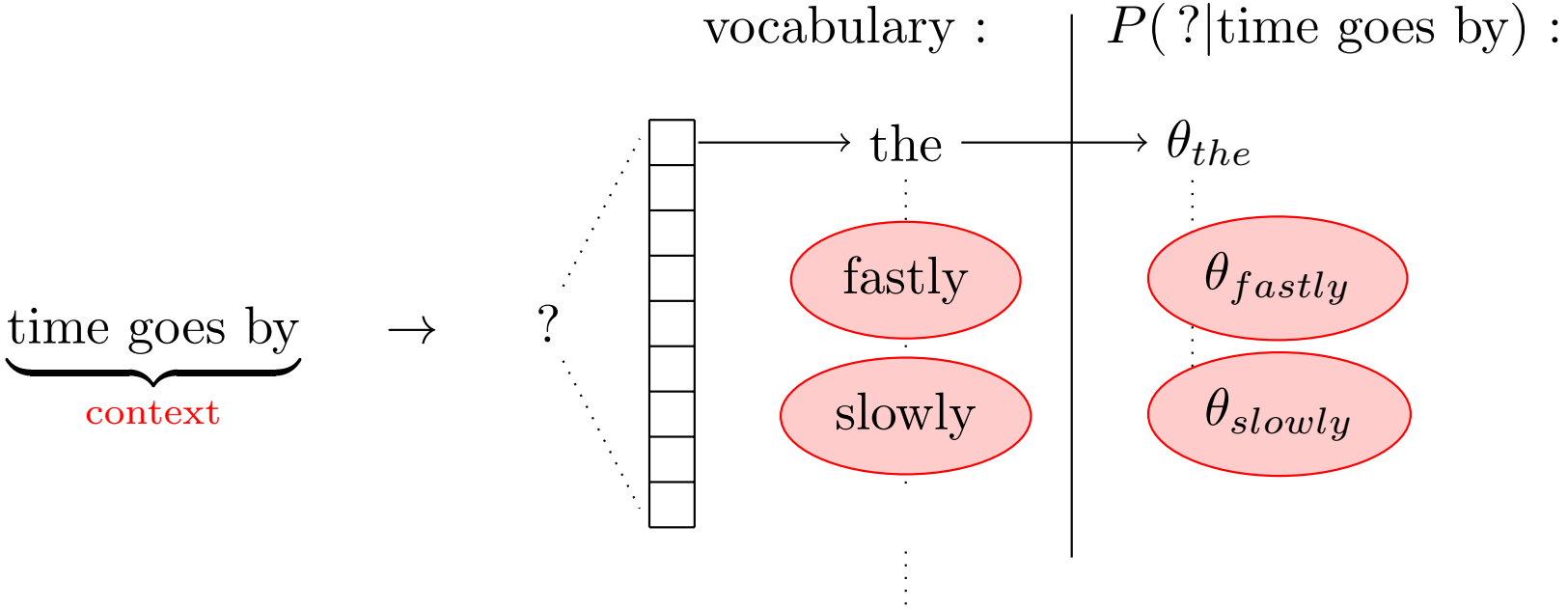
$n = 4$:



Discrete n-gram model (conventional)

A word given its context

$n = 4$:



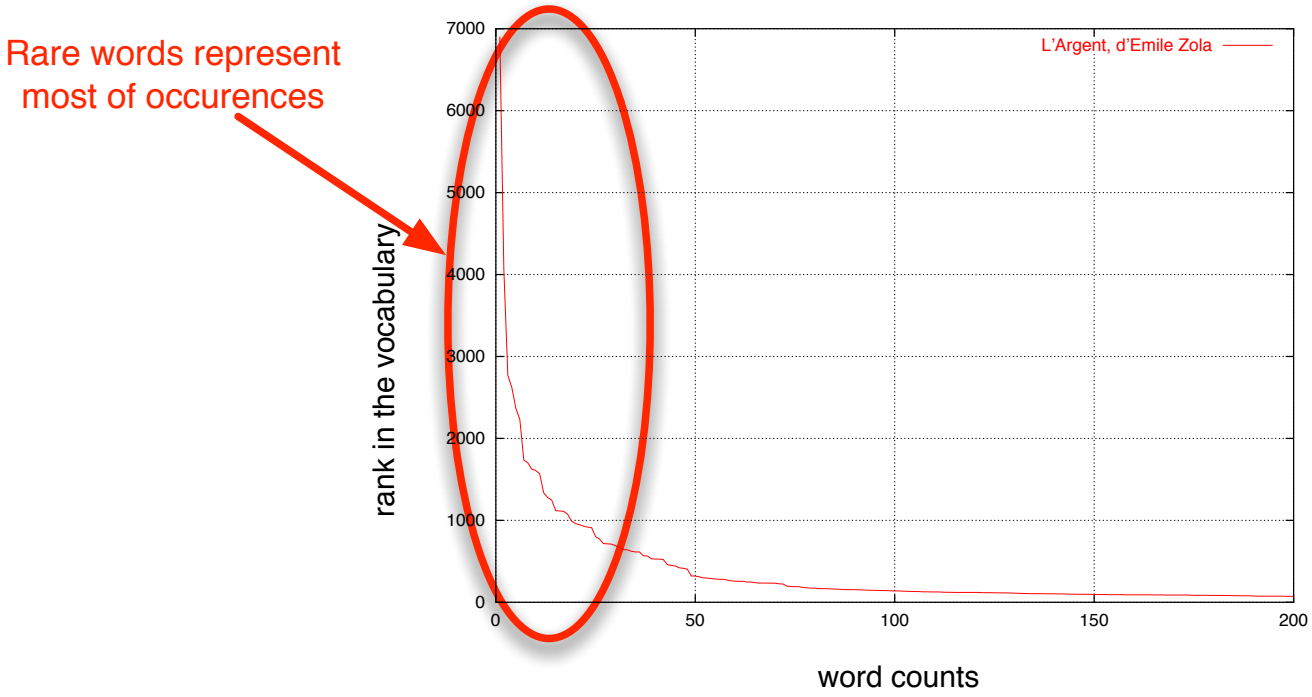
- For each context, a multinomial distribution
- A word in its context = one parameter to learn
- No parameter tying within words

$\Rightarrow$ No knowledge sharing

Practical issues

Data sparsity

- The Zipf law



Practical issues

Data sparsity

- The Zipf law
- Most of the n-grams never occur

$\Rightarrow$ Smoothing methods (Chen and Goodman1998)

Practical issues

Data sparsity

- The Zipf law
- Most of the n-grams never occur

$\Rightarrow$ Smoothing methods (Chen and Goodman1998)

The lack of generalization

- Similarities between words and contexts are neglected

$\Rightarrow$ Still increasing the amount of data

Practical issues

Data sparsity

- The Zipf law
- Most of the n-grams never occur

$\Rightarrow$ Smoothing methods (Chen and Goodman1998)

The lack of generalization

- Similarities between words and contexts are neglected

$\Rightarrow$ Still increasing the amount of data

Restricted context

- The number of parameters grows drastically with n

$\Rightarrow$ The curse of dimensionality.

$\Rightarrow$ In practice $n \leq 5$

Illustration

With 6 billions of running words (English texts) and a vocabulary of 500 000 words :

- $500\ 000^4 \approx 6 \times 10^{22}$ possible 4-grams,
- Less than 2 billions are observed
- Most of them just once

Illustration

With 6 billions of running words (English texts) and a vocabulary of 500 000 words :

- $500\ 000^4 \approx 6 \times 10^{22}$ possible 4-grams,
- Less than 2 billions are observed
- Most of them just once

For a better generalization :

- Learn shared representations for words

$\Rightarrow$ Continuous space language models
i.e neural network language models

Estimate n-gram probabilities in a continuous space

Introduced in (Bengio and Ducharme2001; Bengio et al.2003) and applied to speech recognition and machine translation in (Schwenk and Gauvain2002).

In a nutshell

1. associate each word with a continuous feature vector
2. express the probability function of a word sequence in terms of the feature vectors of these words
3. learn simultaneously the feature vectors and the parameters of that probability function.

Estimate n-gram probabilities in a continuous space

Introduced in (Bengio and Ducharme2001; Bengio et al.2003) and applied to speech recognition and machine translation in (Schwenk and Gauvain2002).

In a nutshell

1. associate each word with a continuous feature vector
2. express the probability function of a word sequence in terms of the feature vectors of these words
3. learn simultaneously the feature vectors and the parameters of that probability function.

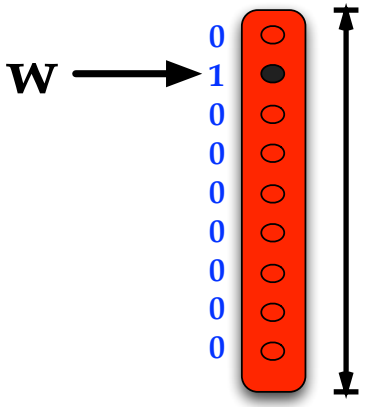
Why should it work ?

- "similar" words are expected to have a similar feature vectors
- the probability function is a smooth function of these feature values

$\Rightarrow$ a small change in the features will induce a small change in the probability

Project a word sequence in a continuous space

- The vocabulary is a neuron layer.

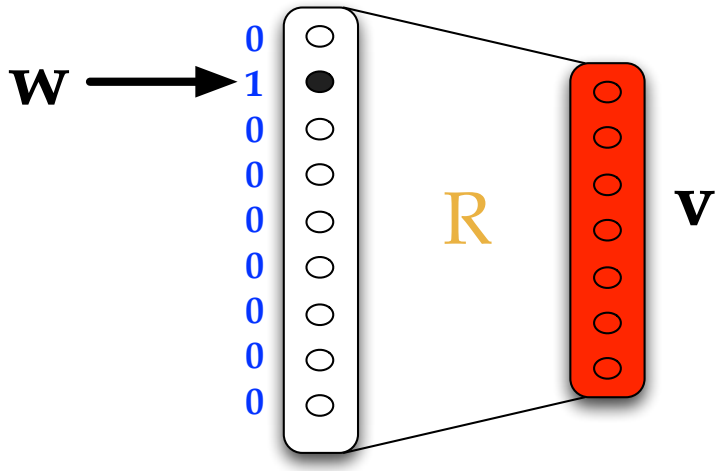


|V|: vocabulary size

- A neuron layer represents a vector of values,
- one neuron per value

Project a word sequence in a continuous space

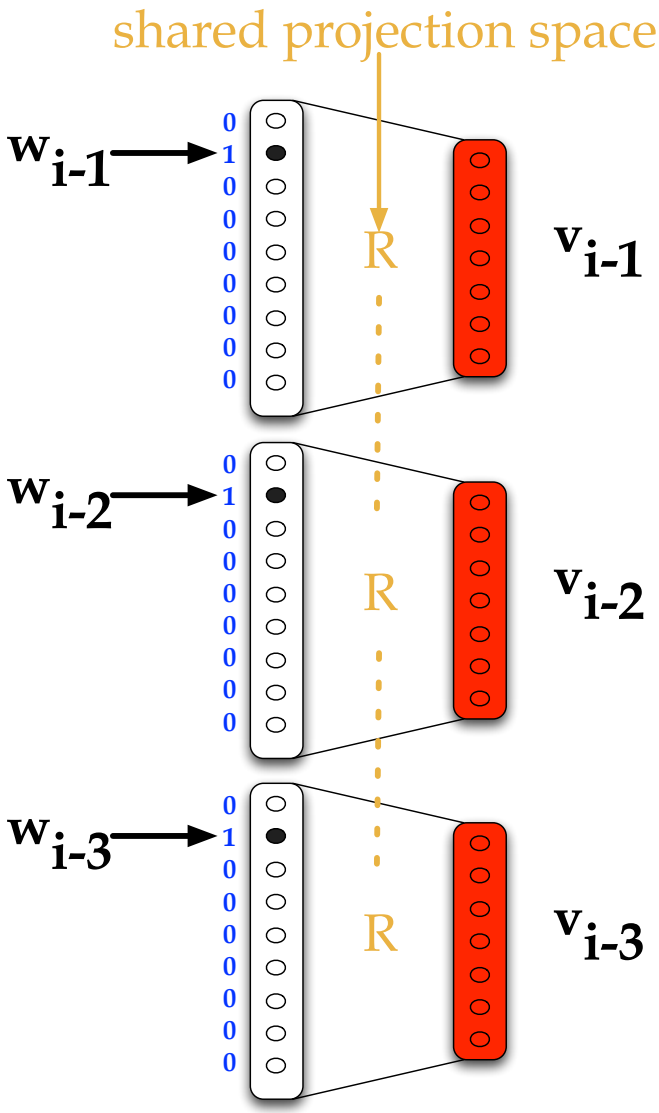
- The vocabulary is a neuron layer.
- Project the word in the continuous space : add a second layer fully connected.



- The connection between two layers is a matrix operation
- The matrix $\mathbf{R}$ contains all the connection weights
- $\mathbf{v}$ is a continuous vector

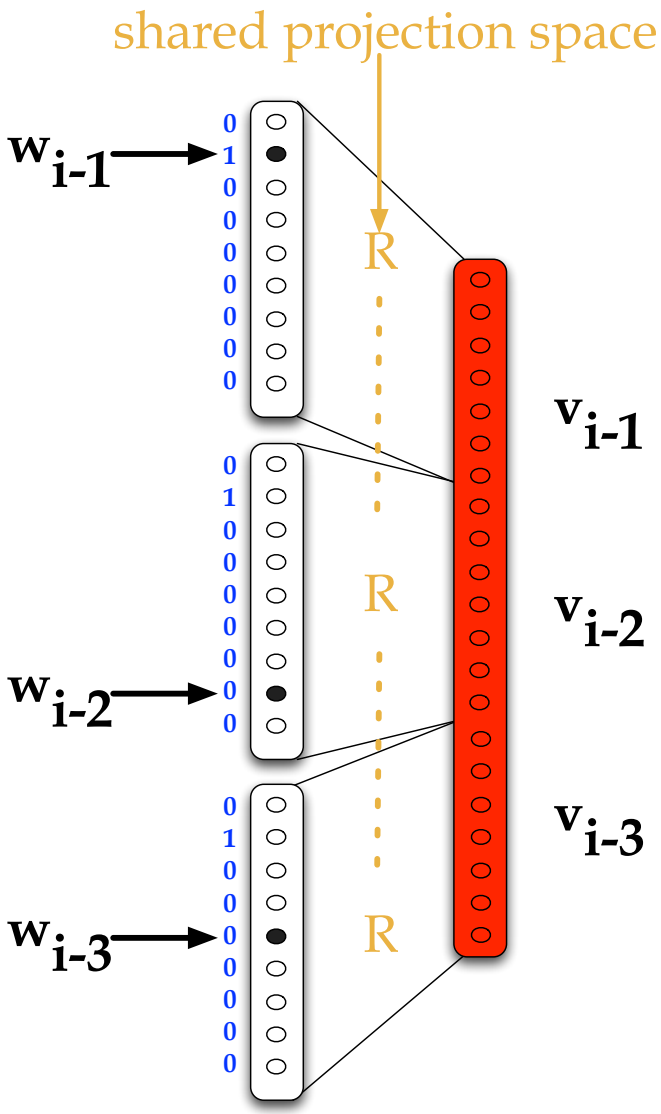
Project a word sequence in a continuous space

- The vocabulary is a neuron layer.
- Project the word in the continuous space : add a second layer fully connected.
- For a 4-gram, the history is a sequence of 3 words.



Project a word sequence in a continuous space

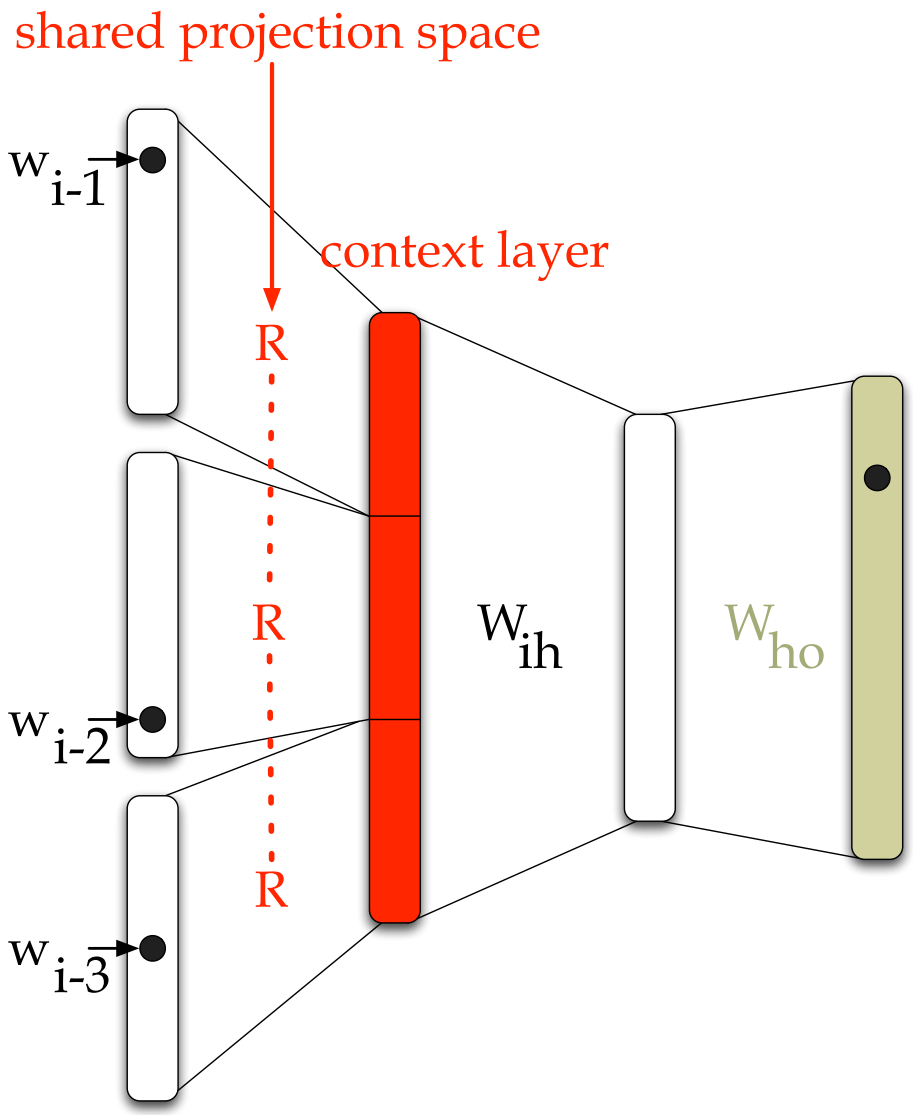
- The vocabulary is a neuron layer.
- Project the word in the continuous space : add a second layer fully connected.
- For a 4-gram, the history is a sequence of 3 words.
- Merge these three vectors to derive a single vector for the history



Estimate the n-gram probability

The program

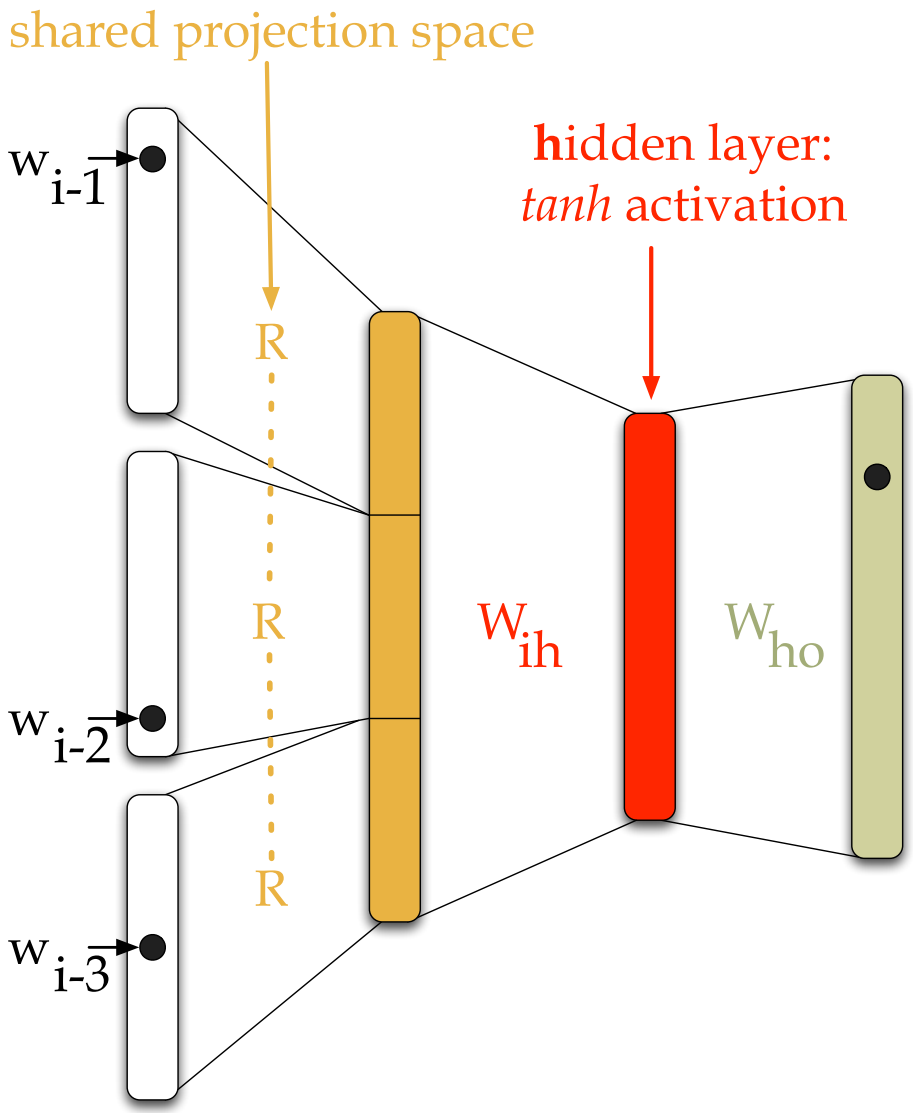
- Given the history expressed as a feature vector.



Estimate the n-gram probability

The program

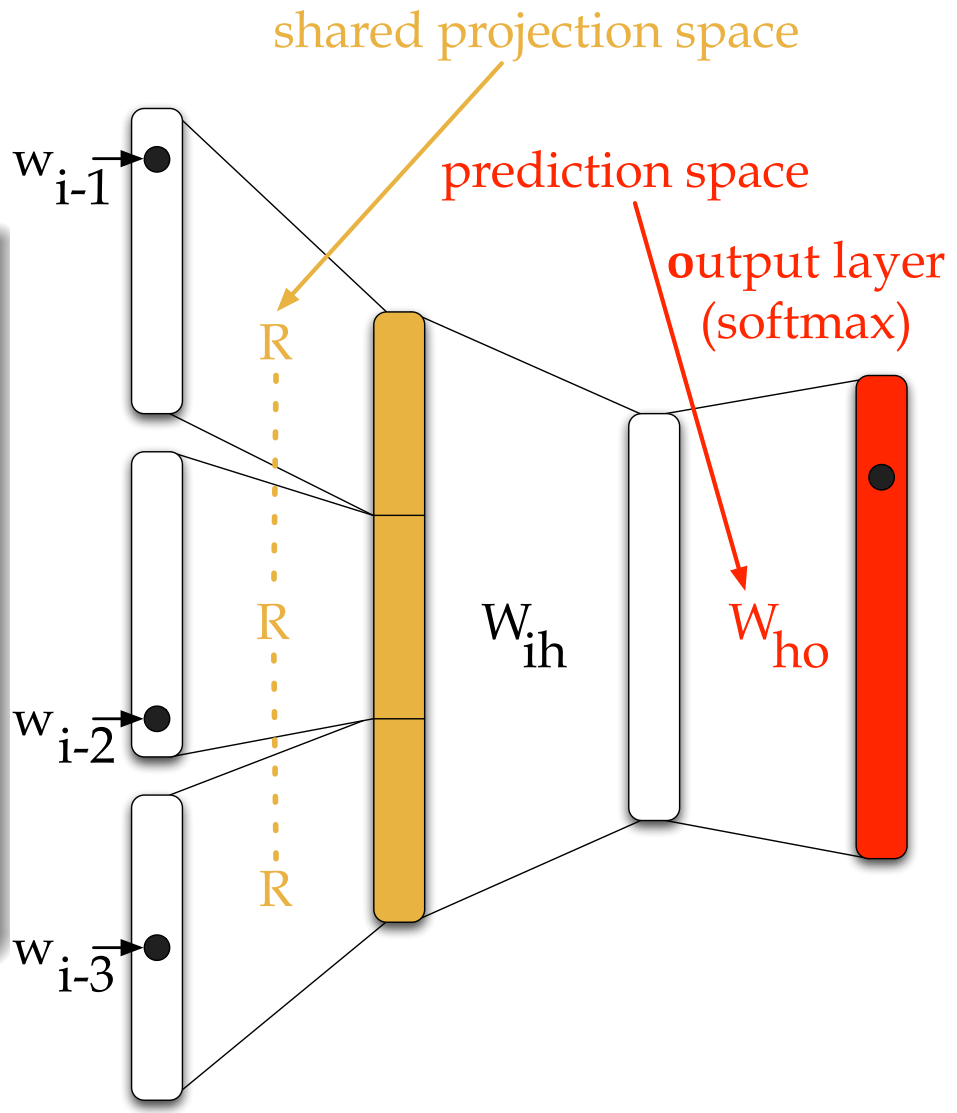
- Given the history expressed as a feature vector.
- Create a feature vector for the word to be predicted in the **prediction space**.



Estimate the n-gram probability

The program

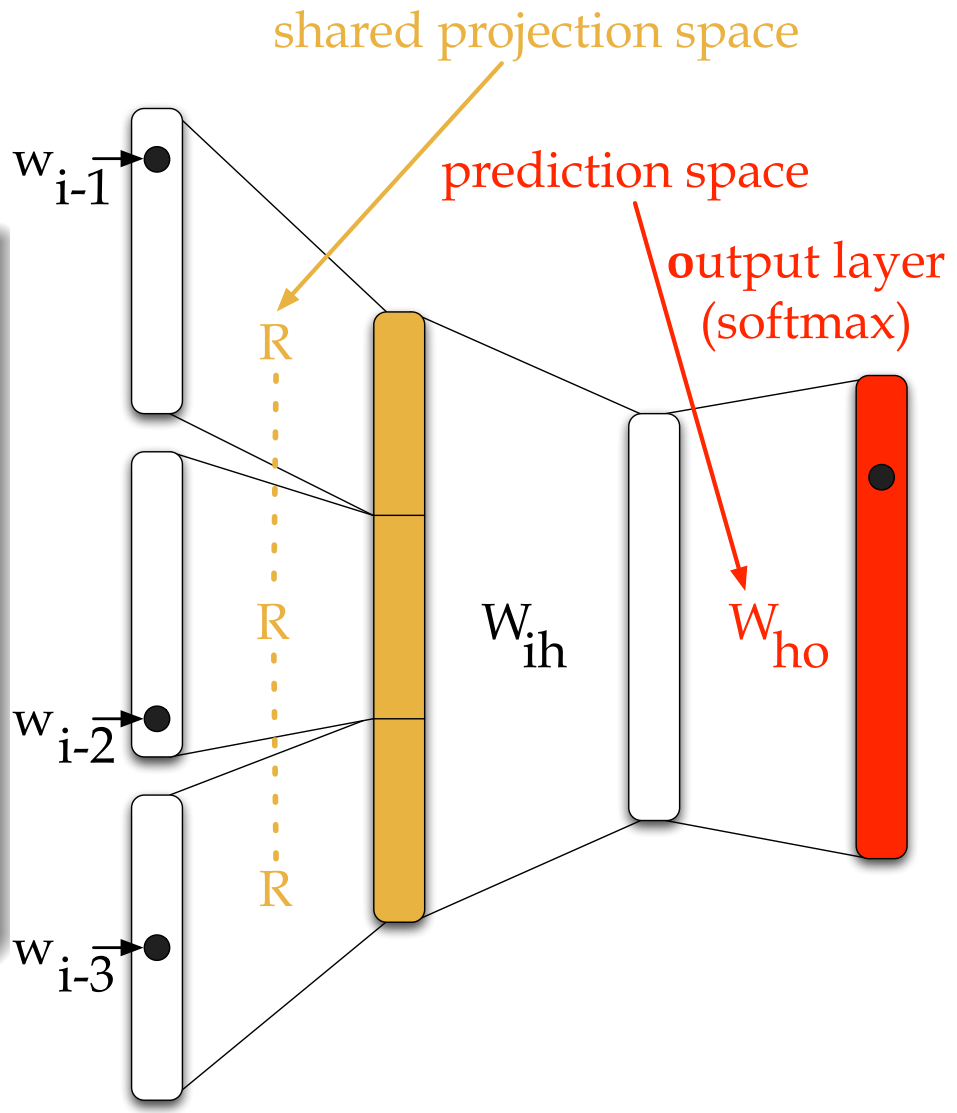
- Given the history expressed as a feature vector.
- Create a feature vector for the word to be predicted in the **prediction space**.
- Estimate probabilities for all words given the history.



Estimate the n-gram probability

The program

- Given the history expressed as a feature vector.
- Create a feature vector for the word to be predicted in the **prediction space**.
- Estimate probabilities for all words given the history.
- All the parameters must be learnt $(\mathbf{R}, \mathbf{W_{ih}}, \mathbf{W_{ho}})$.



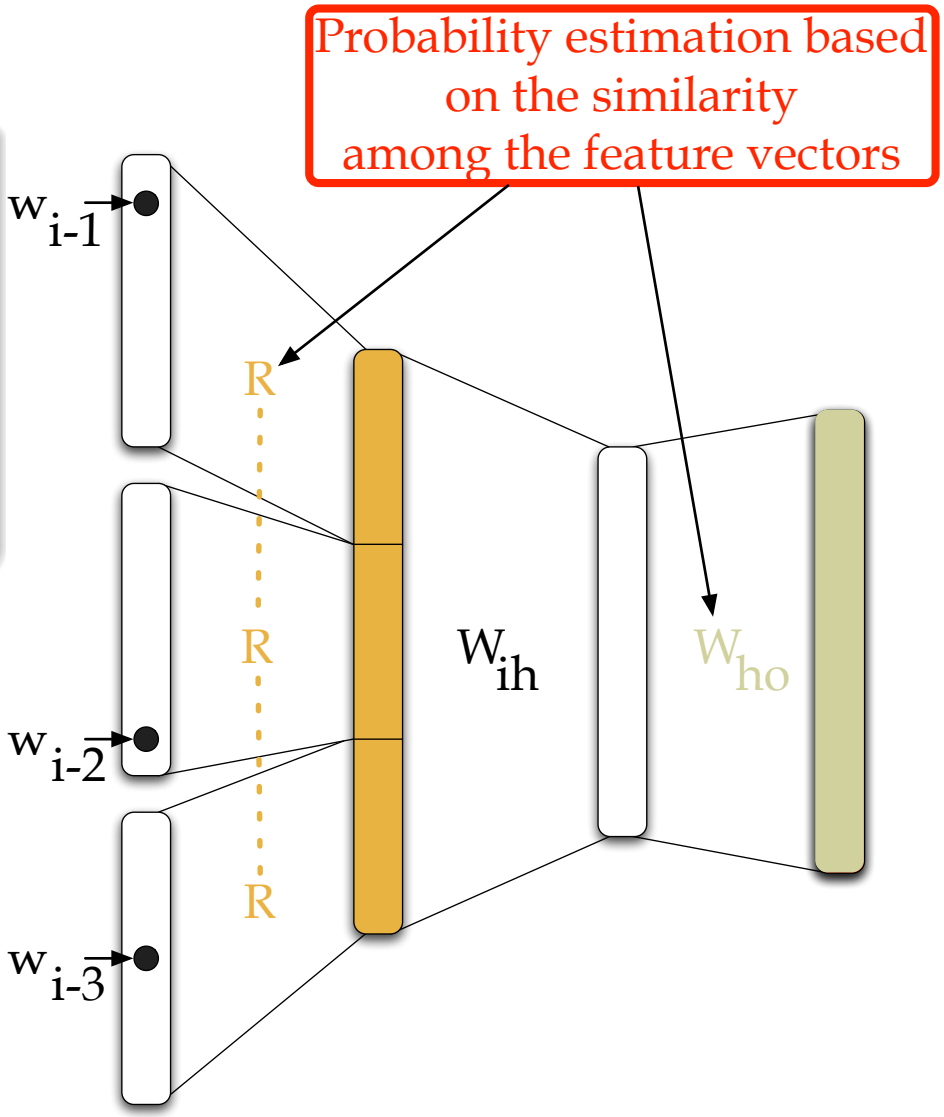
Early assessment

Key points

- The projection **in continuous spaces**

→ reduces the sparsity issues

- Learn simultaneously the projection and the prediction



Early assessment

Key points

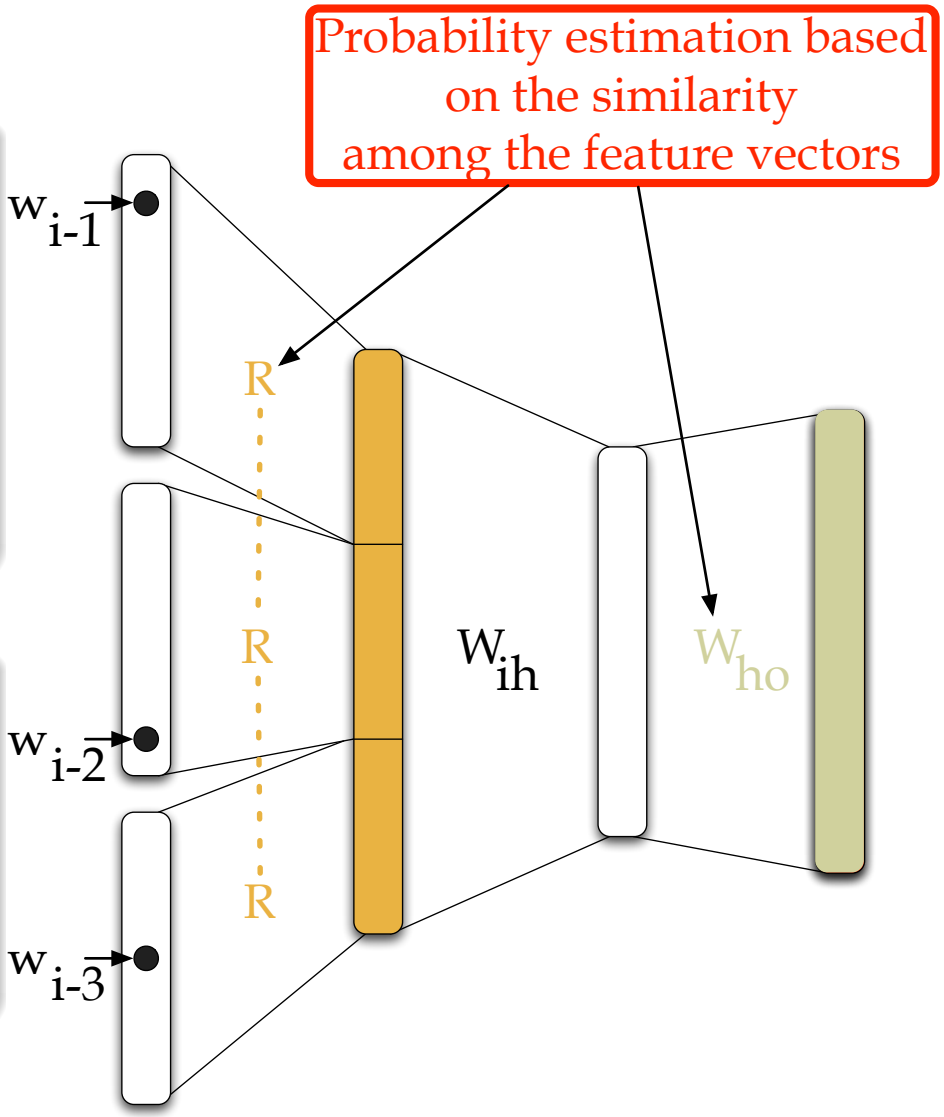
- The projection **in continuous spaces**

$\rightarrow$ reduces the sparsity issues

- Learn simultaneously the projection and the prediction

In practice

- Significant and systematic improvements
- In machine translation and speech recognition tasks



Early assessment

Key points

- The projection **in continuous spaces**

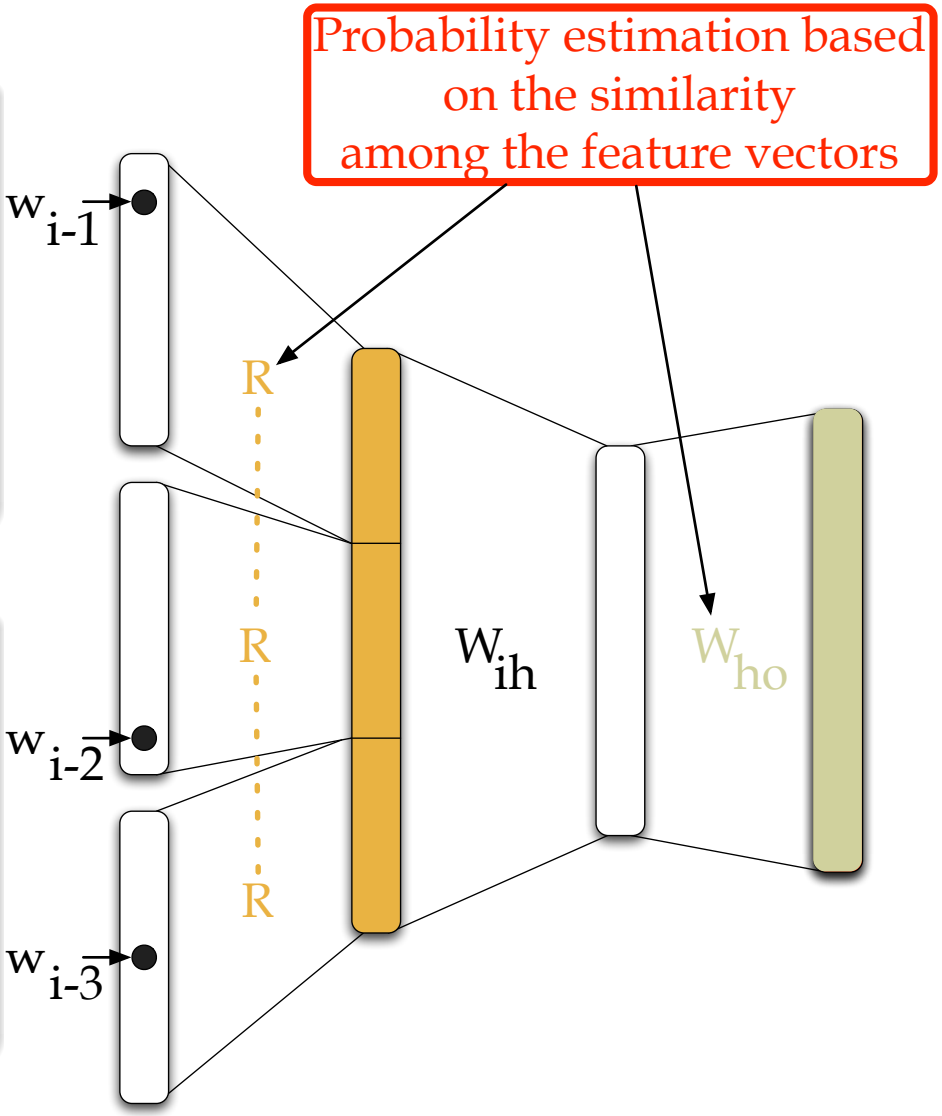
→ reduces the sparsity issues

- Learn simultaneously the projection and the prediction

In practice

- Significant and systematic improvements
- In machine translation and speech recognition tasks

☺ **Everybody should use it !**



Early assessment

Key points

- The projection **in continuous spaces**

→ reduces the sparsity issues

- Learn simultaneously the projection and the prediction

In practice

- Significant and systematic improvements
- In machine translation and speech recognition tasks

☹ **Learning and inference time**

With a small training set

Early assessment

Key points

- The projection **in continuous spaces**

→ reduces the sparsity issues

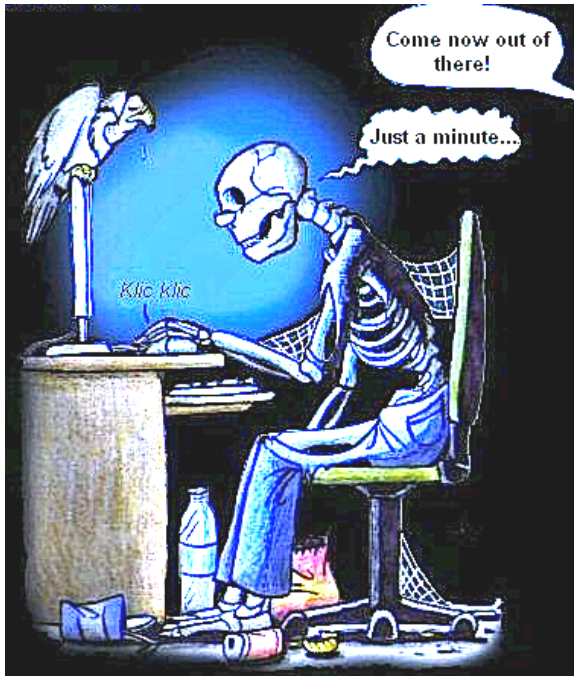
- Learn simultaneously the projection and the prediction

In practice

- Significant and systematic improvements
- In machine translation and speech recognition tasks

☹ **Learning and inference time**

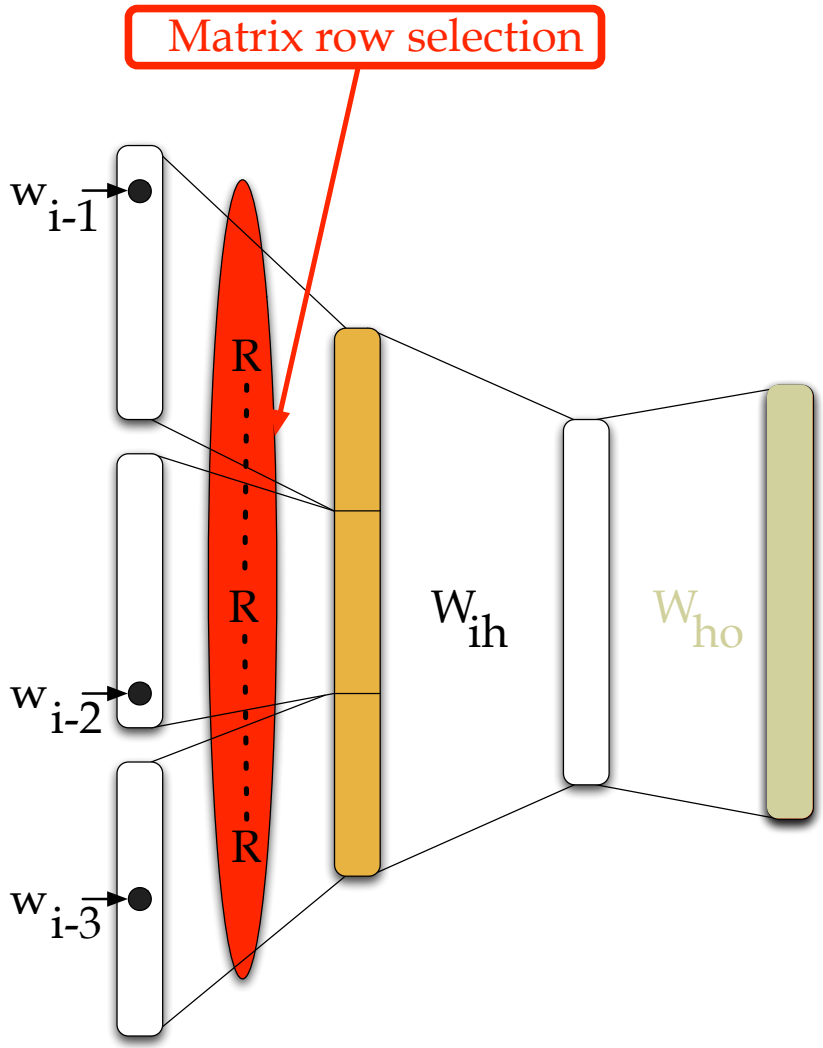
With a large training set



Why it is so long ? - Inference

Forward propagation of the history

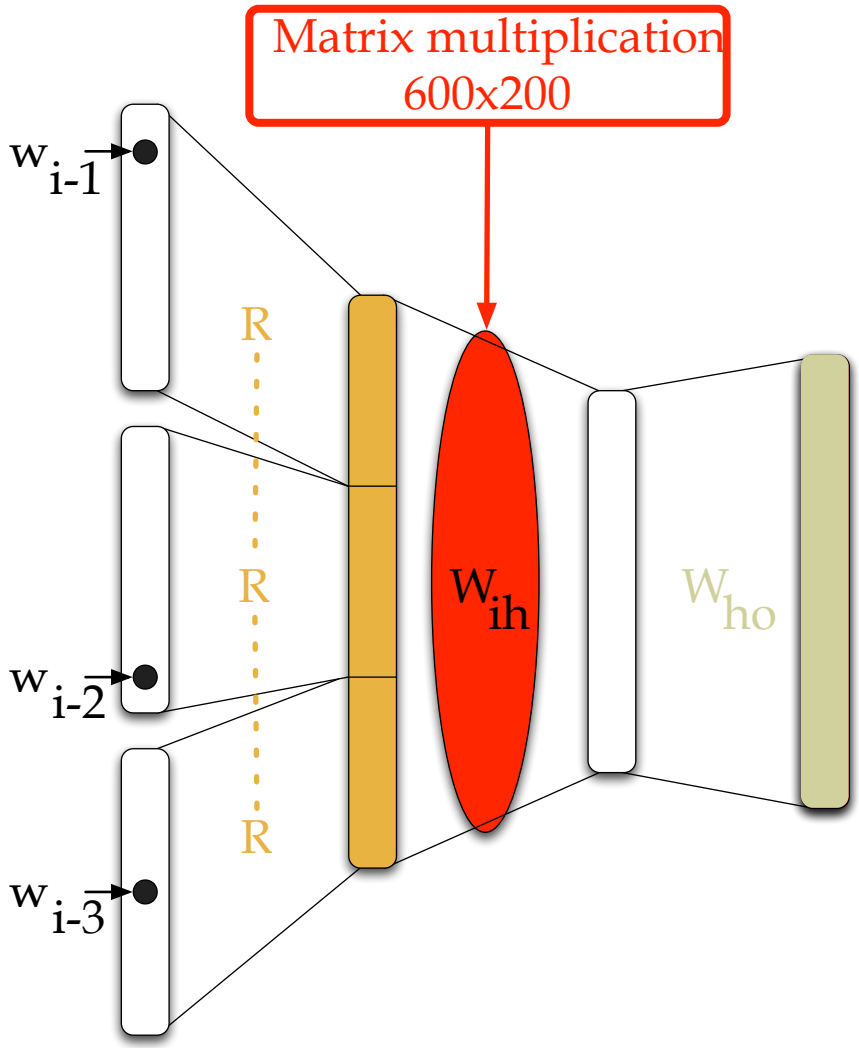
- The projection : select a row in $\mathbf{R}$



Why it is so long ? - Inference

Forward propagation of the history

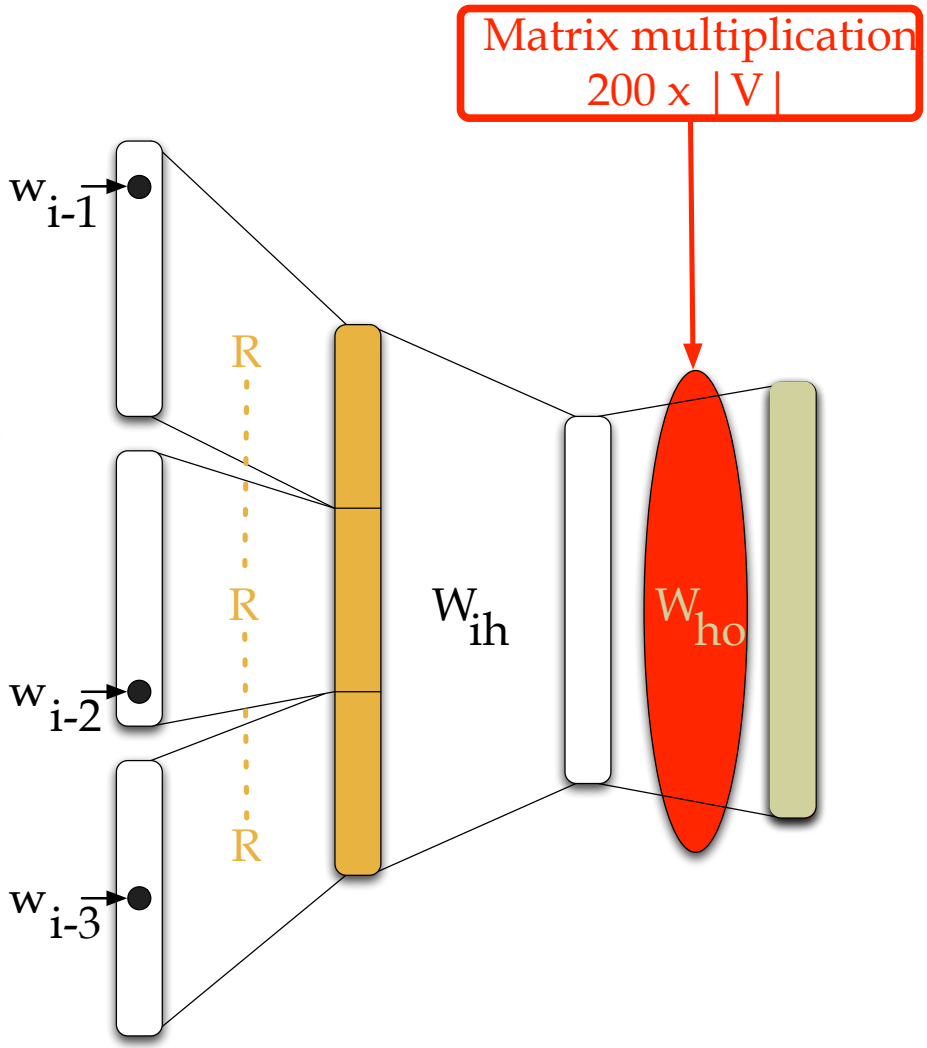
- The projection : select a row in $\mathbf{R}$
- Compute a vector for the predicted word.



Why it is so long ? - Inference

Forward propagation of the history

- The projection : select a row in $\mathbf{R}$
- Compute a vector for the predicted word.
- Estimate the probability for all the words $\in V$



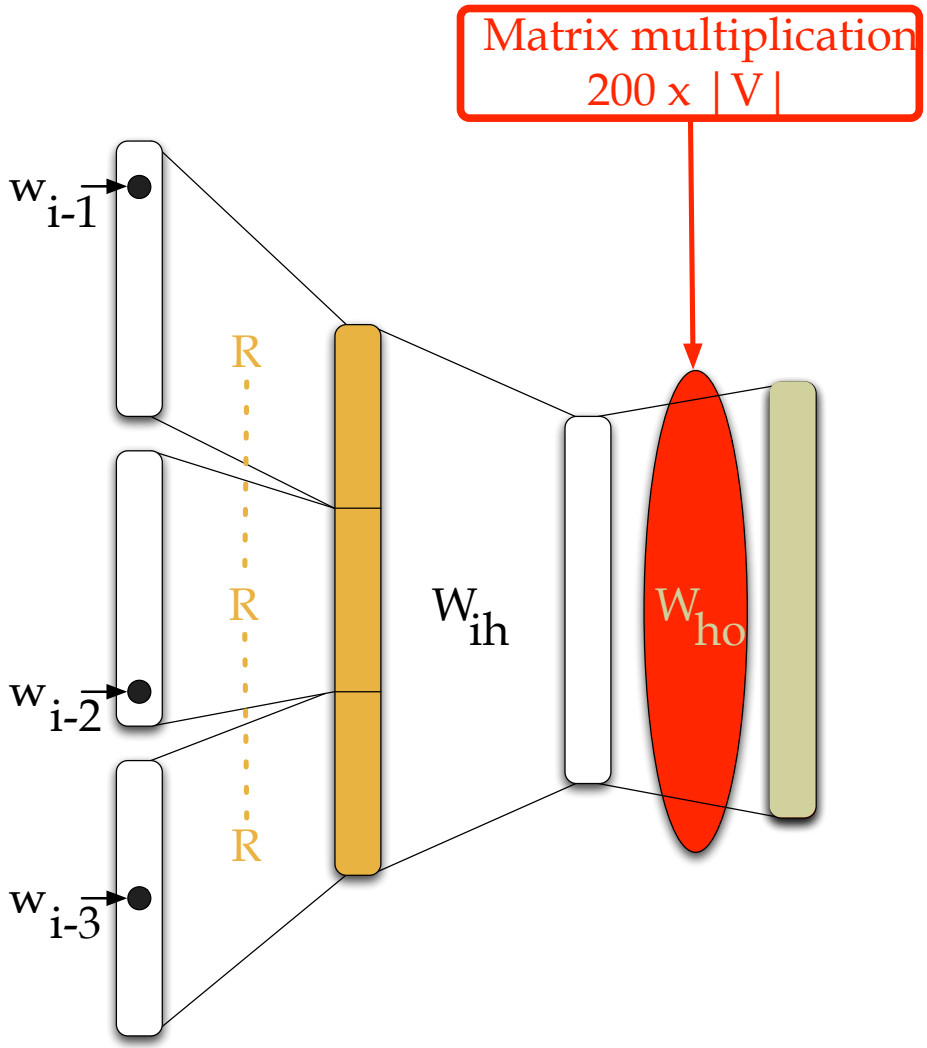
Why it is so long ? - Inference

Forward propagation of the history

- The projection : select a row in $\mathbf{R}$
- Compute a vector for the predicted word.
- Estimate the probability for all the words $\in V$

Complexity issues

- The input vocabulary can be as large as we want.
- Increasing the order of n does not increase the complexity.
- **The problem is the output vocabulary size.**



Usual tricks to speed-up training (and inference)

Re-sampling and batch training

- For each epoch : down-sampling of the training data
- Forward and Back-propagation for a group of n-grams

Reduce the output vocabulary

- Use the Neural network to predict only the K most frequent words.
- For a tractable model : $K = 6,000$ to $20,000$.
- Requires the normalization of the distribution for the whole vocabulary.

$\Rightarrow$ use the standard n-gram LM.

A solution : class-based language model

Main ideas

As proposed by (Mnih and Hinton2008) :

- Represent the vocabulary as a clustering tree (Brown et al.1992).
- Predict the path in this clustering tree.

A solution : class-based language model

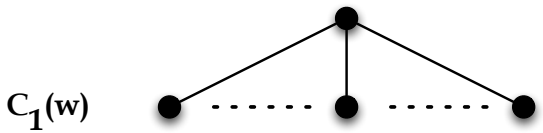
Main ideas

As proposed by (Mnih and Hinton2008) :

- Represent the vocabulary as a clustering tree (Brown et al.1992).
- Predict the path in this clustering tree.

Word clustering

- Put each word w with a single root class $c_1(w)$
- Split these word classes in sub-classes $(c_2(w))$ and so on.



A solution : class-based language model

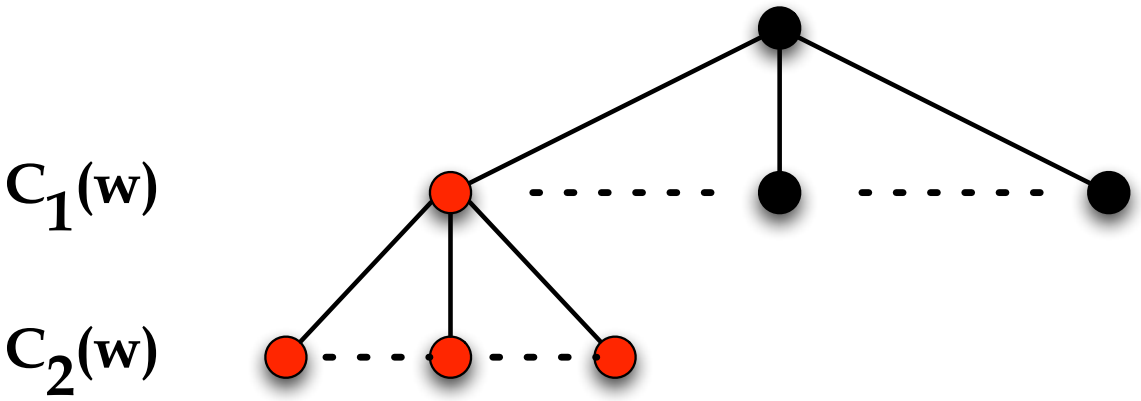
Main ideas

As proposed by (Mnih and Hinton2008) :

- Represent the vocabulary as a clustering tree (Brown et al.1992).
- Predict the path in this clustering tree.

Word clustering

- Put each word w with a single root class $c_1(w)$
- Split these word classes in sub-classes ($c_2(w)$) and so on.



A solution : class-based language model

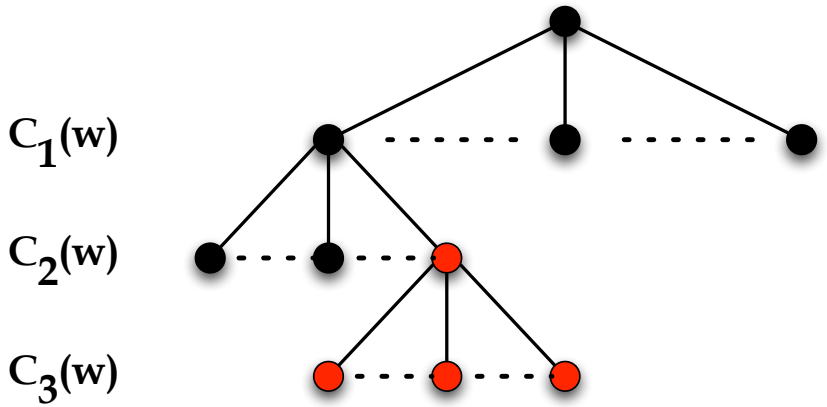
Main ideas

As proposed by (Mnih and Hinton2008) :

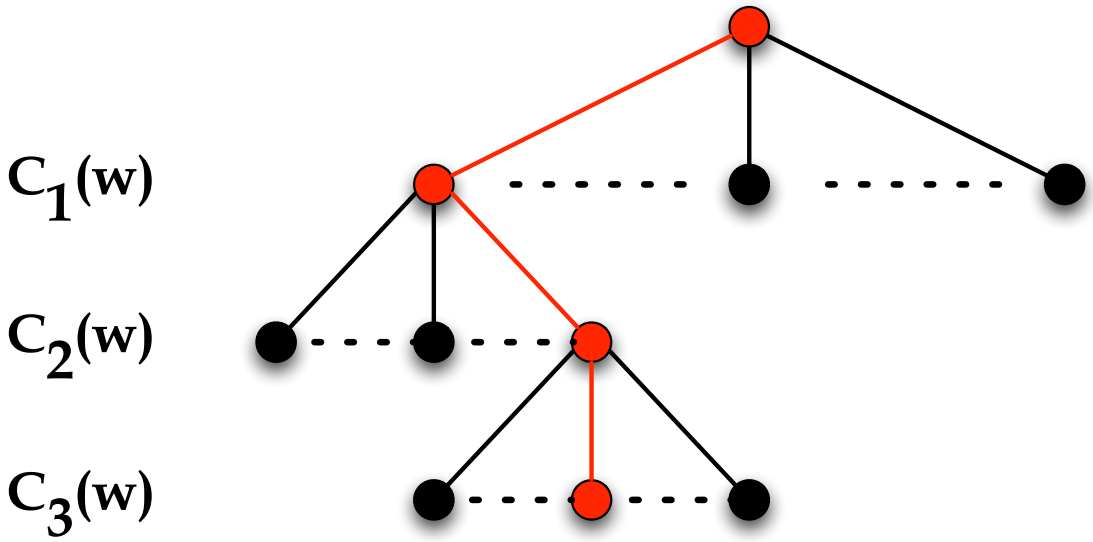
- Represent the vocabulary as a clustering tree (Brown et al.1992).
- Predict the path in this clustering tree.

Word clustering

- Put each word w with a single root class $c_1(w)$
- Split these word classes in sub-classes $(c_2(w))$ and so on.



Word probabilities

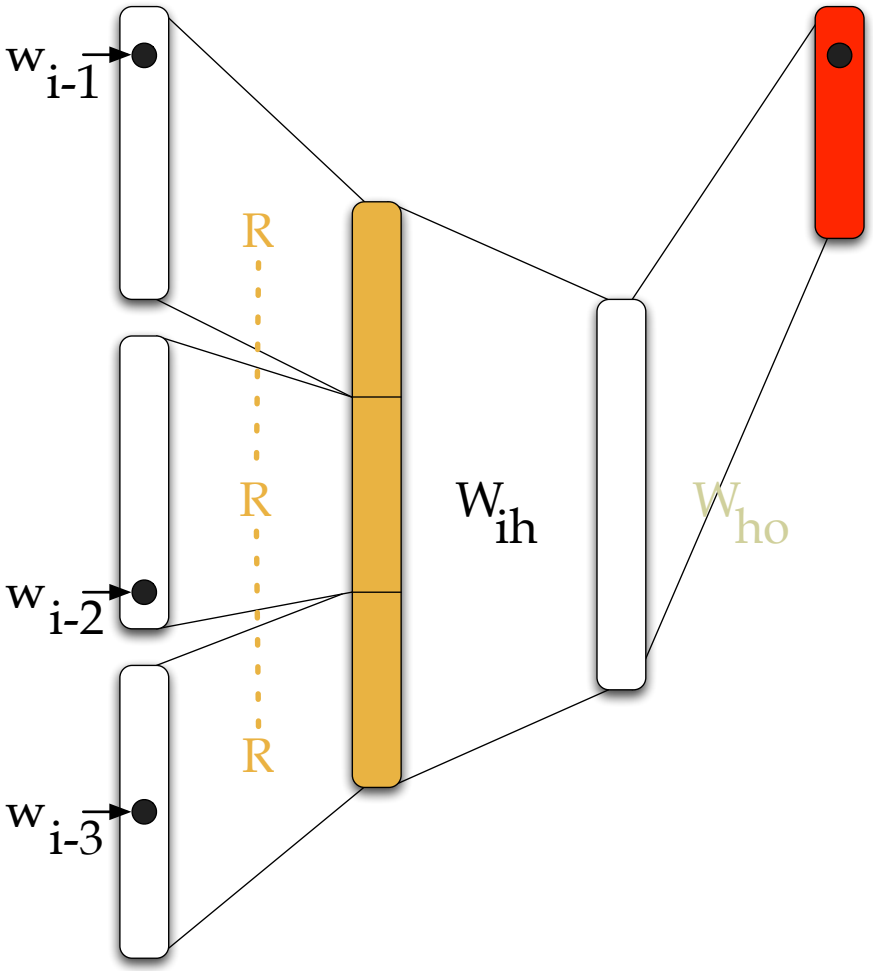


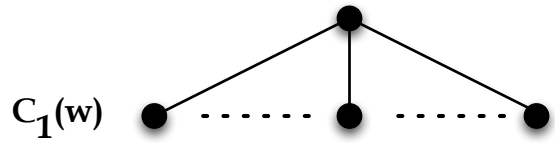
$$P(w_i|h) = P(c_1(w_i)|h) \prod_{d=2}^{D} P(c_d(w_i)|h, c_{1:d-1})$$

- $c_{1:D}(w_i) = c_1, \ldots, c_D$: path for the word w_i in the clustering tree,
- D : depth of the tree,
- $c_d(w_i)$: (sub-)class,
- $c_D(w_i)$: leaf.

The SOUL language model

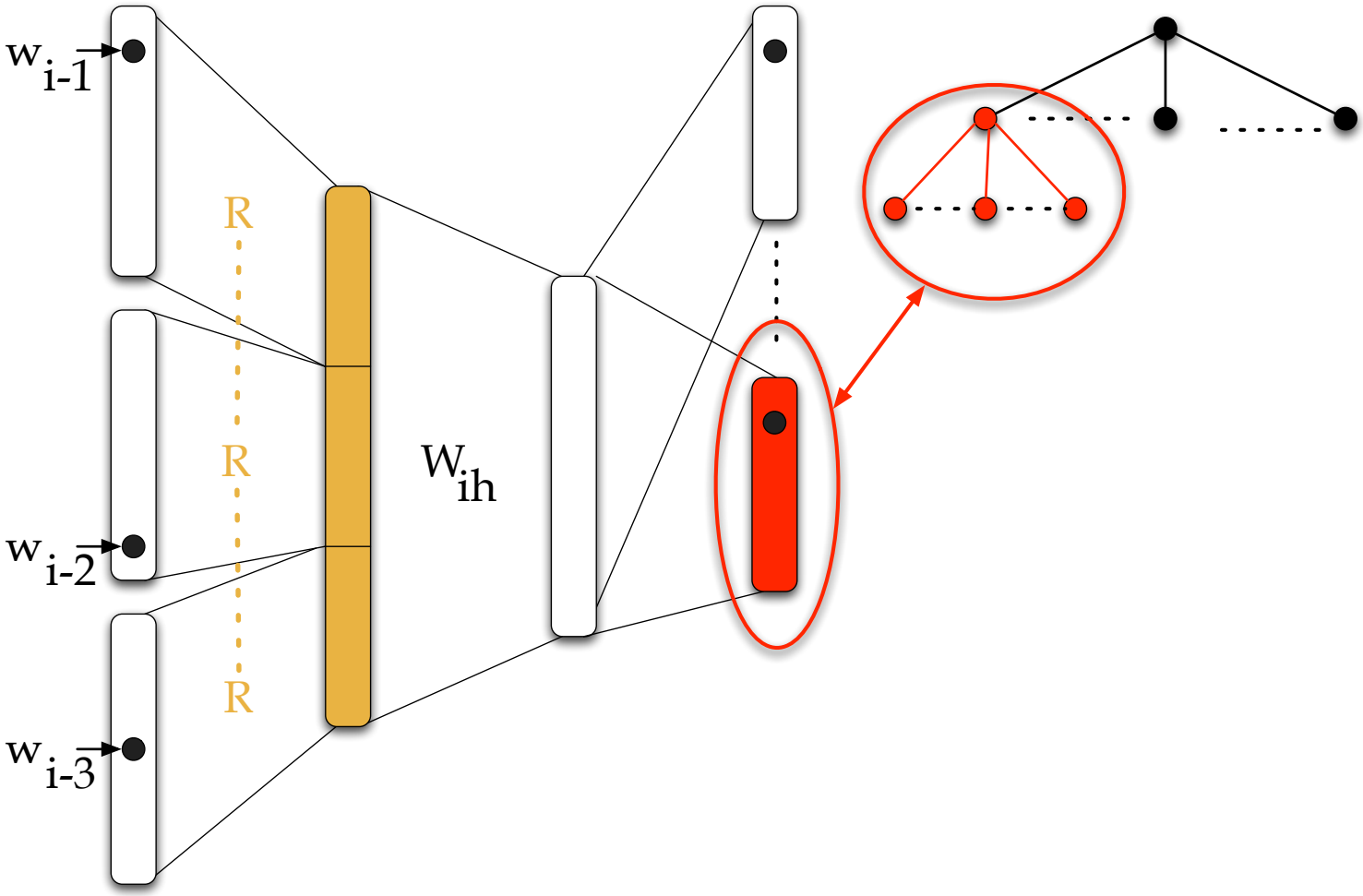
(Le et al.2011)





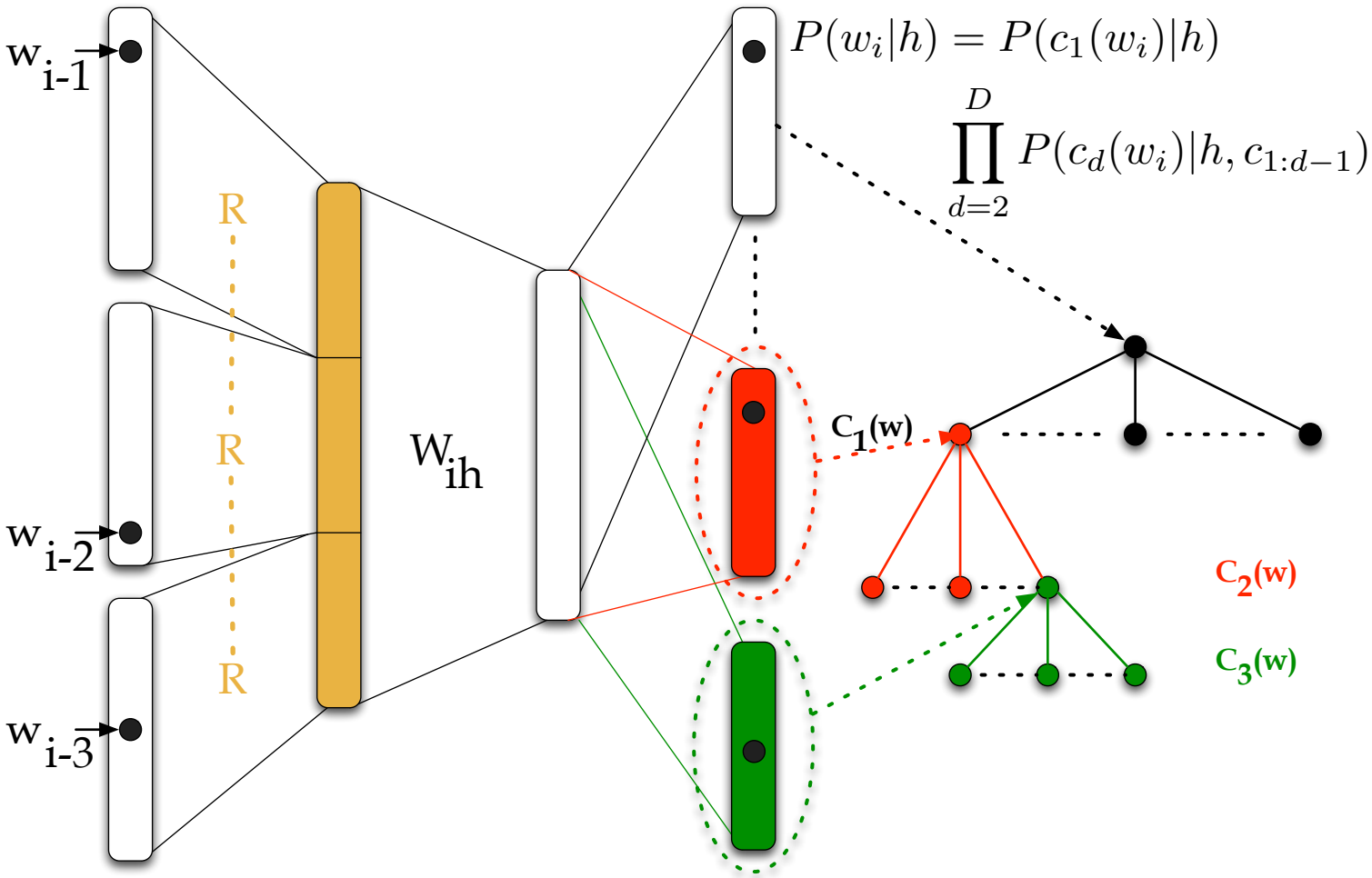
The SOUL language model

(Le et al.2011)



The SOUL language model

(Le et al.2011)



The SOUL language model

Overview

- A class-based language model
- Estimated in a continuous space with neural networks
- The class introduction $\Rightarrow$ many small output layers instead of a single but big layer.

The SOUL language model

Overview

- A class-based language model
- Estimated in a continuous space with neural networks
- The class introduction $\Rightarrow$ many small output layers instead of a single but big layer.

In practice

- The first level : The 8k most frequent words + 4k word classes
- The depth is between 3 and 4.

Another solution : self-normalized model

Gradient computation

For a word w in its context h, the model provides $P^h(w)$ and the gradient is :

$$\frac{\partial l(\boldsymbol{\theta}, w, h)}{\partial \boldsymbol{\theta}} = \frac{\partial a^h_{\boldsymbol{\theta}}(w)}{\partial \boldsymbol{\theta}} - E_{P^h_{\boldsymbol{\theta}}}\left[\frac{\partial a^h_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}\right]$$

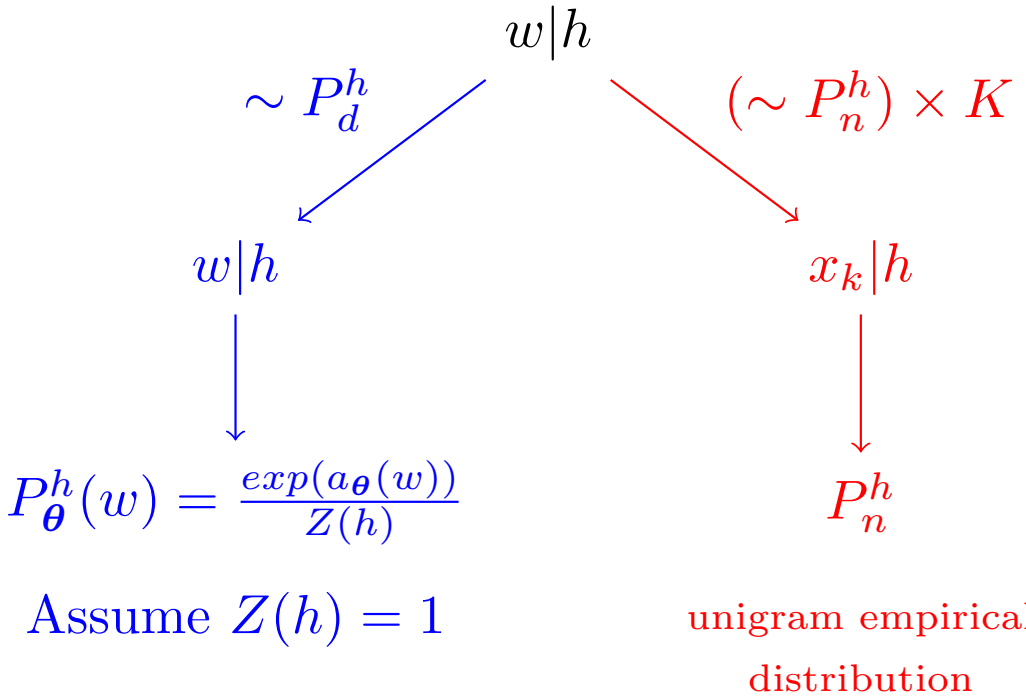
with $a^h_{\boldsymbol{\theta}}(w)$ the pre-activation associated to the word w.

How to approximate the second term

- Important sampling (Bengio and Sénécal2003) is not efficient

$\Rightarrow$ Noise Contrastive Estimation, introduced in (Gutmann and Hyvärinen2010) and applied to language model in (Mnih and Teh2012).

Noise Contrastive Estimation (NCE)



$$J^h(\theta) = E_{P_d^h}\left[log\frac{P_\theta^h(w)}{P_\theta^h(w) + kP_n^h(w)}\right] + kE_{P_n^h}\left[log\frac{kP_n^h(w)}{P_\theta^h(w) + kP_n^h(w)}\right]$$

Why does it work ?

The gradient of the NCE objective function

The gradient can be written as follows :

$$\sum_{w \in V} \frac{k P_n^h(w)}{P_\theta^h(w) + k P_n^h(w)} \times (P_d^h(w) - P_\theta^h(w)) \frac{\partial \log P_\theta^h(w)}{\partial \theta}$$

- The noise distribution must be non-zero wherever the data distribution is.
- It onverges to the maximum likelihood gradient when $k \longrightarrow \infty$

Inference

- The model converge to an approximately normalized model

$\Rightarrow$ During inference, just need to compute $a_{\boldsymbol{\theta}}^h(w)$

Summary for large vocabulary application

The size of the output layer has a great impact on the computational cost.

Structured output

- Replace one large softmax by several smaller ones,
- with a hierarchical softmax (Morin and Bengio2005).
- See the work of (Mnih and Hinton2008; Le et al.2011; Mikolov et al.2011).

Un-normalized model

- Preserve the conventional output structure
- A simplified version : Negative Sampling (Mikolov et al.2013)
- Variant : include the normalizer in the objective function (Devlin et al.2014)

Outline

Sentence level representation

The goal

- To derive a continuous representation of a sentence (or a document).
- Not a bag of words !

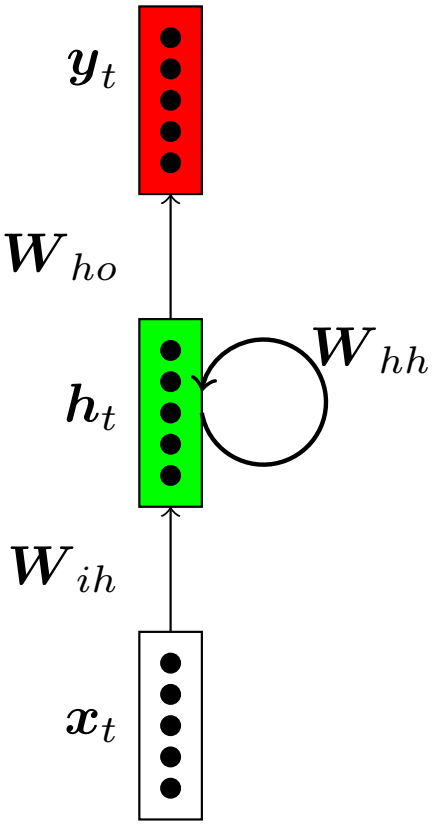
"Deep Net" Solutions

There are two ways to deal with sequences of arbitrary size.

- Recurrent network (Mikolov et al.2011)
- Convolutional network + pooling (Kalchbrenner and Blunsom2013; Kalchbrenner et al.2014)

Recurrent network

Interlude introducing new architectures



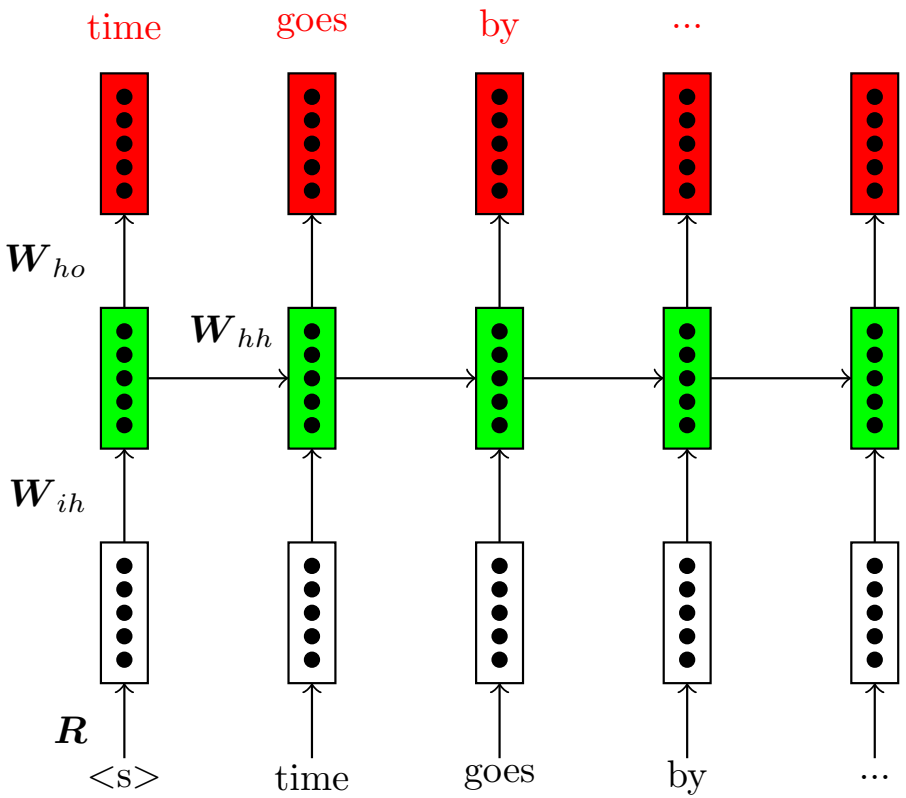
A dynamic system, at time t :

- maintains a hidden representation, the internal state : $\boldsymbol{h}_t$
- Updated with the observation of $\boldsymbol{x}_t$ and the previous state $\boldsymbol{h}_{t-1}$
- The prediction $\boldsymbol{y}_t$ depends on the internal state ($\boldsymbol{h}_t$)
- For a language model, $\boldsymbol{x}_t$ comes from word embeddings

The same parameter set is shared across time steps

Recurrent network language model

Unfolding the structure : a deep-network



At each step t

- Read the word $w_t \rightarrow \boldsymbol{x}_t$ from $\boldsymbol{R}$
- Update the hidden state $\boldsymbol{h}_t = f(\boldsymbol{W}_{ih}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hh}}\boldsymbol{h}_{\boldsymbol{t-1}})$
- The word at $t+1$ can be predicted from $\boldsymbol{h}_t$:

$$\boldsymbol{y}_t = g(\boldsymbol{W}_{ho}\boldsymbol{h}_t)$$

- g is the softmax function over the vocabulary

Training recurrent language model

Training algorithm

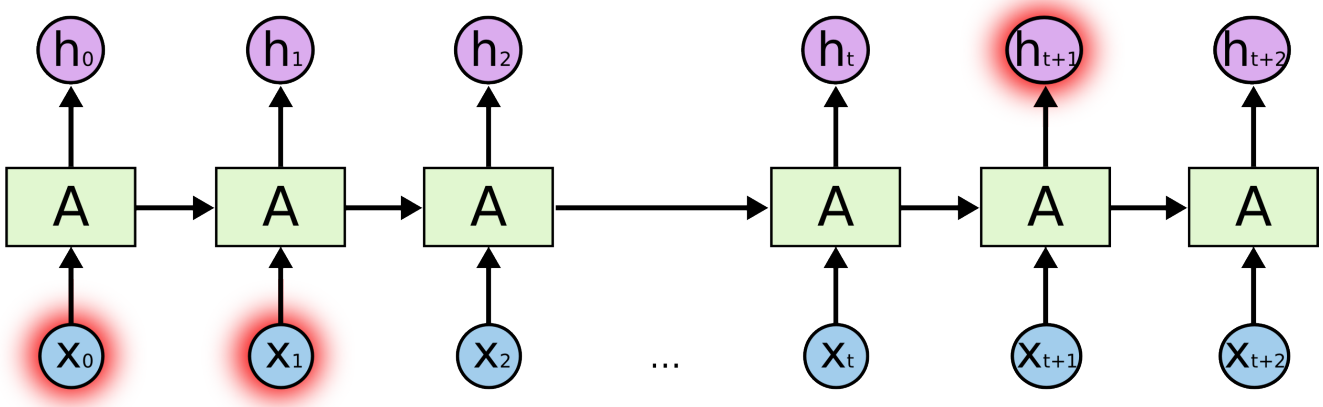
Back-Propagation through time or BPTT (Rumelhart et al.1986; Mikolov et al.2011) :

- for each step t
  - compute the loss gradient
  - Back-Propagation through the unfolded structure

Inference

- Cannot be easily integrated to conventional approaches (ASR, SMT, ...)
- A powerful device for end-to-end system

The Problem of Long-Term Dependencies



ex : "I grew up in France... I speak fluent French"

- Recent observations hide the older ones (Bengio et al.1994)
- The vanishing (exploding) gradient is a real issue (Pascanu et al.2013)

Solutions

Improved optimization

- Gradient clipping (Pascanu et al.2013)
- Hessian-Free optimzation (Martens and Sutskever2012) or natural gradient (Desjardins et al.2013; Ollivier2015)

Modified unit

A recurrent network should be able to mitigate the observations *vs* its internal state :

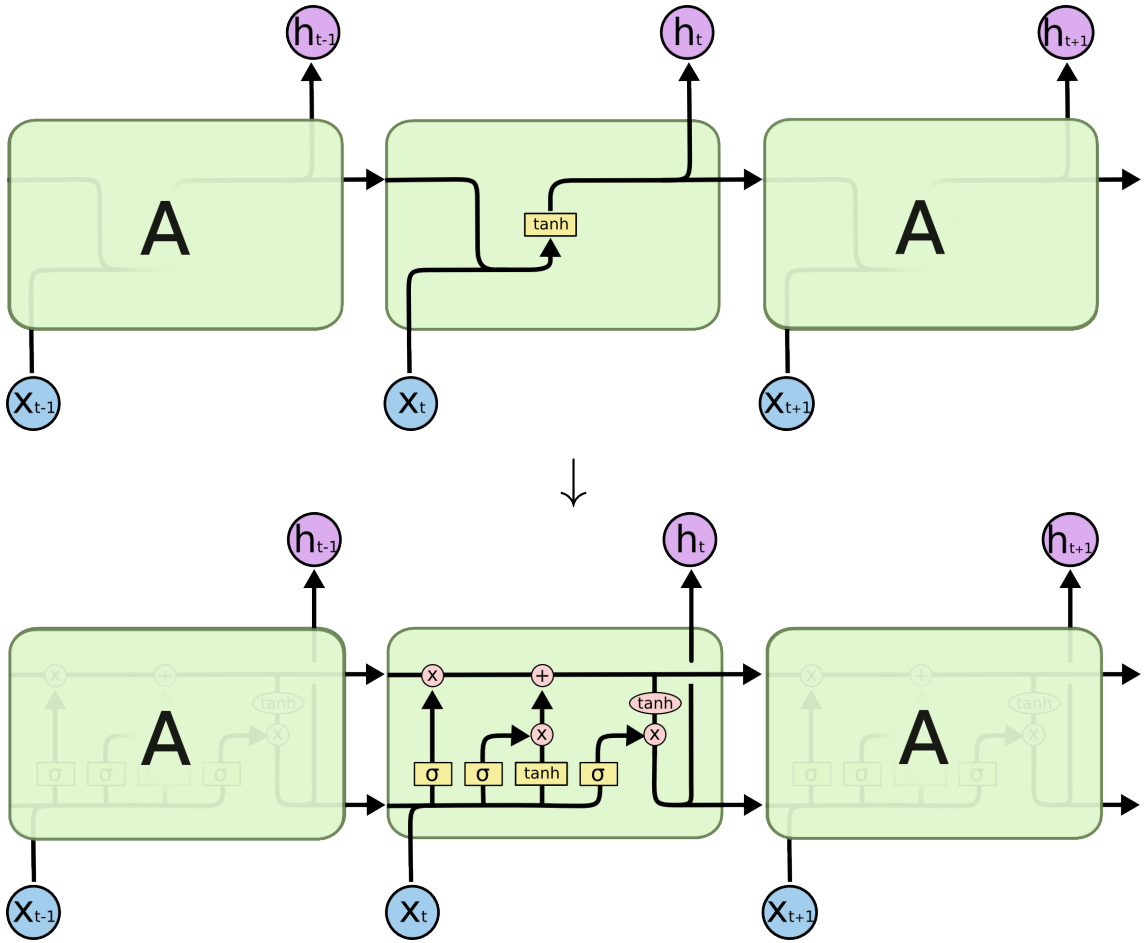
- LSTM or Long-Short-Term-Memory cell (Hochreiter and Schmidhuber1997; Graves and Schmidhuber2009)
- Gated Recurrent Unit or GRU (Cho et al.2014)

Introduction to LSTM

Based on http://colah.github.io/posts/2015-08-Understanding-LSTMs/

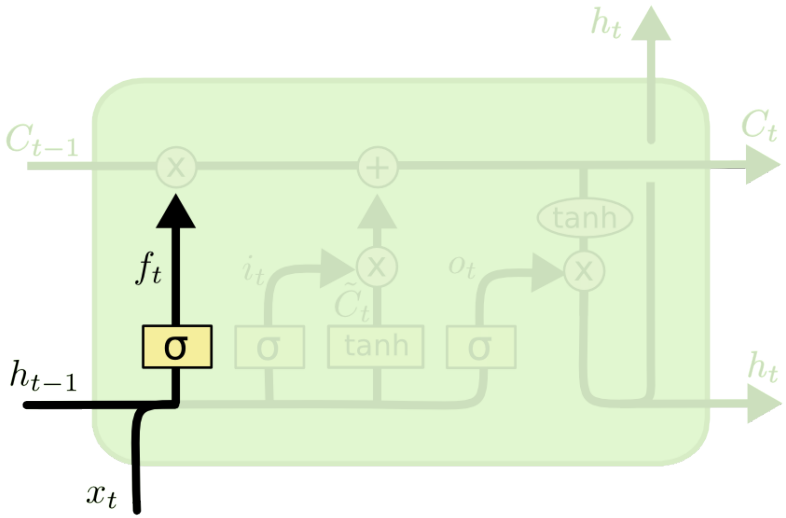
Instead of having a single neural network layer, there are four, interacting in a very special way.

- Lines carry a vector
- A recurrent net transmits its hidden states
- LSTM introduces a second channel : the cell state
- It acts as a memory
- Gates control the memory



LSTM : Control flow - 1

What should be forgotten from the previous cell state ?



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Action

The sigmoid (forget gate) answers for each component :

- 1 : to keep it,
- 0 to forget it, or
- a value in-between to mitigate its influence

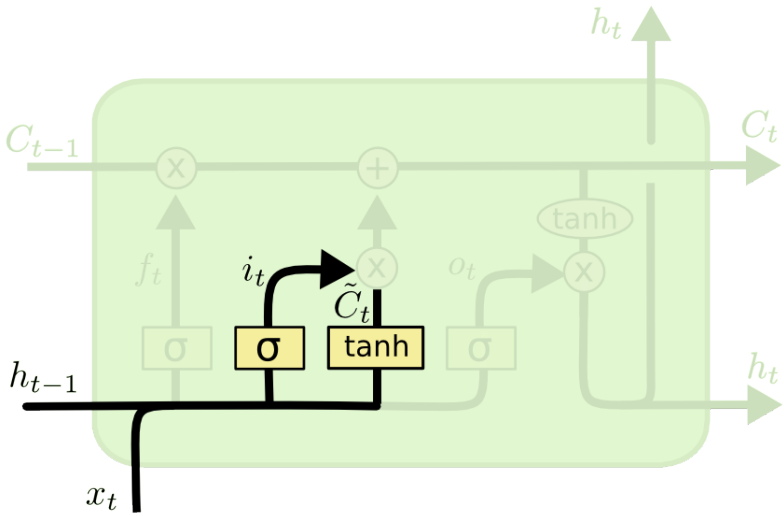
Intuition for language modeling

The cell state might embed the gender of the present subject :

- keep it to predict the correct pronouns,
- or forget it, when a new subject appears

LSTM : Control flow - 2

What should be taken into account ?



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Actions

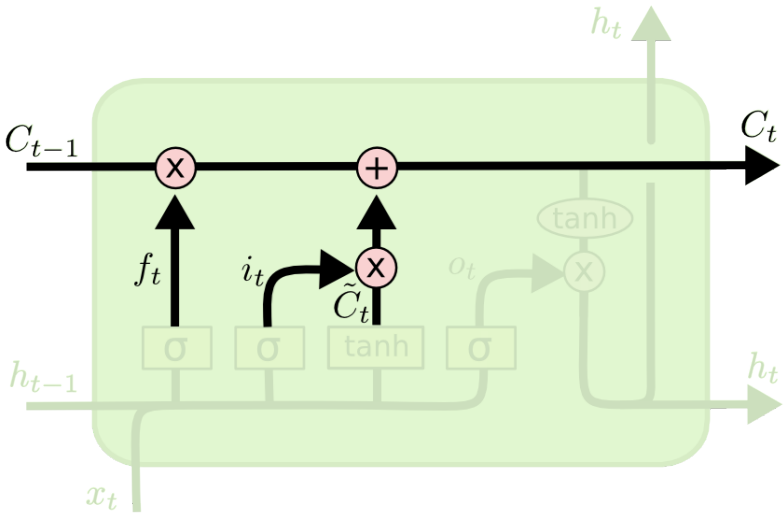
- Create the update $\tilde{C}_t$ of the cell state
- and its contribution i_t (the input gate with a sigmoid activation)

Intuition

- Add the gender of the new subject to the cell state,
- to replace the old one we're forgetting.

LSTM : Control flow - 3

Write the new state



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Actions

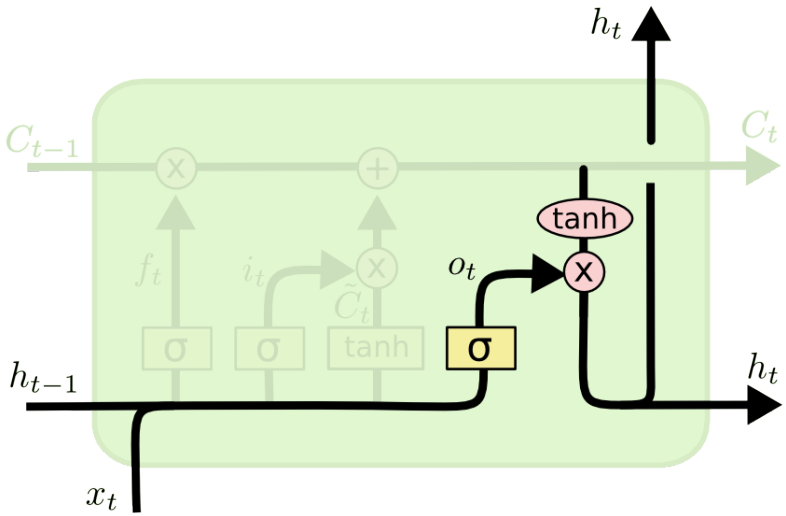
- Merge the old cell state modified by the forget gate
- with the new input

Intuition for language modeling

- Decide to drop the information about the old subject
- Refresh the memory

LSTM : Control flow - 4

Write the new hidden state



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Actions

- Decide what parts of the (filtered) cell state to output o_t
- Compute the hidden state

Intuition for language modeling

- Since we just saw a subject,
- output the relevant information for the future (gender, number)

LSTM summary

A special kind of recurrent architecture

The internal cell state allows the model to :

- keep information in memory
- select the relevant outputs
- to reset or mitigate the long-term memory

Consequences

- An efficient model of sequences
- Overcome the vanishing gradient issue
- Very promising results in generation

Variants

- Gated recurrent units (GRU) (Cho et al.2014) or a more complicated one (Gers and Schmidhuber2000)
- Some recent comparisons (Józefowicz et al.2015; Greff et al.2015)

Outline

Motivations

For morphologically-rich and non-canonical languages

- Very productive word formation processes

$\Rightarrow$ generate a proliferation of word forms.

Freundschaftsbezeigungen, görüntülenebilir $\leftrightarrow$ *MYL, AFAIK, cul8r*

Consequences

- Morphologically-rich and under-resourced language processing
- Social Media implies fast change in the language use

$\rightarrow$ An evolving vocabulary with new compound tokens, abbreviations, ...

Tokens decipherment/encoding

Ex : German POS-Tagging

The Task

The TIGER corpus defines a POS-tagging task with very rich tagset (around 600 tags)

State of the art results (Mueller et al.2013)

- A second order CRF
- with an intensive feature engineering to describe the morphology

Deep Net approach

- A lot of information about words can be leveraged from subword features.

$\Rightarrow$ Learn to infer a word representation from the character level

$\Rightarrow$ Make these representations aware of the context (sentence level)

A training example

Sentence	POS-tags
Er	PPER-case=nom|@gender=masc|number=sg|person=3
fürchtet	VVFIN-mood=ind|number=sg|person=3|tense=pres
noch	ADV
Schlimmeres	NN-case=acc|gender=neut|number=sg
.	$.

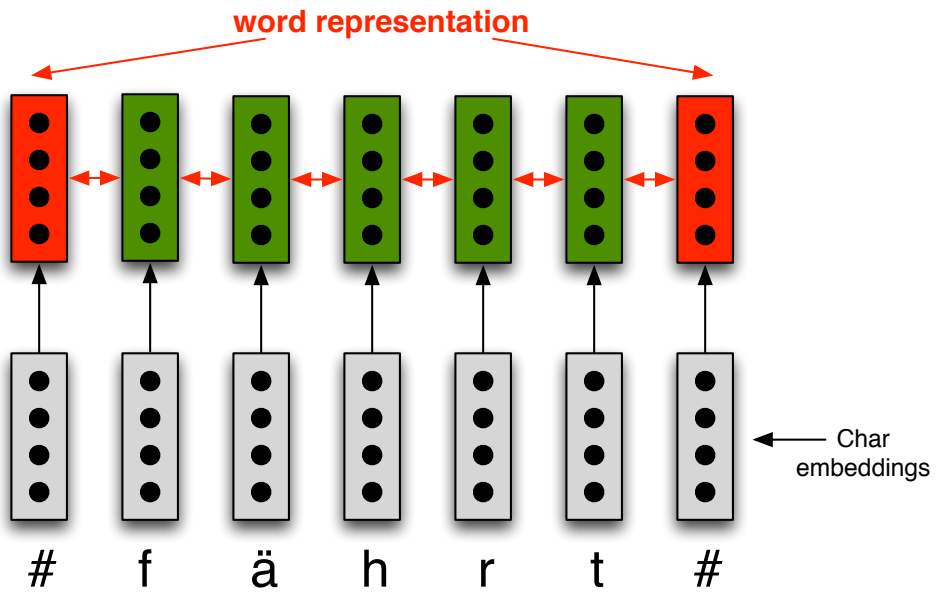
From characters to word representation

A word is a sequence of characters !

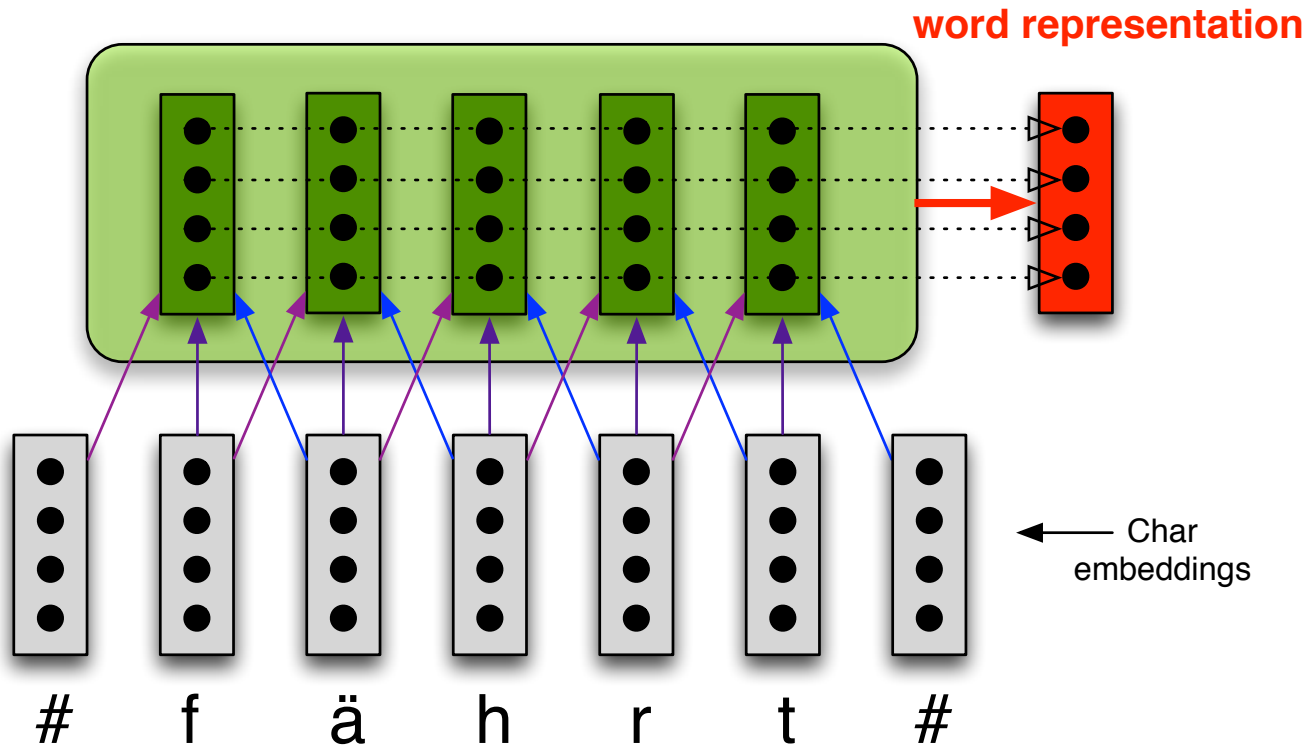
Sequence representation

- Recurrent network (Elman1990; Mikolov et al.2011)
- Convolutional network + pooling (Kalchbrenner et al.2014; Santos and Zadrozny2014)

The "recurrent" way

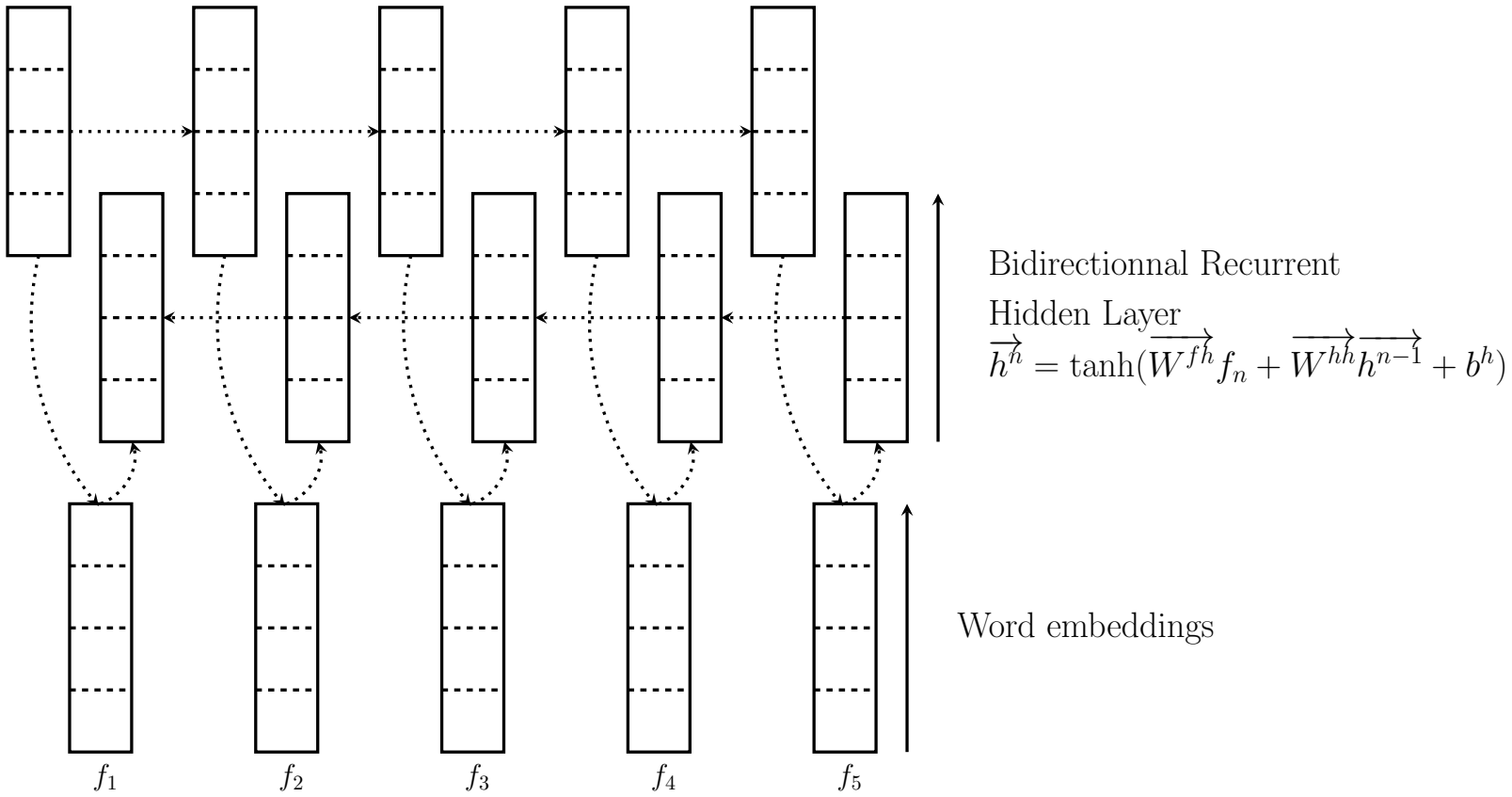


Convolutional net and pooling



- A convolution net is applied at each position
- In 1-D, it mixes the inputs within a window (represent the local context)
- Max-pooling reduces this sequence in one vector

From word to a contextual representation



For each word, derive a new representation, sentence dependant

From sequence to prediction

Simple prediction : position by position

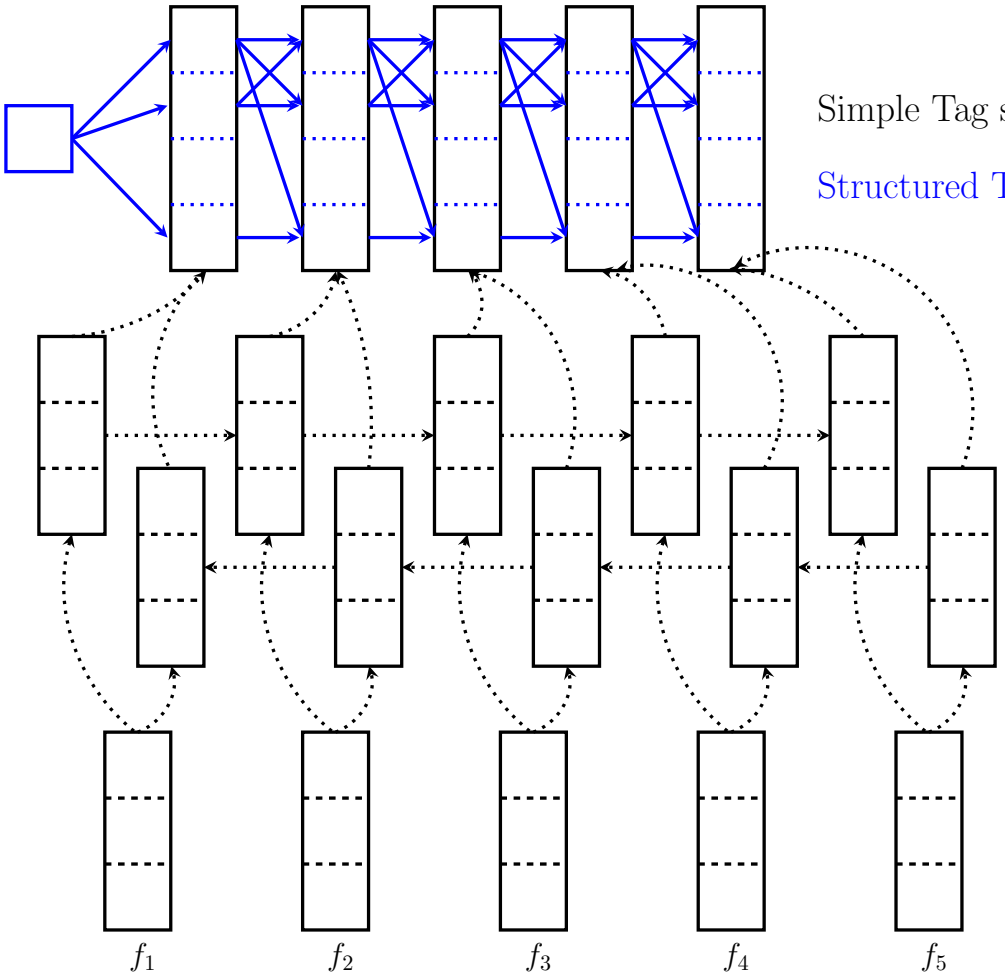
$$s_n = W^o(\overrightarrow{h^n} : \overleftarrow{h^n}) + b^o.$$

Sequence prediction (Collobert2011; Santos and Zadrozny2014)

Add a transition matrix W^{trans} and compute with the Viterbi algorithm, the best tag sequence $\{t_1, \ldots, t_{|s|}\}$ *wrt* :

$$s(\{t\}_1^{|s|}, \{w\}_1^{|s|}) = \sum_{1 \leq n \leq |s|} \left(W^{trans}_{t_{n-1}, t_n} + [s_n]_{t_n} \right)$$

Sequence prediction



Simple Tag scoring : $s_n = W^o(\overrightarrow{h^h} : \overleftarrow{h^n}) + b^o$

Structured Tag prediction (Viterbi inference)

Bidirectionnal Recurrent Hidden Layer

$\overrightarrow{h^h} = \tanh(\overrightarrow{W^{fh}} f_n + \overrightarrow{W^{hh}} \overrightarrow{h^{n-1}} + b^h)$

Word embeddings

Putting all together

(Labeau et al.2015)

A unified model that can

- Infer word representation from the character level
- which takes sentence level information into account
- Make sequence prediction

Training

- Maximize the conditional log-likelihood of the tag sequence given the word sequence
- Optimization with AdaGrad (Duchi et al.2011)

Results

- This model achieves state of the art performance
- without any feature engineering

Summary

The basics : the Feedforward architecture

- Basically matrix operation and a specific terminology
- Training : Back-propagation and SGD
- Regularization : weight decay and dropout

For structured i/o

- Recurrent network and LSTM
- Convolutional Net and pooling

Conclusion

Deep Networks learn higher levels of abstraction

Higher-level abstractions disentangle the factors of variation :

$\Rightarrow$ Hierarchical layers of representation

$\Rightarrow$ Allows the model a much easier generalization and transfer

Why today ?

- After a long and intensive experimental exploration,
- efficient recipies and tools are available
- but Training Deep Nets is not so "simple"

In practice

- Often very simple matrix derivatives (backprop) for training and matrix multiplications for inference
- Fast inference and well suited for multicore CPUs/GPUs and parallelization across machines

Conclusion - 2

A success story

Many state of the art results on a variety of language and vision tasks

For natural language processing

Continuous representation of words, sentences, ... opens perspectives

- Sequence tagging (Collobert et al.2011), Parsing (Vinyals et al.2015), question answering (Weston et al.2015), **Semantics** (Socher et al.2013; Baroni et al.2014), ...
- Machine Translation (a special issue in Computer Speech and Language)

Warning :
This is not magical and this won't solve all the problems.

> "Natural language processing (NLP) has not been revolutionized by deep learning, though there has been huge progress during the last years."
>
> (Y. Bengio @ DL-workshop, ICML2015)

Future work

In NLP

- Under-resourced, non-canonical, and morphologically-rich languages
- Transfer learning
- A unified framework for joint tasks, *e.g* speech-to-speech translation

Unsupervised and semi-supervised learning

There is a huge amount of unlabeled data!

- Unsupervised learning (RBM, Auto-Encoder, ...) strikes back (Larochelle et al.2009)
- Semi-supervised learning (Kingma et al.2014; Rasmus et al.2015)
- Multi-task learning (sharing parameters across scarce tasks) (Collobert et al.2011)

Toolkits

Theano : http://deeplearning.net/software/theano/

- in python, works on CPU and GPU and several wrappers
- http://lasagne.readthedocs.org/
- http://keras.io/
- https://www.cs.cmu.edu/~ymiao/pdnntk.html
- http://deeplearning.net/software/pylearn2/
- http://blocks.readthedocs.org/

Torch7 : http://torch.ch/

Lua interface to C/CUDA

TensorFlow https://www.tensorflow.org

API in C++ and Python

Further reading

- *Deep Learning Book* by Yoshua Bengio, Ian Goodfellow and Aaron Courville (`http://www.iro.umontreal.ca/~bengioy/dlbook/`)
- *Deep Learning for Natural Language Processing* by Stanford University `http://cs224d.stanford.edu/`
- *Deep Learning - Methods and Applications* by Li Deng and Dong Yu `http://research.microsoft.com/pubs/219984/BOOK2014.pdf`
- *Reading lists for new LISA students* by University of Montreal.


Marco Baroni, Georgiana Dinu, and Germán Kruszewski.
2014.
Don't count, predict ! a systematic comparison of context-counting vs. context-predicting semantic vectors.
In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 238–247, Baltimore, Maryland, June. Association for Computational Linguistics.


Yoshua Bengio and Réjean Ducharme.
2001.
A neural probabilistic language model.
In *Advances in Neural Information Processing Systems (NIPS)*, volume 13. Morgan Kaufmann.


Yoshua Bengio and Jean-Sébastien Sénécal.
2003.
Quick training of probabilistic neural nets by importance sampling.
In *Proceedings of the conference on Artificial Intelligence and Statistics (AISTATS)*.


Y. Bengio, P. Simard, and P. Frasconi.
1994.
Learning long-term dependencies with gradient descent is difficult.
Trans. Neur. Netw., 5(2) :157–166, March.


Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin.

2003.

A neural probabilistic language model.

Journal of Machine Learning Research, 3 :1137–1155.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai.

1992.

Class-based n-gram models of natural language.

Computational Linguistics, 18(4) :467–479.

Stanley F. Chen and Joshua T. Goodman.

1998.

An empirical study of smoothing techniques for language modeling.

Technical Report TR-10-98, Computer Science Group, Harvard University.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio.

2014.

Learning phrase representations using rnn encoder–decoder for statistical machine translation.

In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa.

2011.

Natural language processing (almost) from scratch.

Journal of Machine Learning Research, 12 :2493–2537.

Ronan Collobert.

2011.

Deep learning for efficient discriminative parsing.

In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 224–232.

Guillaume Desjardins, Razvan Pascanu, Aaron Courville, and Yoshua Bengio.

2013.

Metric-free natural gradient for joint-training of boltzmann machines.

In *International Conference on Learning Representations (ICLR'2013)*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul.

2014.

Fast and robust neural network joint models for statistical machine translation.

In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 1370–1380, Baltimore, Maryland, June. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer.

2011.

Adaptive subgradient methods for online learning and stochastic optimization.
J. Mach. Learn. Res., 12 :2121–2159, July.

Jeffrey L. Elman.
1990.
Finding structure in time.
Cognitive Science, 14(2) :179–211.

F.A. Gers and J. Schmidhuber.
2000.
Recurrent nets that time and count.
In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, pages 189–194 vol.3.

Alex Graves and Juergen Schmidhuber.
2009.
Offline handwriting recognition with multidimensional recurrent neural networks.
In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber.
2015.
Lstm : A search space odyssey.
arXiv preprint arXiv :1503.04069.

M. Gutmann and A. Hyvärinen.

2010.

Noise-contrastive estimation : A new estimation principle for unnormalized statistical models.

In Y.W. Teh and M. Titterington, editors, *Proceedings of th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 297–304.

Sepp Hochreiter and Jürgen Schmidhuber.

1997.

Long short-term memory.

Neural Comput., 9(8) :1735–1780, November.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever.

2015.

An empirical exploration of recurrent network architectures.

In *Proceedings of the International Conference of Machine Learning (ICML)*, pages 2342–2350.

Nal Kalchbrenner and Phil Blunsom.

2013.

Recurrent convolutional neural networks for discourse compositionality.

In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126, Sofia, Bulgaria, August. Association for Computational Linguistics.


Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom.
2014.
A convolutional neural network for modelling sentences.
In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.


Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling.
2014.
Semi-supervised learning with deep generative models.
In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc.


Matthieu Labeau, Kevin Löser, and Alexandre Allauzen.
2015.
Non-lexical neural architecture for fine-grained pos tagging.
In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 232–237, Lisbon, Portugal, September. Association for Computational Linguistics.


Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin.
2009.
Exploring strategies for training deep neural networks.
J. Mach. Learn. Res., 10 :1–40, June.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon.
2011.
Structured output layer neural network language model.
In *Proceedings of ICASSP*, pages 5524–5527.

James Martens and Ilya Sutskever.
2012.
Training deep and recurrent networks with hessian-free optimization.
In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks : Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 479–535. Springer.

Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur.
2011.
Extensions of recurrent neural network language model.
In *Proceedings of ICASSP*, pages 5528–5531.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean.
2013.
Distributed representations of words and phrases and their compositionality.
In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Andriy Mnih and Geoffrey E Hinton.

2008.

A scalable hierarchical distributed language model.

In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, volume 21, pages 1081–1088.

A. Mnih and Y. W. Teh.

2012.

A fast and simple algorithm for training neural probabilistic language models.

In *Proceedings of the International Conference on Machine Learning*.

Frederic Morin and Yoshua Bengio.

2005.

Hierarchical probabilistic neural network language model.

In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze.

2013.

Efficient higher-order CRFs for morphological tagging.

In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October. Association for Computational Linguistics.


Yann Ollivier.
2015.
Riemannian metrics for neural networks i : feedforward networks.
Information and Inference.


Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio.
2013.
On the difficulty of training recurrent neural networks.
In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Proceedings*, pages 1310–1318. JMLR.org.


Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko.
2015.
Semi-supervised learning with ladder network.
CoRR, abs/1507.02672.


D. E. Rumelhart, G. E. Hinton, and R. J. Williams.
1986.
Parallel distributed processing : explorations in the microstructure of cognition, vol. 1.
chapter Learning internal representations by error propagation, pages 318–362. MIT Press, Cambridge, MA, USA.


Cicero D. Santos and Bianca Zadrozny.
2014.

Learning character-level representations for part-of-speech tagging.

In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826. JMLR Workshop and Conference Proceedings.

Holger Schwenk and Jean-Luc Gauvain.

2002.

Connectionist Language Modeling for Large Vocabulary Continuous Speech Recognition.

In *Proceedings of ICASSP*, pages 765–768, Orlando, May.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts.

2013.

Recursive deep models for semantic compositionality over a sentiment treebank.

In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton.

2015.

Grammar as a foreign language.

In *Advances in Neural Information Processing Systems (NIPS) 27*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov.

2015.

Towards ai-complete question answering : A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.