Structured OUtput Layer (**SOUL**) Neural Network Language Model

Le Hai Son, **Ilya Oparin**,
Alexandre Allauzen, Jean-Luc Gauvain, François Yvon

25/05/2011

Outline

Plan

N-gram models

- Very successful **but** sparsity issues and lack of generalization
- Flat vocabulary
  - Each word is only a possible outcome of a discrete random variable, an index in the vocabulary

Estimate *n*-gram probabilities in a continuous space

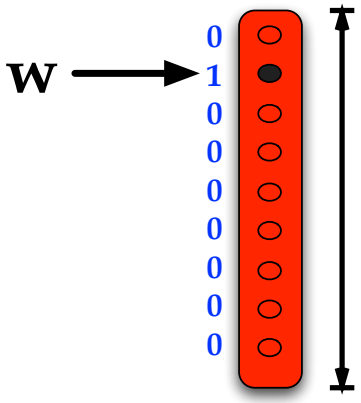
NNLMs were introduced in [Bengio et al., 2001] and applied to speech recognition in [Schwenk and Gauvain, 2002].

Why should it work ?

- ”similar” words are expected to have similar feature vectors
- Probability function is a smooth function of feature values
  - A small change in features will induce a small change in the probability

Project a word sequence in a continuous space

- Represent words as as 1-of-$|V|$ vectors
- 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

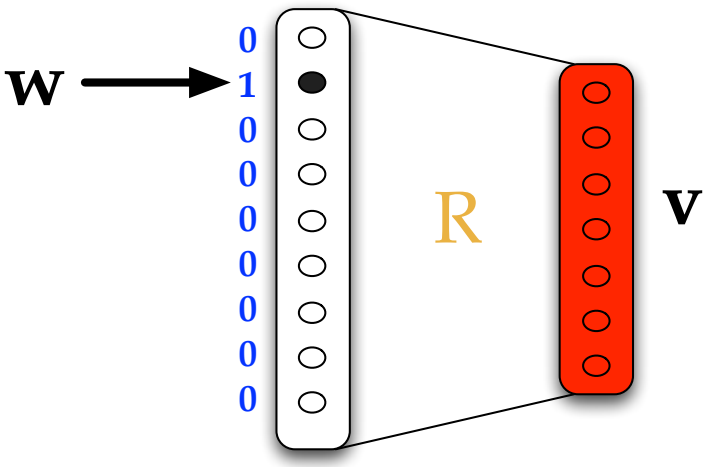


|V|: vocabulary size

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

Project a word sequence in a continuous space

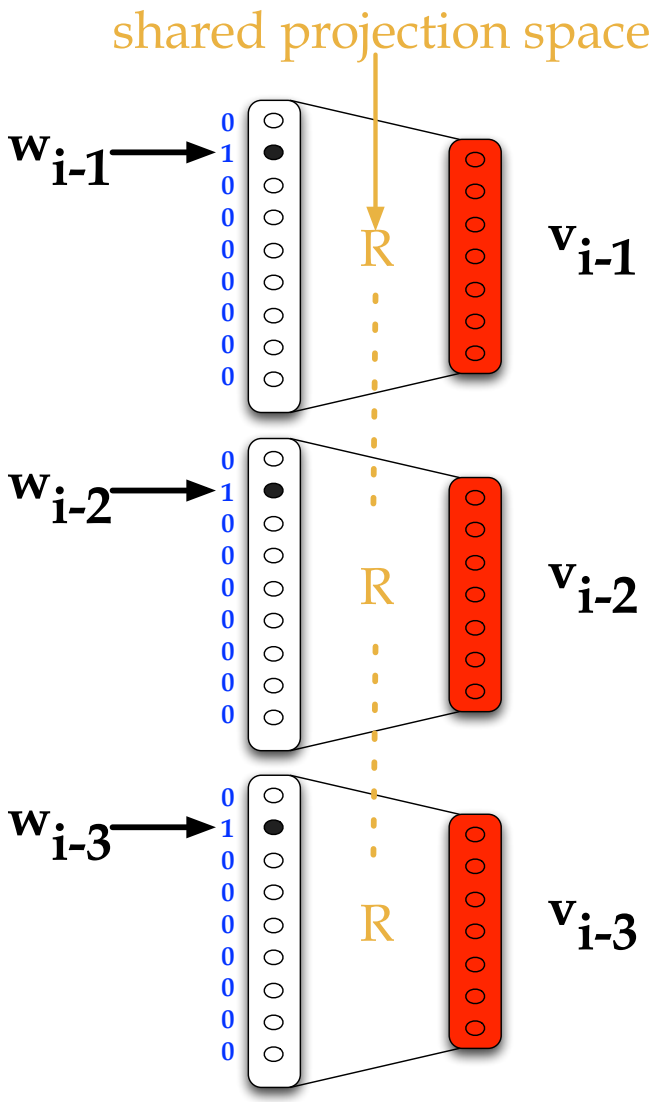
- Represent words as as 1-of-$|V|$ vectors
- Project the word in the continuous space: add a second layer fully connected
- For a 4-gram, the history is a sequence of 3 words
- Merge these three vectors to derive a single vector for the history



- The connection between two layers is a matrix operation
- The matrix **R** contains all the connection weights
- **v** is a continuous vector

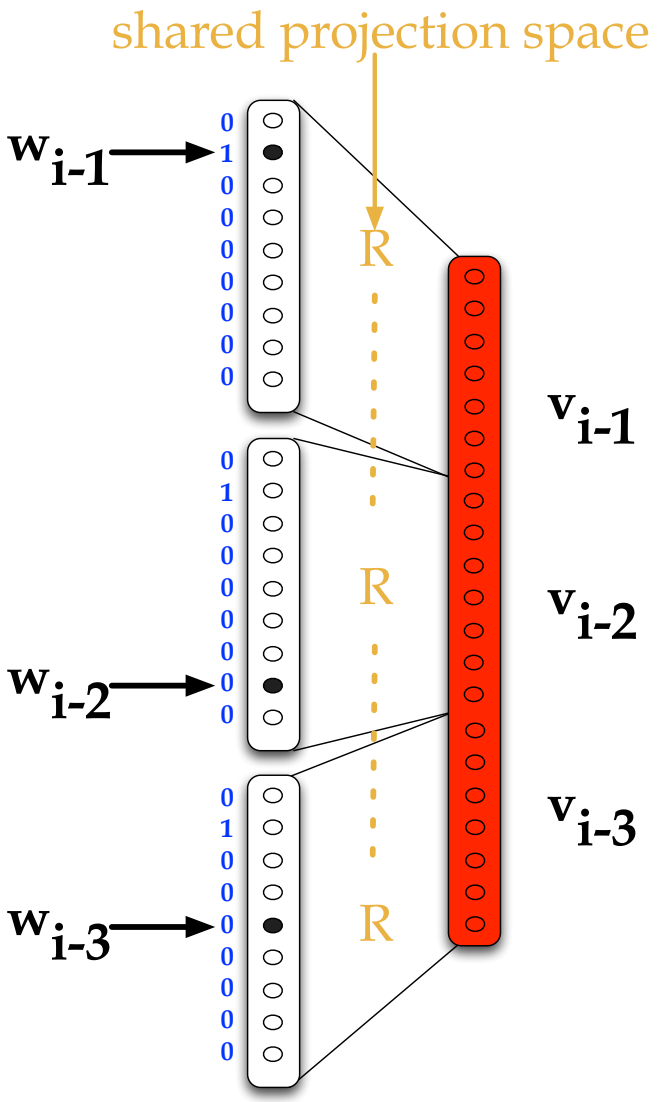
Project a word sequence in a continuous space

- Represent words as as 1-of-$|V|$ vectors
- 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



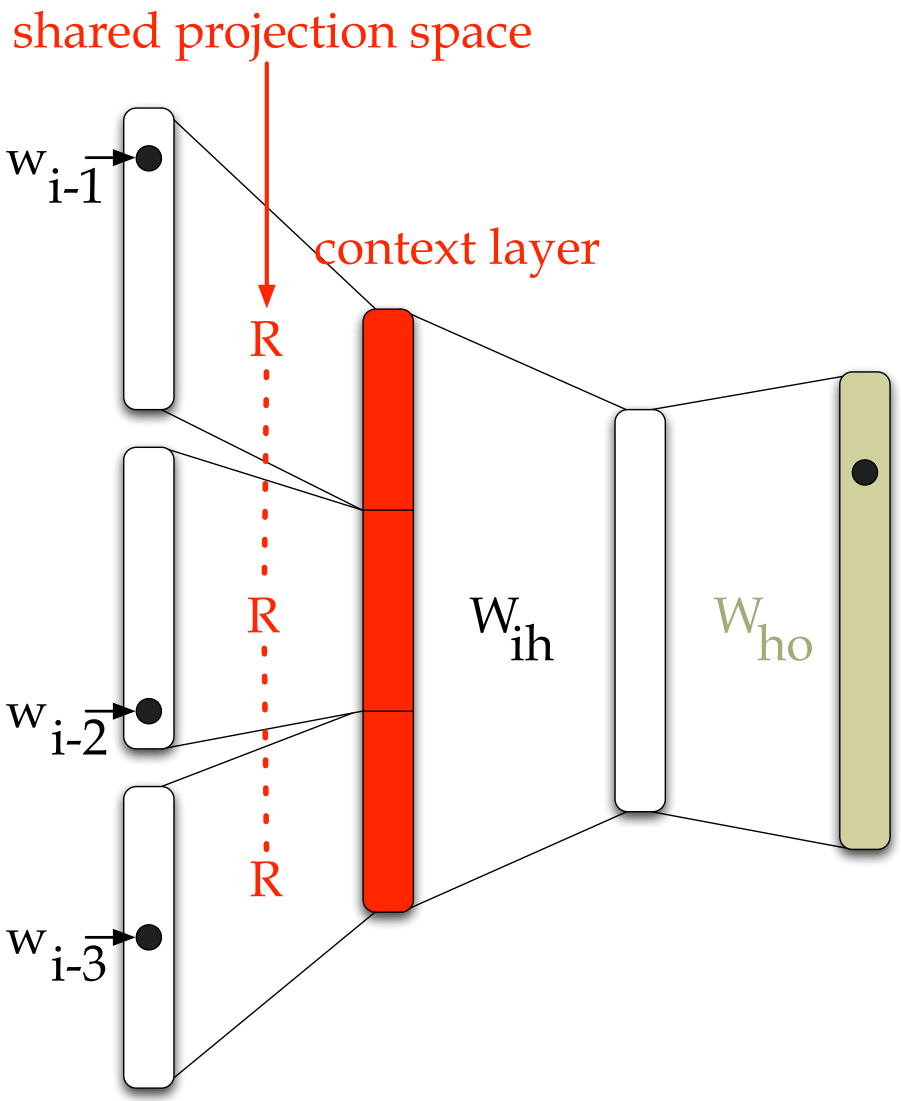
Project a word sequence in a continuous space

- Represent words as as 1-of-$|V|$ vectors
- 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



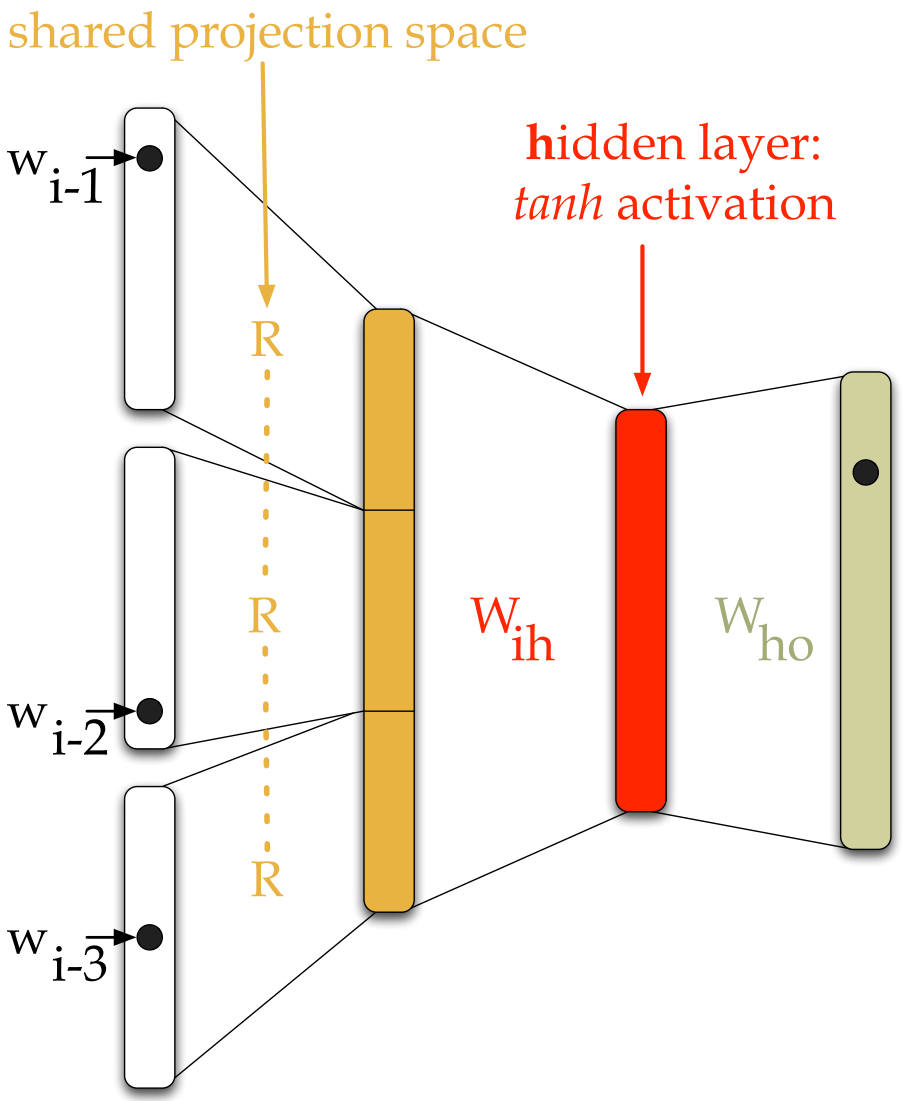
Estimate the *n*-gram probability

- 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 learned ($\mathbf{R}, \mathbf{W_{ih}}, \mathbf{W_{ho}}$).



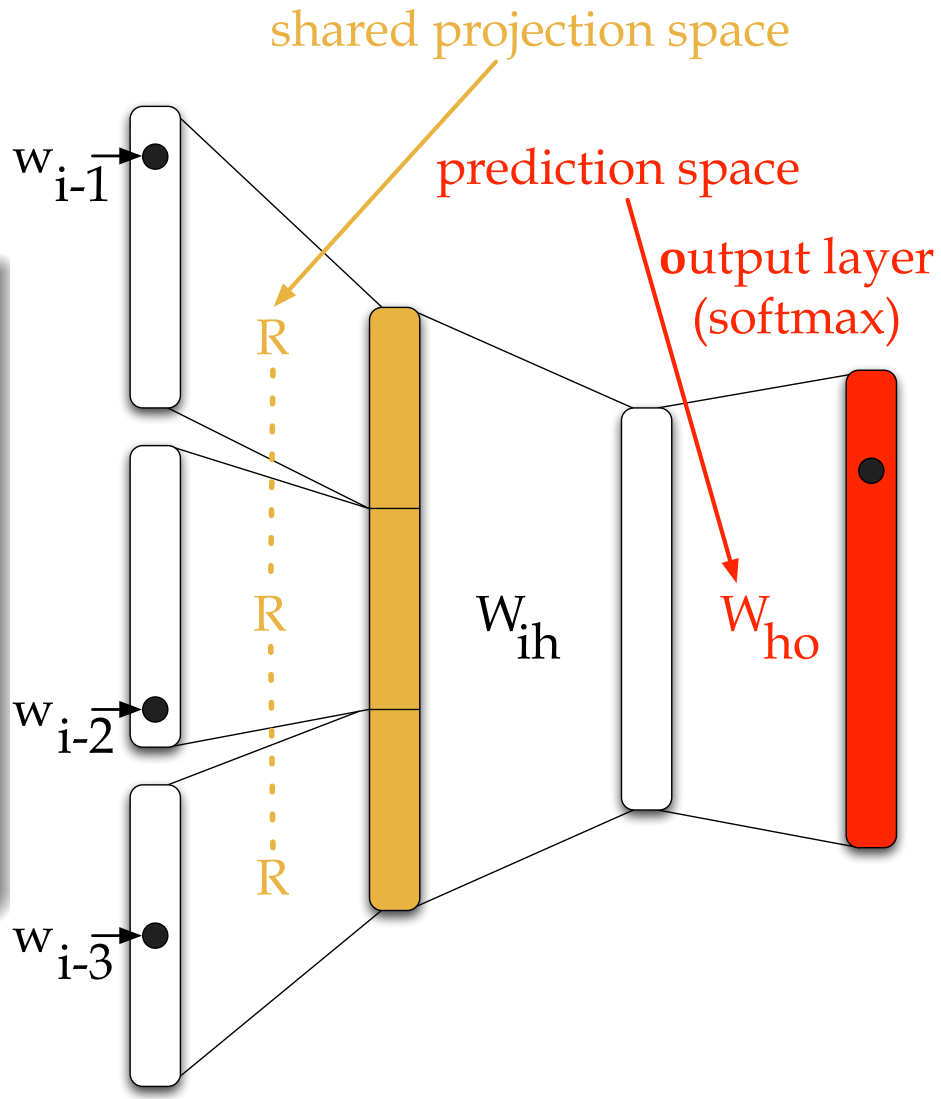
Estimate the *n*-gram probability

- 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 learned ($\mathbf{R}$, $\mathbf{W_{ih}}$, $\mathbf{W_{ho}}$).



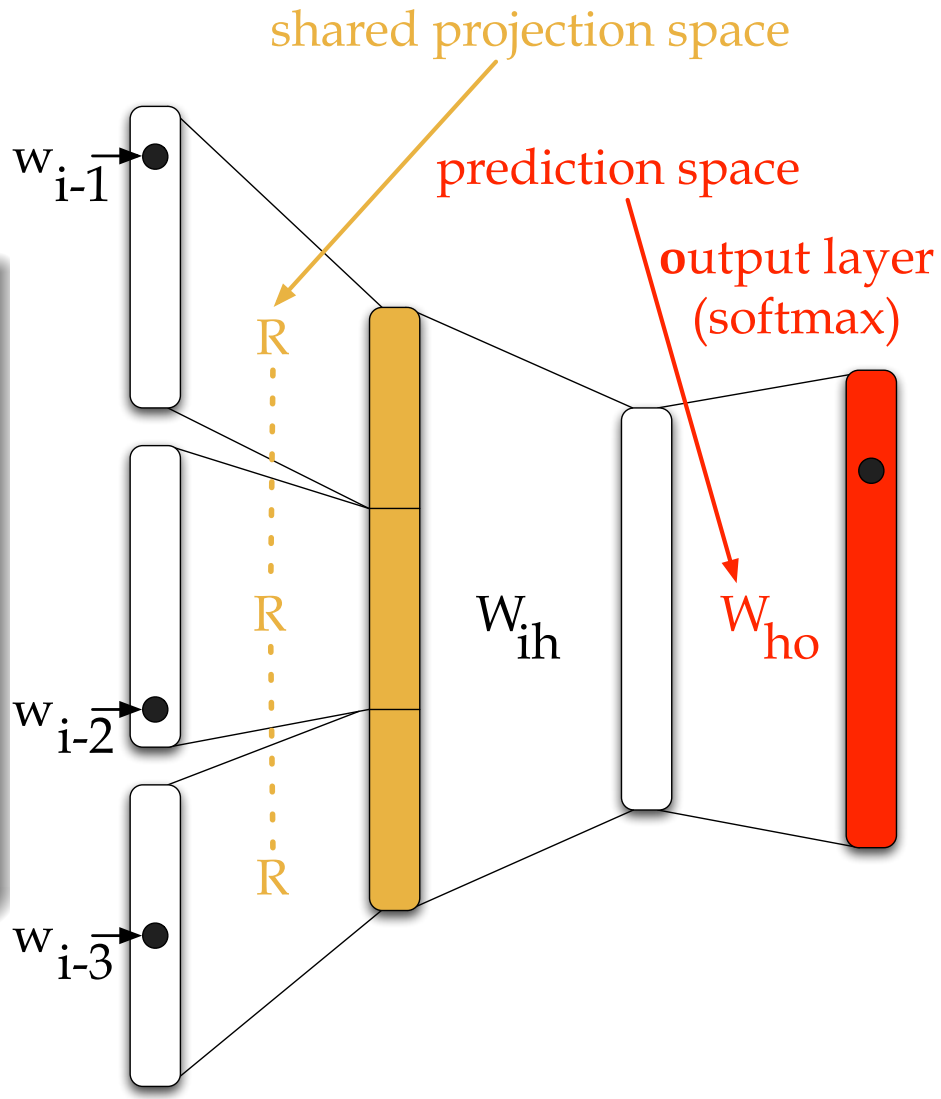
Estimate the *n*-gram probability

- 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 learned ($\mathbf{R}$, $\mathbf{W_{ih}}$, $\mathbf{W_{ho}}$).



Estimate the *n*-gram probability

- 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 learned ($\mathbf{R}, \mathbf{W_{ih}}, \mathbf{W_{ho}}$).



Early assessment

Key points

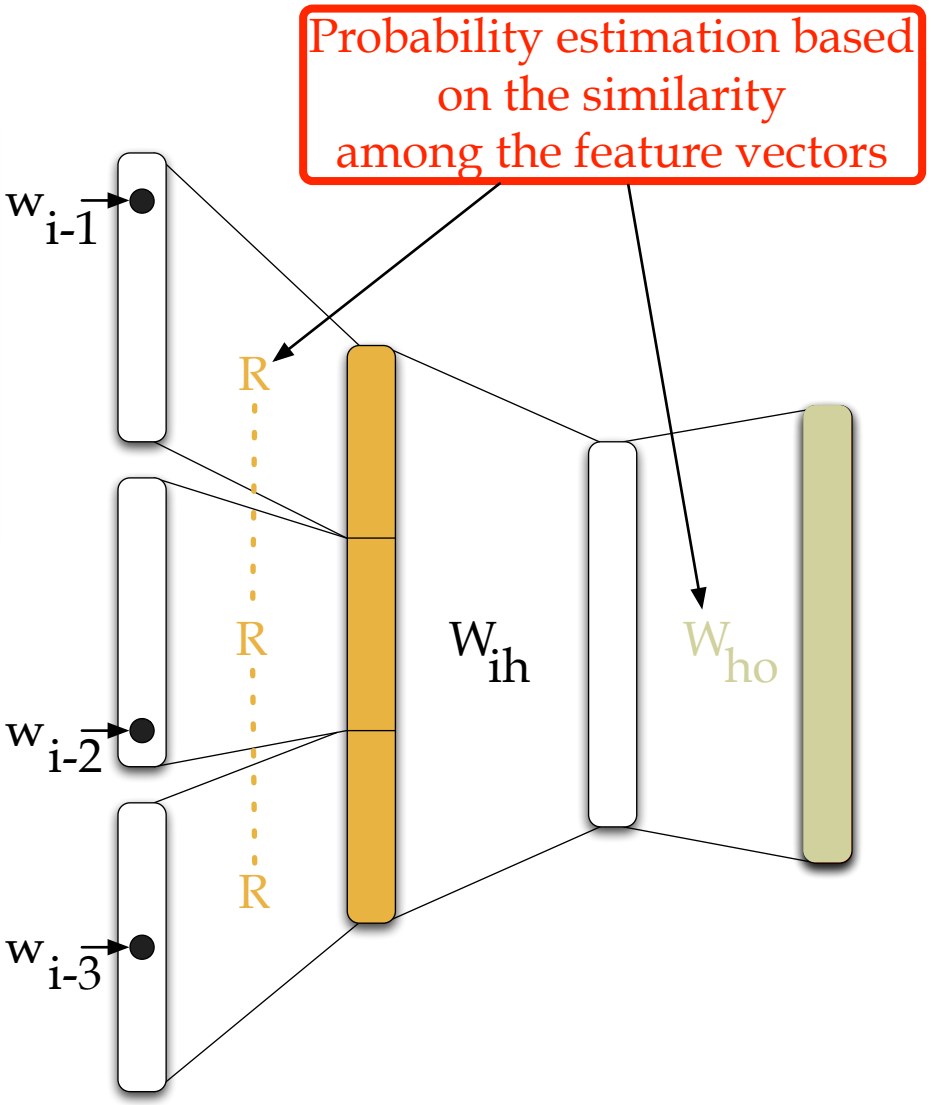
- The projection **in continuous spaces**

→ reduces the sparsity issues

- Learn simultaneously the projection and the prediction

In practice

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



Early assessment

Key points

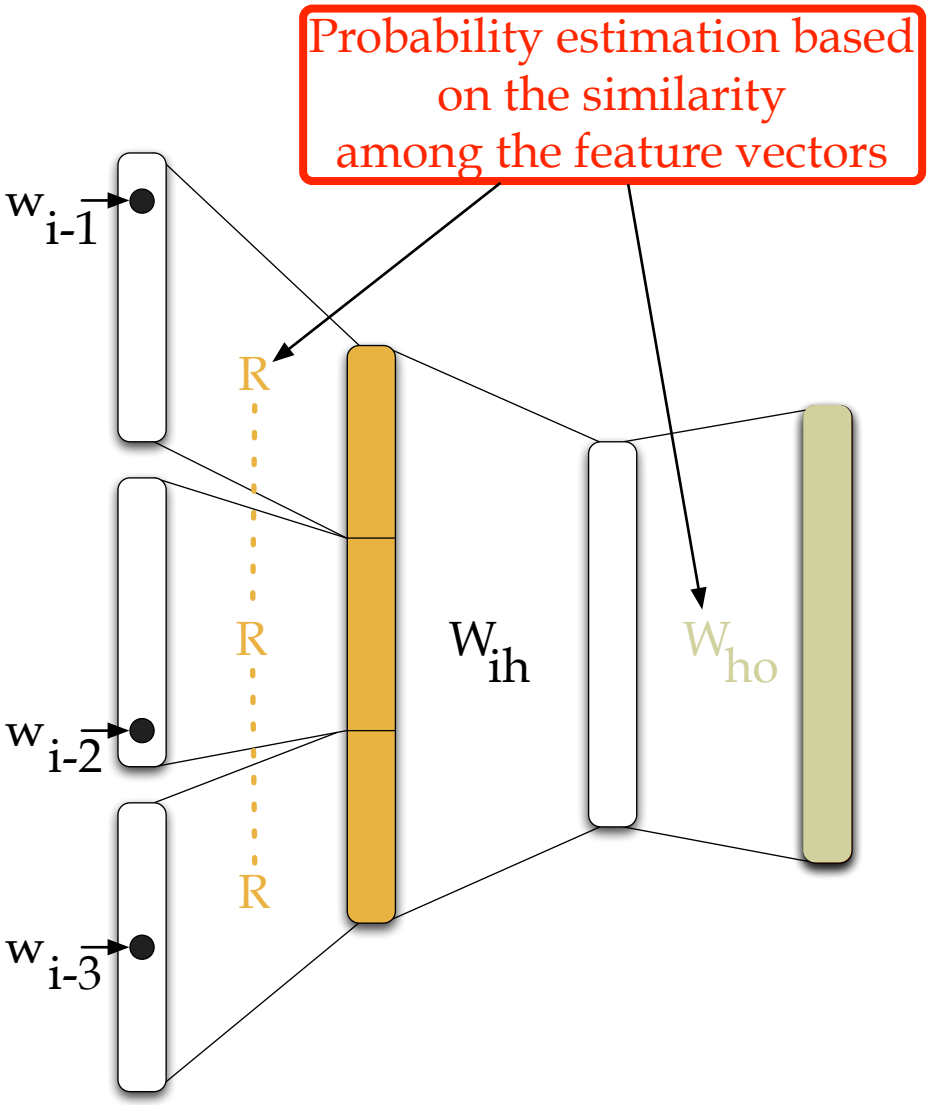
- The projection **in continuous spaces**

→ reduces the sparsity issues

- Learn simultaneously the projection and the prediction

In practice

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



Early assessment

Key points

- The projection **in continuous spaces**

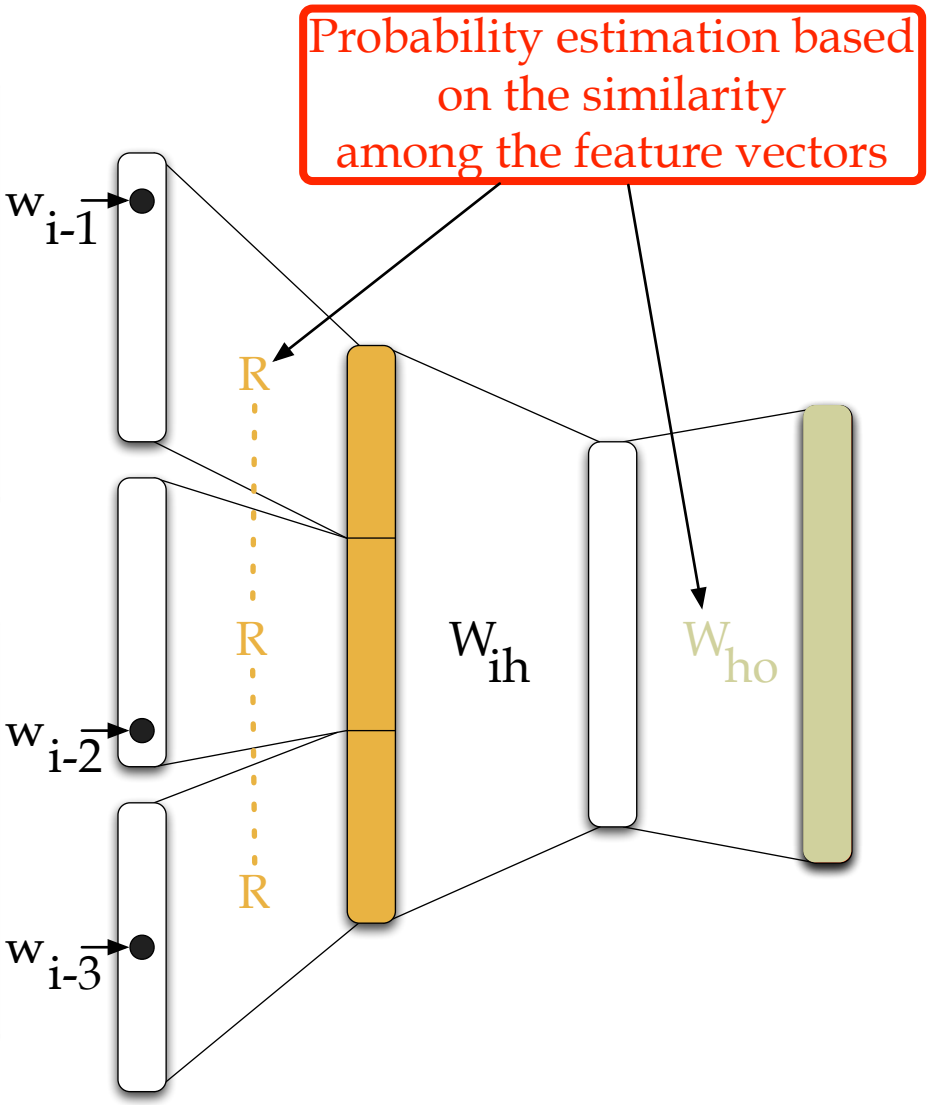
→ reduces the sparsity issues

- 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

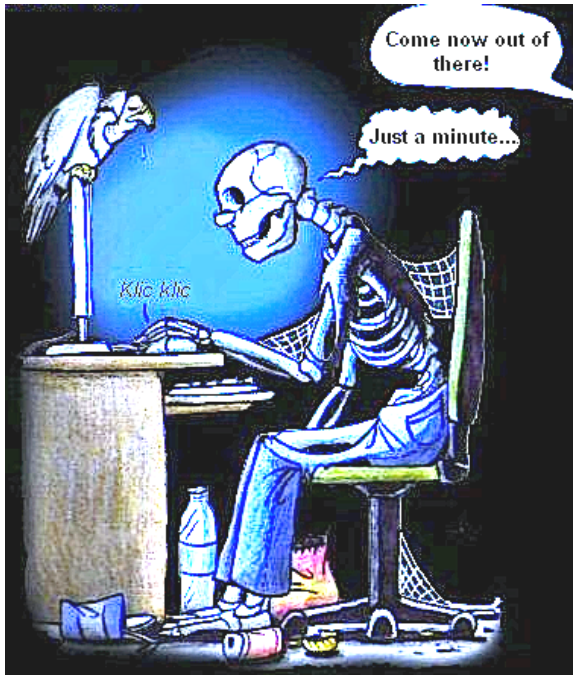
- Learn simultaneously the projection and the prediction

In practice

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

☹ **Learning and inference time**

With a large training set



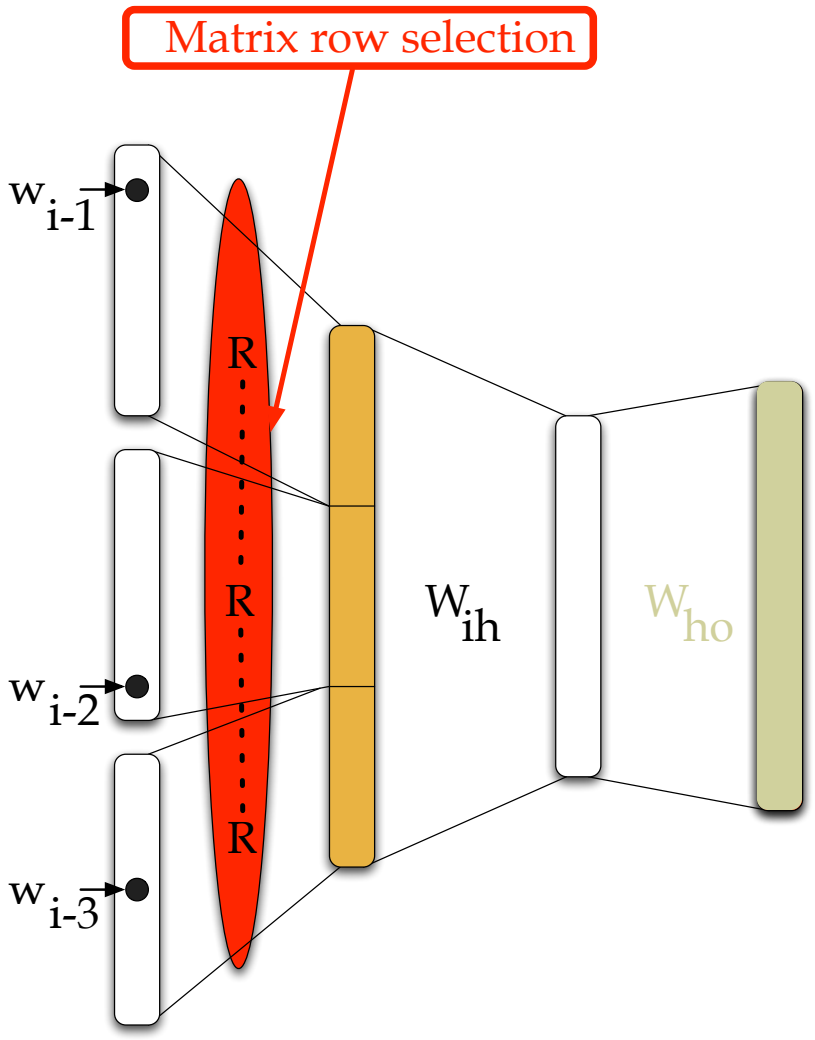
Why it is so long ? - Inference

Forward propagation of the history

- The projection: select a row in **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 does not drastically increase the complexity
- **The problem is the output vocabulary size**



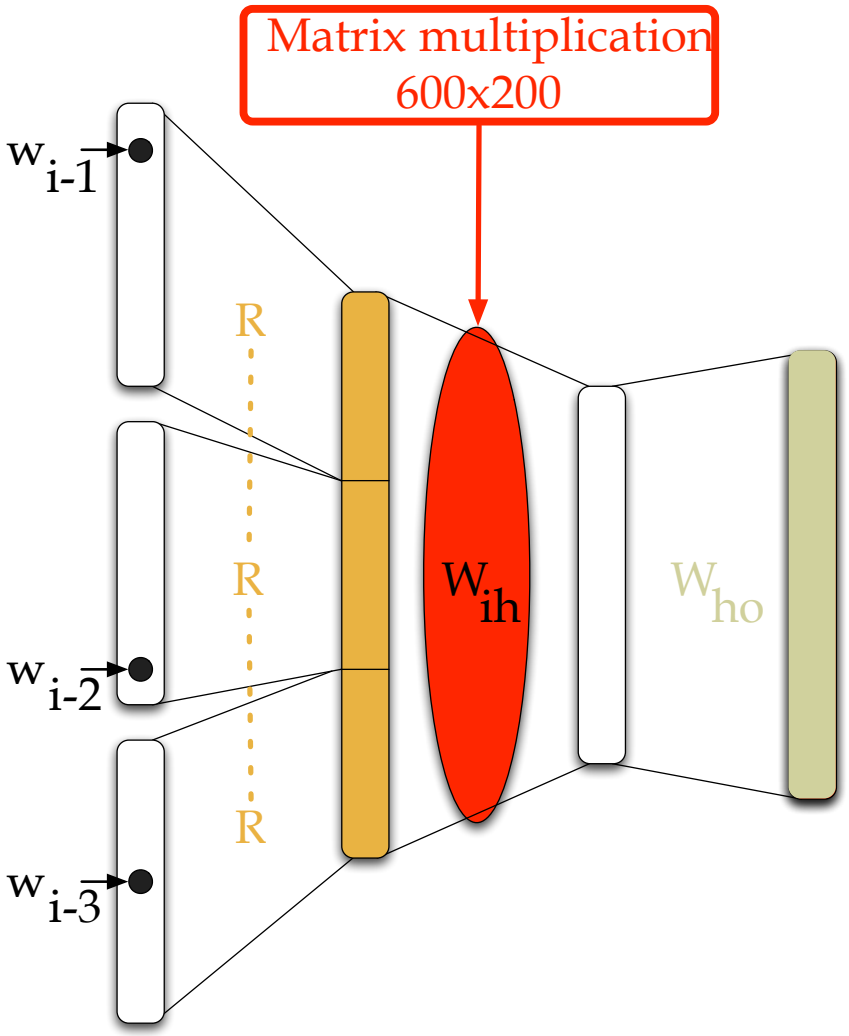
Why it is so long ? - Inference

Forward propagation of the history

- The projection: select a row in **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 does not drastically increase the complexity
- **The problem is the output vocabulary size**



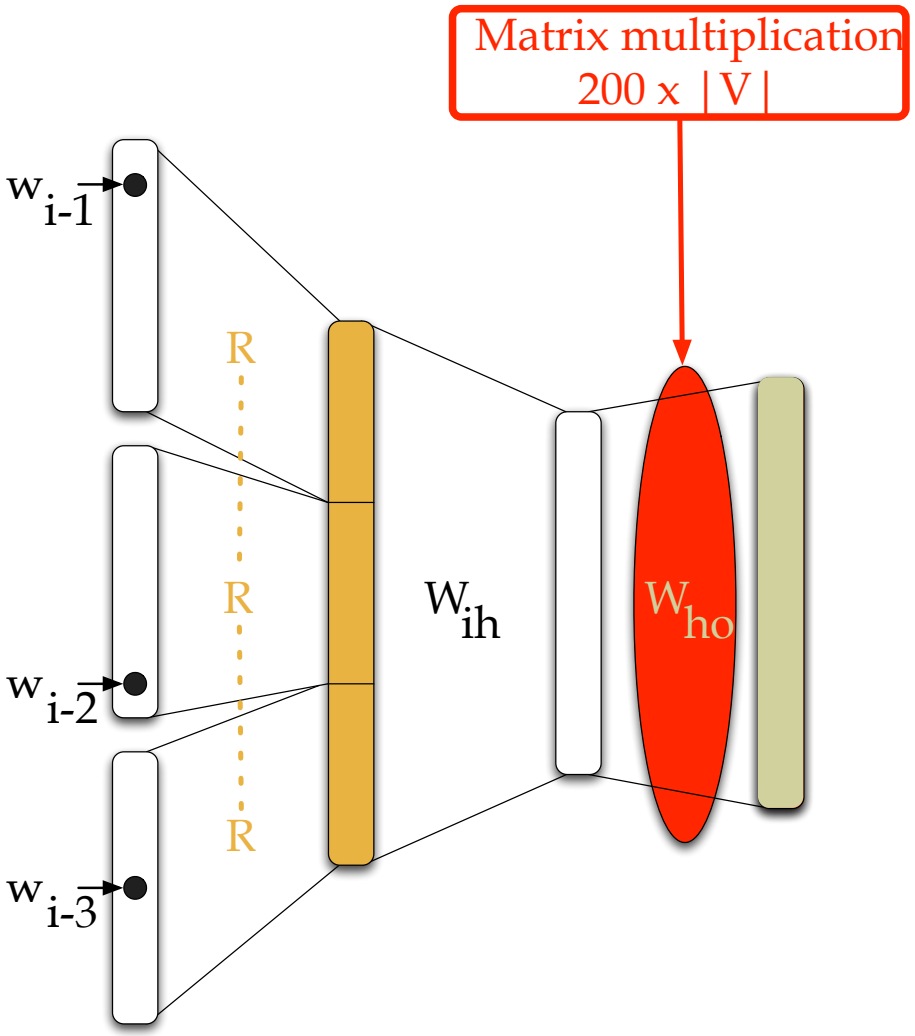
Why it is so long ? - Inference

Forward propagation of the history

- The projection: select a row in **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 does not drastically increase the complexity
- **The problem is the output vocabulary size**



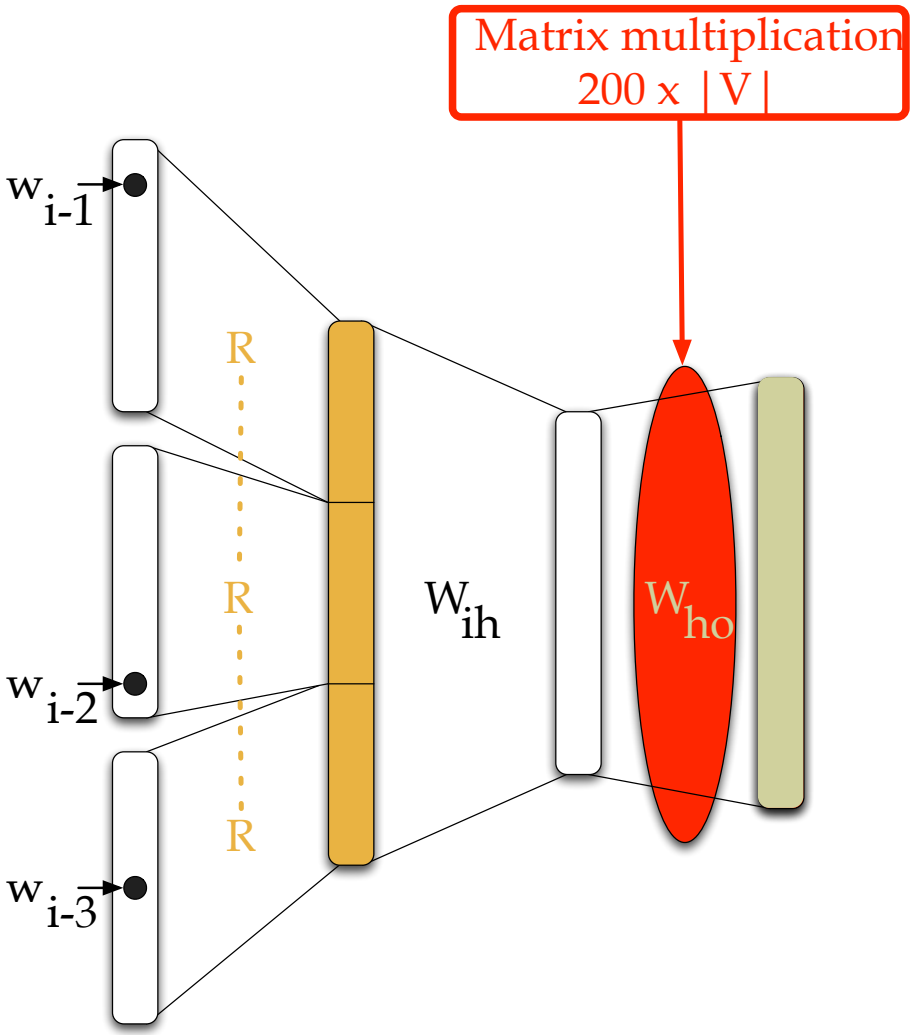
Why it is so long ? - Inference

Forward propagation of the history

- The projection: select a row in **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 does not drastically increase the complexity
- The problem is the output vocabulary size



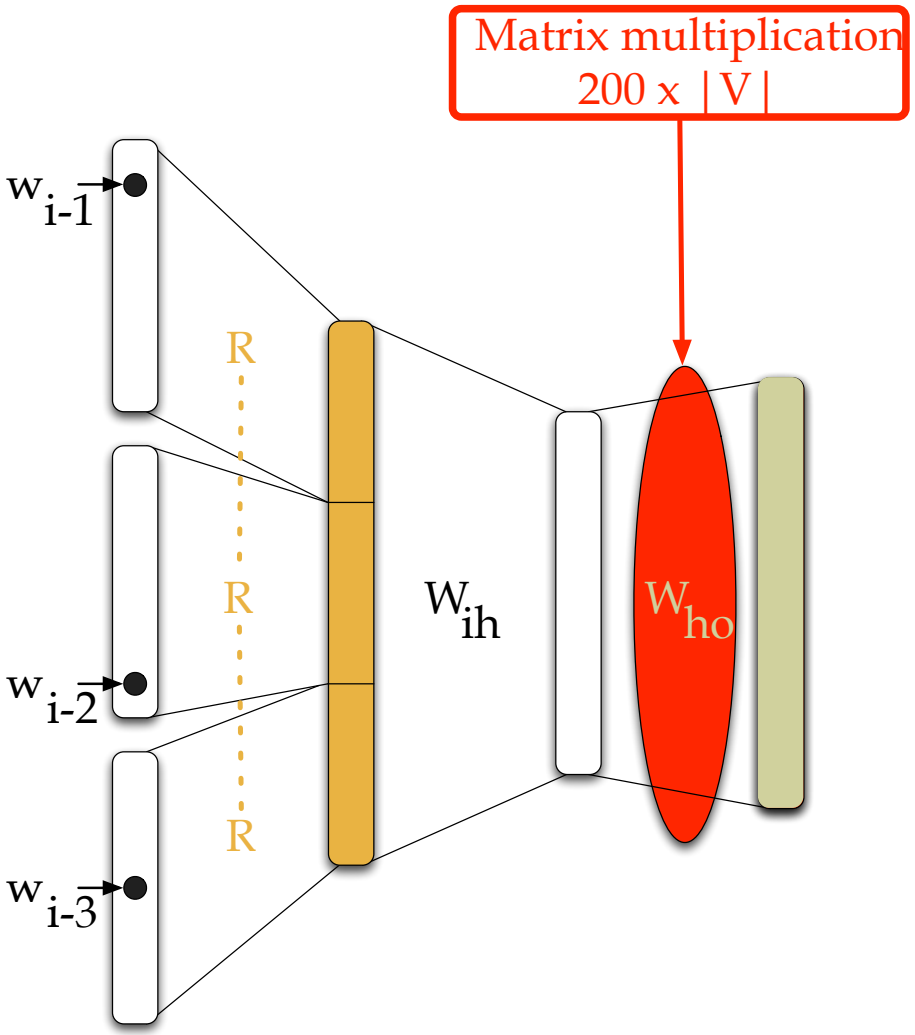
Why it is so long ? - Inference

Forward propagation of the history

- The projection: select a row in **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 does not drastically 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

Plan

Speeding up MaxEnt models

Main ideas as proposed in [Goodman, 2001]

- Instead of computing directly $P(w|h)$, make use of clustering of words into classes:
- $P(w|h) = P(w|c(w), h)P(c(w)|h)$
- Any classes can be used, but generalization may be better for classes for which it's easier to learn $P(c(w)|h)$

Example of reduction

- 10000 word vocabulary with 100 classes
- 2 normalizations over 100 outcomes
- 10000 $\rightarrow$ 200 (reduction by 50)

Hierarchical Probabilistic NNLM

Main ideas as proposed in [Morin and Bengio, 2005]

- Perform binary hierarchical clustering of the vocabulary
- Predict words as paths in this clustering tree

Details

- Clustering is constrained by WordNet semantic hierarchy
- Predicting next bit in hierarchy as $P(b|node, w_{t-1}, \ldots, w_{t-n+1})$

Results

- Brown corpus, 1M words, 10000 words vocabulary
- Speed-up but loss in perplexity as compared to a standard NNLM

Scalable Hierarchical Distributed LM

Main ideas as proposed in [Mnih and Hinton, 2008]

- Use automatic clustering instead of WordNet
- Implement as log-bilinear model
- One-to-many word class mapping

Results

- APNews dataset, 14M words, 18k vocabulary
- Perplexity improvements over *n*-gram model, similar performance to a non-hierarchical LBL
- No comparison with non-linear NNLMs used in STT

Plan

Structured OUtput Layer NNLM

Main ideas

- Trees are not binary
  - Multiple output layers with a softmax in each
- No clustering for frequent words
  - Compromise between speed and complexity
- Efficient clustering scheme
  - Word vectors in projection space are used for clustering

Task

- Improving state-of-the-art STT system that makes use of shortlist NNLMs
- Large vocabulary and the baseline *n*-gram LM trained on billions of words

Word clustering

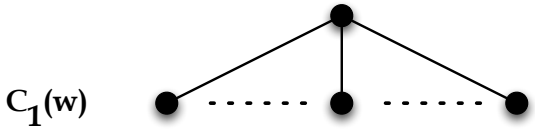
- Associate each frequent word with a single class $c_1(w)$
- Split other words in sub-classes ($c_2(w)$) and so on

Word clustering

- Associate each frequent word with a single class $c_1(w)$
- Split other words in sub-classes ($c_2(w)$) and so on

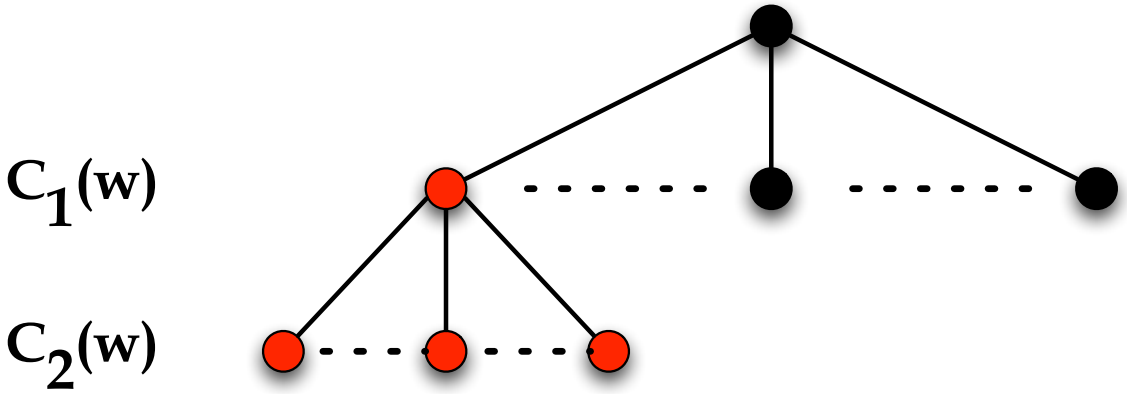
Word clustering

- Associate each frequent word with a single class $c_1(w)$
- Split other words in sub-classes ($c_2(w)$) and so on



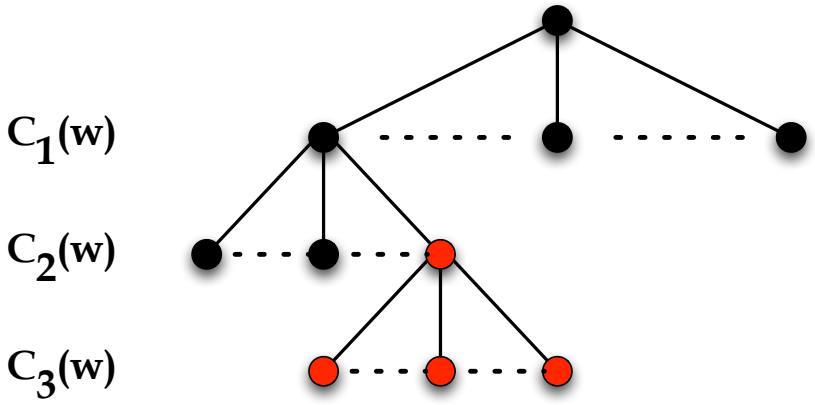
Word clustering

- Associate each frequent word with a single class $c_1(w)$
- Split other words in sub-classes ($c_2(w)$) and so on

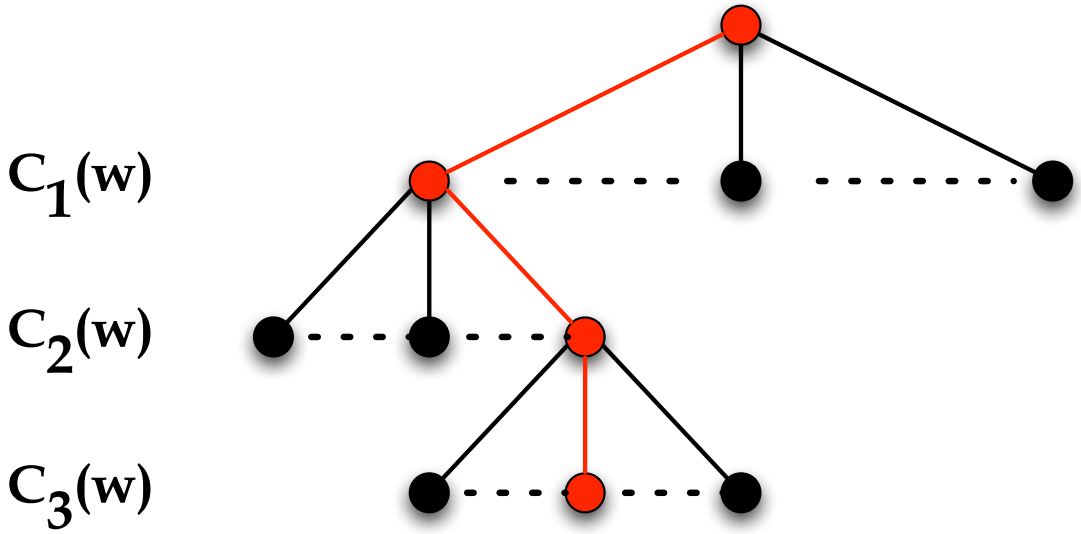


Word clustering

- Associate each frequent word with a single class $c_1(w)$
- Split other words in sub-classes ($c_2(w)$) and so on



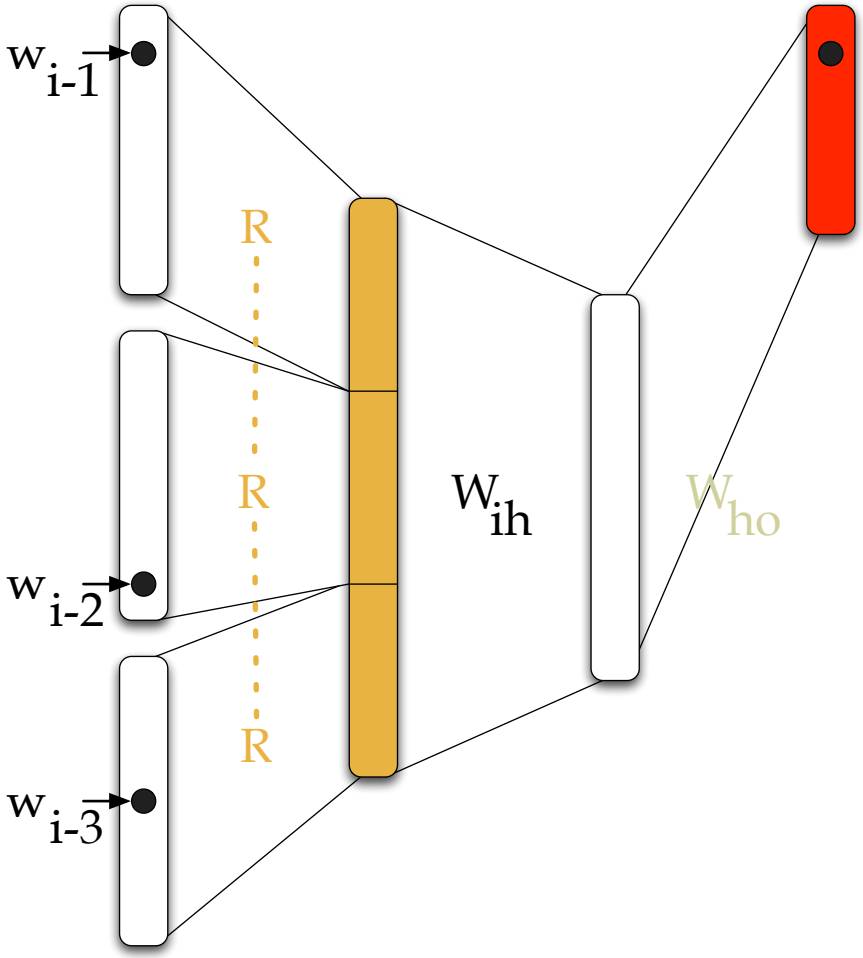
Word probability

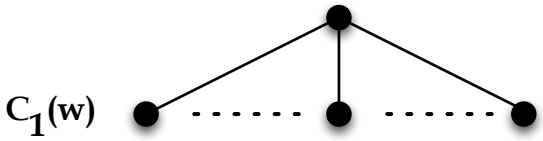


$$P(w_i|h) = P(c_1(w_i)|h) \prod_{d=2}^{D} P(c_d(w_i)|h, c_{1:d-1})$$

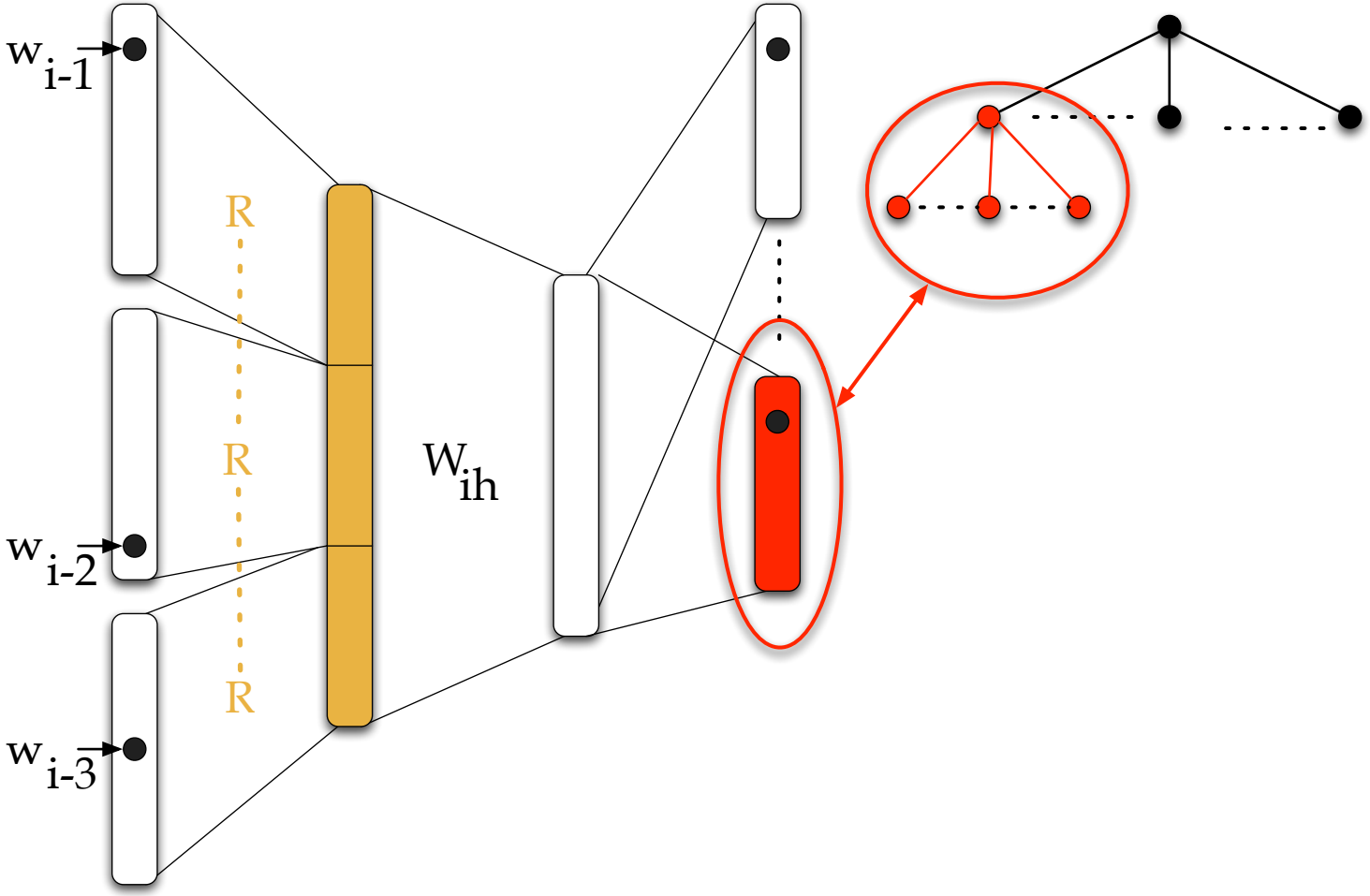
- $c_{1:D}(w_i) = c_1, \ldots, c_D$: path for the word w_i in the clustering tree,
- D : depth of the tree,
- $c_d(w_i)$: (sub-)class,
- $c_D(w_i)$: leaf

The SOUL language model





The SOUL language model



Training algorithm

Step 1:

Train a standard NNLM model with the shortlist as an output
(3 epochs and a shortlist of 8k words)

Step 2:

Reduce the dimension of the context space using with PCA
(final dimension is 10 in our experiments)

Step 3:

Perform a recursive K-means word clustering based on the distributed representation induced by the continuous space
(except for words in the shortlist)

Step 4:

Train the whole model

Training algorithm

Step 1:

Train a standard NNLM model with the shortlist as an output
(3 epochs and a shortlist of 8k words)

Step 2:

Reduce the dimension of the context space using with PCA
(final dimension is 10 in our experiments)

Step 3:

Perform a recursive K-means word clustering based on the distributed representation induced by the continuous space
(except for words in the shortlist)

Step 4:

Train the whole model

Training algorithm

Step 1:

Train a standard NNLM model with the shortlist as an output
(3 epochs and a shortlist of 8k words)

Step 2:

Reduce the dimension of the context space using with PCA
(final dimension is 10 in our experiments)

Step 3:

Perform a recursive K-means word clustering based on the distributed representation induced by the continuous space
(except for words in the shortlist)

Step 4:

Train the whole model

Training algorithm

Step 1:

Train a standard NNLM model with the shortlist as an output
(3 epochs and a shortlist of 8k words)

Step 2:

Reduce the dimension of the context space using with PCA
(final dimension is 10 in our experiments)

Step 3:

Perform a recursive K-means word clustering based on the distributed representation induced by the continuous space
(except for words in the shortlist)

Step 4:

Train the whole model

STT results with SOUL NNLMs

Mandarin GALE task

- LIMSI Mandarin STT system
  - 56k vocabulary
  - Baseline LM trained on 3.2 billion words
- 4 NNLMs trained on 25M words after resampling

model	**ppx**	**CER**	
	dev09	*dev09s*	*eval09*
Baseline 4-gram	211	9.8%	8.9%
+4-gram NNLM 8k	187	9.5%	8.6%
+4-gram NNLM 12k	185	9.4%	8.6%
+4-gram SOUL NNLM	180	9.3%	8.5%
+6-gram NNLM 8k	177	9.4%	8.5%
+6-gram NNLM 12k	172	9.3%	8.5%
+6-gram SOUL NNLM	**162**	**9.1%**	**8.3%**

Conclusion

- Neural network and class-based language models combined together
- SOUL LM is able to deal with vocabularies of arbitrary sizes
- Speech recognition improvements are achieved on a large-scale task and over challenging baselines
- SOUL LM improves better for longer contexts

Bengio, Y., Ducharme, R., and Vincent, P. (2001).
A neural probabilistic language model.
Neural Information Processing Systems, 13:933–938.

Goodman, J. (2001).
Classes for fast maximum entropy training.
In *Proc. of ICASSP'01*.

Mnih, A. and Hinton, G. (2008).
A scalable hierarchical distributed language model.
In *Neural Information Processing Systems*, volume 21, pages 1081–1088.

Morin, F. and Bengio, Y. (2005).
Hierarchical probabilistic neural network language model.
In *Proc. of AISTATS'05*, pages 246–252.

Schwenk, H. and Gauvain, J.-L. (2002).
Connectionist language modeling for large vocabulary continuous speech recognition.
In *Proc. of ICASSP'02*, pages 765–768.