# Continuous Space (or neural network) language model

Alexandre Allauzen

# Outline

1. Statistical language modeling : the *n*-gram model
2. Neural network language model
3. SOUL language model

# Plan

# The goal of statistical language model (LM)

Aims to estimate a probability for all possible word sequences $W$ build on a finite vocabulary $V$.

**The *n*-gram assumption**

A word can be predicted by its truncated **history** : $n-1$ previous words.

**$n = 4$, the *four*-gram**

$$P(w_i|w_1, \cdots, w_{i-2}, w_{i-1}) = P(w_i|w_{i-3}, w_{i-2}, w_{i-1}) \quad (1)$$

**Applications**

Automatic speech recognition, Machine translation, ...

# The goal of statistical language model (LM)

Aims to estimate a probability for all possible word sequences $W$ build on a finite vocabulary $V$.

### The *n*-gram assumption

A word can be predicted by its truncated **history** : $n-1$ previous words.

### $n = 4$, the *four*-gram

$$P(w_i|w_1, \cdots, w_{i-2}, w_{i-1}) = P(w_i|w_{i-3}, w_{i-2}, w_{i-1}) \quad (1)$$

### Applications

Automatic speech recognition, Machine translation, ...

# The goal of statistical language model (LM)

Aims to estimate a probability for all possible word sequences $W$ build on a finite vocabulary $V$.

### The *n*-gram assumption

A word can be predicted by its truncated **history** : $n-1$ previous words.

### $n = 4$, the *four*-gram

$$P(w_i|w_1, \cdots, w_{i-2}, w_{i-1}) = P(w_i|w_{i-3}, w_{i-2}, w_{i-1}) \quad (1)$$

### Applications

Automatic speech recognition, Machine translation, ...

# The goal of statistical language model (LM)

Aims to estimate a probability for all possible word sequences $W$ build on a finite vocabulary $V$.

### The *n*-gram assumption

A word can be predicted by its truncated **history** : $n-1$ previous words.

### $n = 4$, the *four*-gram

$$P(w_i|w_1, \cdots, w_{i-2}, w_{i-1}) = P(w_i|w_{i-3}, w_{i-2}, w_{i-1}) \quad (1)$$

### Applications

Automatic speech recognition, Machine translation, ...

# The *n*-gram language model in practice

A *n*-gram language model is a set of discrete distribution, one per history.

### Training and inference

- *n* ranges from 2 to 4
- One parameter for each observed *n*-gram
- \+ smoothing parameters for unseen *n*-grams
- Inference is straightforward

### Data sparsity issue

English 4-gram LM for WMT:

- training corpus: 6 billions of words
- number of parameters: more than 2,4 billions
- Most of the *n*-grams appear only once in the training data

# The lack of generalization

## A flat vocabulary

- Each word is only a possible outcome of a discrete random variable,
- an index in the vocabulary.

What is the relationship between two words ?

## An illustration

A training sentence:

LAST SUNDAY EVENING I FOUND A PLACE TO EAT IN **PARIS**.

What can be infered for KARLSRUHE in the following sentence ?

LAST SUNDAY EVENING I FOUND A PLACE TO EAT IN **KARLSRUHE** .

# The lack of generalization

## A flat vocabulary

- Each word is only a possible outcome of a discrete random variable,
- an index in the vocabulary.

What is the relationship between two words ?

## An illustration

A training sentence:

LAST SUNDAY EVENING I FOUND A PLACE TO EAT IN **PARIS**.

What can be infered for KARLSRUHE in the following sentence ?

LAST SUNDAY EVENING I FOUND A PLACE TO EAT IN **KARLSRUHE** .

# The lack of generalization

## A flat vocabulary

- Each word is only a possible outcome of a discrete random variable,
- an index in the vocabulary.

What is the relationship between two words ?

## An illustration

A training sentence:

LAST SUNDAY EVENING I FOUND A PLACE TO EAT IN **PARIS**.

What can be infered for KARLSRUHE in the following sentence ?

LAST SUNDAY EVENING I FOUND A PLACE TO EAT IN **KARLSRUHE** .

# The lack of generalization

## A flat vocabulary

- Each word is only a possible outcome of a discrete random variable,
- an index in the vocabulary.

What is the relationship between two words ?

## An illustration

A training sentence:

LAST SUNDAY EVENING I FOUND A PLACE TO EAT IN **PARIS**.

What can be infered for KARLSRUHE in the following sentence ?

LAST SUNDAY EVENING I FOUND A PLACE TO EAT IN **KARLSRUHE** .

# Plan

# Estimate *n*-gram probabilities in a continuous space

Introduced in [Bengio et al., 2001, Bengio et al., 2003] and applied to speech recognition and machine translation in [Schwenk and Gauvain, 2002].

## 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
  - a small change in the features will induce a small change in the probability
  - Remember: PARIS and KARLSRUHE

# Estimate *n*-gram probabilities in a continuous space

Introduced in [Bengio et al., 2001, Bengio et al., 2003] and applied to speech recognition and machine translation in [Schwenk and Gauvain, 2002].

## In a nutshell

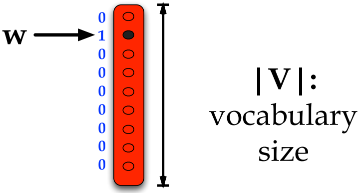
1. associate each word with a continuous feature vector
2. express the probability function of a word sequence in terms of the feature vectors of these words
3. learn simultaneously the feature vectors and the parameters of that probability function.

## 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
  - a small change in the features will induce a small change in the probability
  - Remember: PARIS and KARLSRUHE

# Project a word sequence in a continuous space

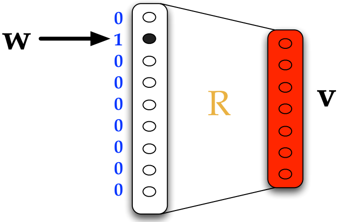
- The vocabulary is a neuron layer.
- Project the word in the continuous space: add a second layer fully connected.
- For a 4-gram, the history is a sequence of 3 words.
- Merge these three vectors to derive a single vector for the history




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

# Project a word sequence in a continuous space

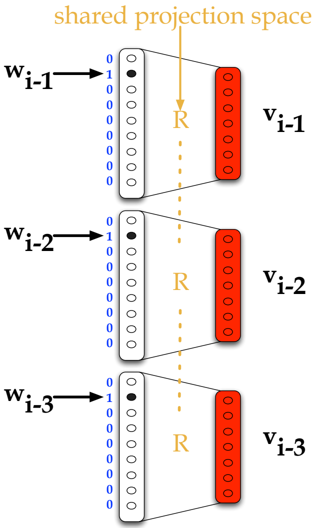
- The vocabulary is a neuron layer.
- Project the word in the continuous space: add a second layer fully connected.
- For a 4-gram, the history is a sequence of 3 words.
- Merge these three vectors to derive a single vector for the history




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

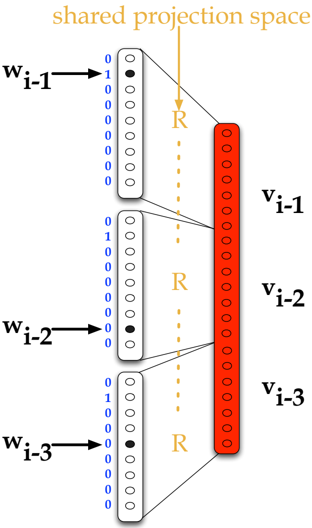
# Project a word sequence in a continuous space

- 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



# Project a word sequence in a continuous space

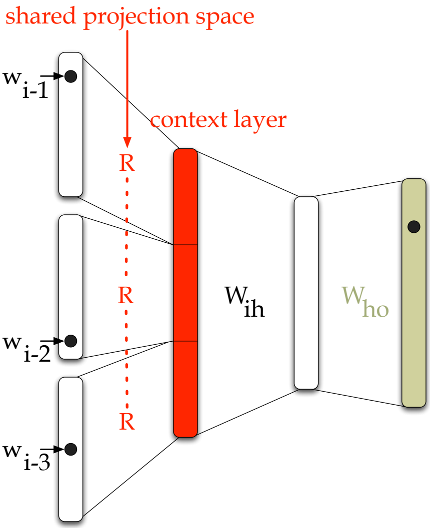
- The vocabulary is a neuron layer.
- Project the word in the continuous space: add a second layer fully connected.
- For a 4-gram, the history is a sequence of 3 words.
- Merge these three vectors to derive a single vector for the history



# Estimate the *n*-gram probability

## The program

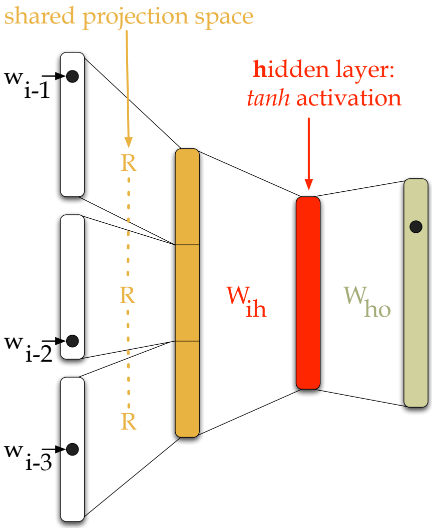
- Given the history expressed as a feature vector.
- Create a feature vector for the word to be predicted in the **prediction space**.
- Estimate probabilities for all words given the history.
- All the parameters must be learnt (**R**, $\mathbf{W_{ih}}$, $\mathbf{W_{ho}}$).



# Estimate the *n*-gram probability

## The program

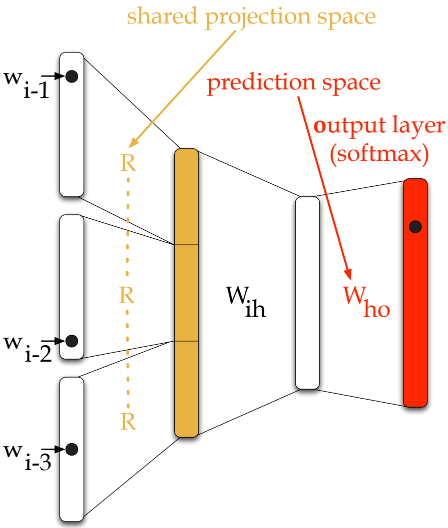
- Given the history expressed as a feature vector.
- Create a feature vector for the word to be predicted in the **prediction space**.
- Estimate probabilities for all words given the history.
- All the parameters must be learnt (**R**, $\mathbf{W_{ih}}$, $\mathbf{W_{ho}}$).



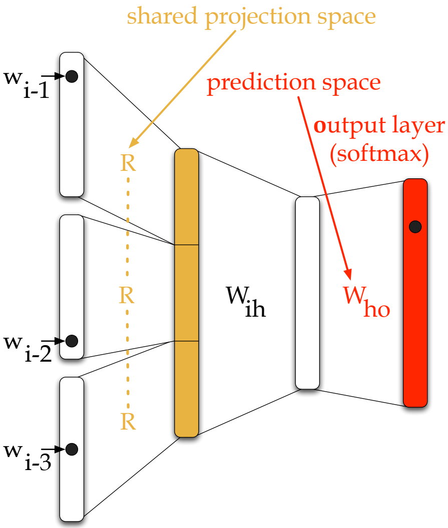
# Estimate the *n*-gram probability

### 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 (**R**, $\mathbf{W_{ih}}$, $\mathbf{W_{ho}}$).



# 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

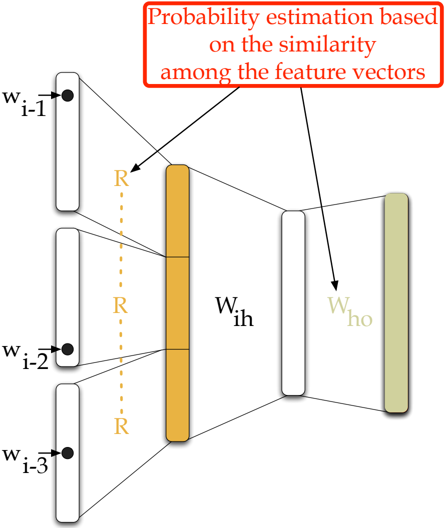
- The projection **in continuous spaces**

$\rightarrow$ 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

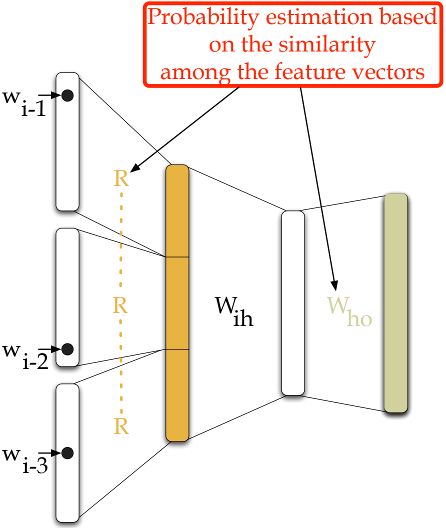
- The projection **in continuous spaces**

$\rightarrow$ 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**

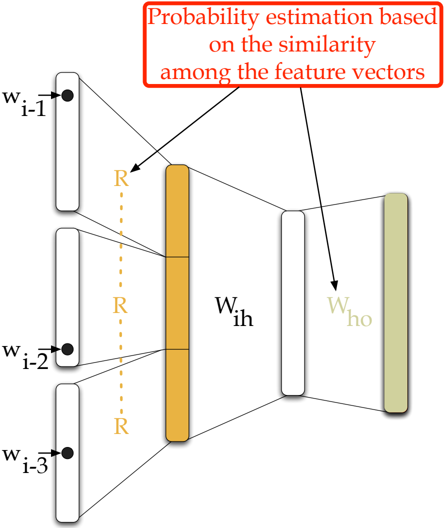
→ reduces the sparsity issues

- 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

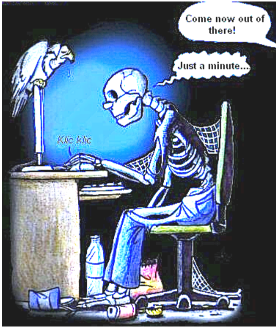
- Learn simultaneously the projection and the prediction

## In practice

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

☹ **Learning and inference time**

With a large training set



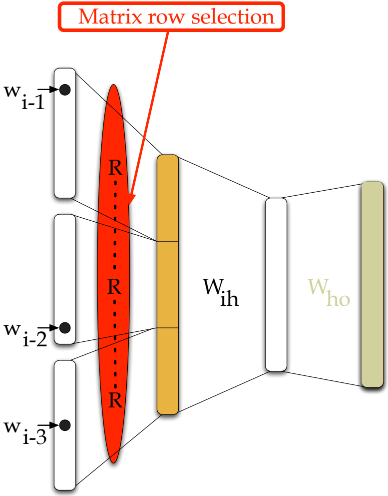
# Why it is so long ? - Inference

## Forward propagation of the history

- The projection: select a row in **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.**



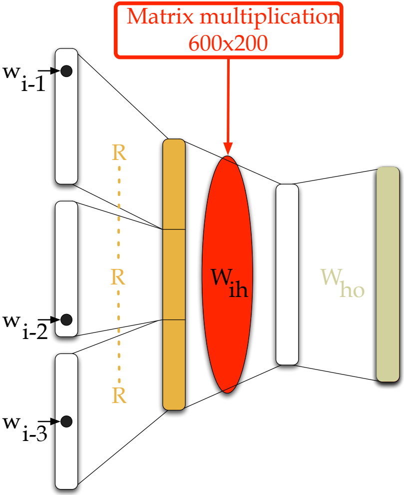
# Why it is so long ? - Inference

## 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 of *n* does not increase the complexity.
- **The problem is the output vocabulary size.**



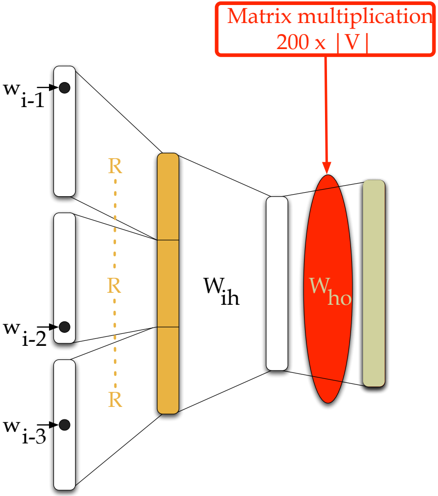
# Why it is so long ? - Inference

## 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 of *n* does not increase the complexity.
- **The problem is the output vocabulary size.**



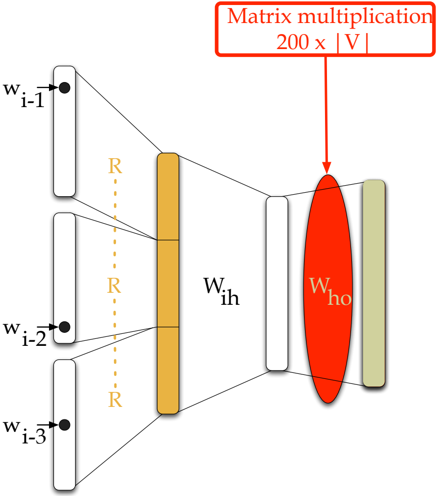
# Why it is so long ? - Inference

## 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 of *n* does not increase the complexity.
- **The problem is the output vocabulary size.**



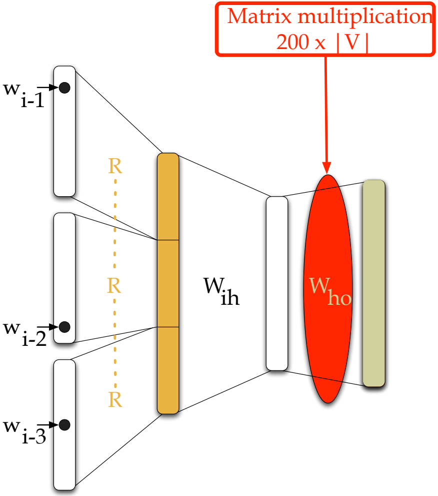
# Why it is so long ? - Inference

## 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 of *n* does not increase the complexity.
- **The problem is the output vocabulary size.**



# Why it is so long ? - Training

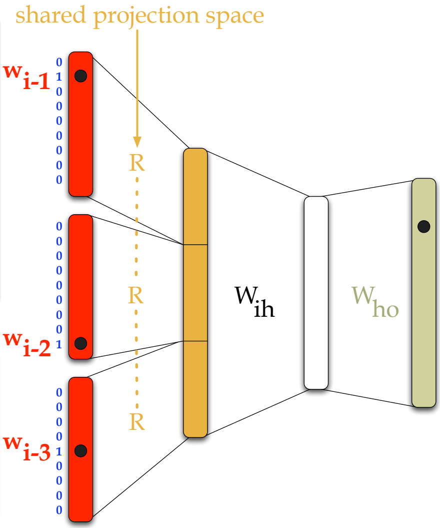
## Stochastic Gradient Ascent using back-propagation

One epoch of training = for each training example:

- Present the example (1 *n*-gram)
- Forward propagation
- Back-propagation of the error to update each matrix.

## In practice

- A billion of training *n*-grams
  ⇒ a billion of inferences
- Many epochs ⇒ many billions of inferences.



# Why it is so long ? - Training

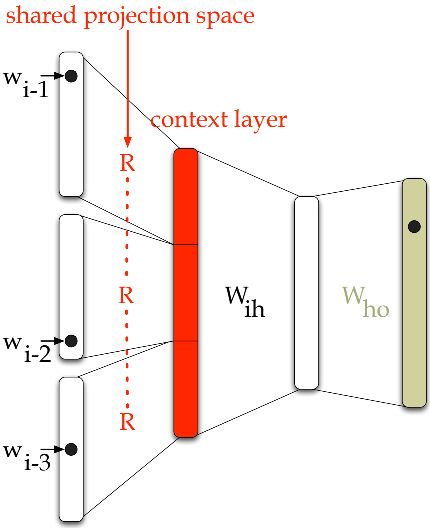
### Stochastic Gradient Ascent using back-propagation

One epoch of training = for each training example:

- Present the example (1 *n*-gram)
- Forward propagation
- Back-propagation of the error to update each matrix.

### In practice

- A billion of training *n*-grams
  $\Rightarrow$ a billion of inferences
- Many epochs $\Rightarrow$ many billions of inferences.



# Why it is so long ? - Training

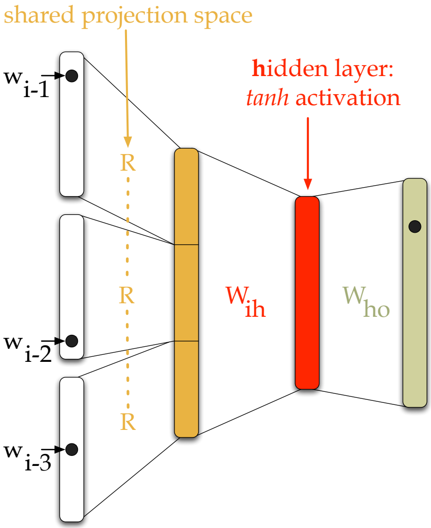
### Stochastic Gradient Ascent using back-propagation

One epoch of training = for each training example:

- Present the example (1 *n*-gram)
- Forward propagation
- Back-propagation of the error to update each matrix.

### In practice

- A billion of training *n*-grams
  $\Rightarrow$ a billion of inferences
- Many epochs $\Rightarrow$ many billions of inferences.



# Why it is so long ? - Training

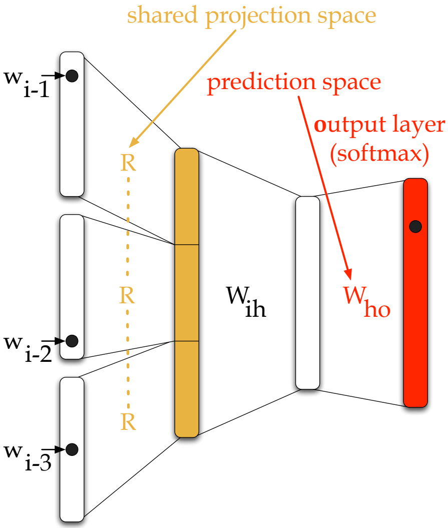
## Stochastic Gradient Ascent using back-propagation

One epoch of training = for each training example:

- Present the example (1 *n*-gram)
- Forward propagation
- Back-propagation of the error to update each matrix.

## In practice

- A billion of training *n*-grams
  $\Rightarrow$ a billion of inferences
- Many epochs $\Rightarrow$ many billions of inferences.



# Why it is so long ? - Training

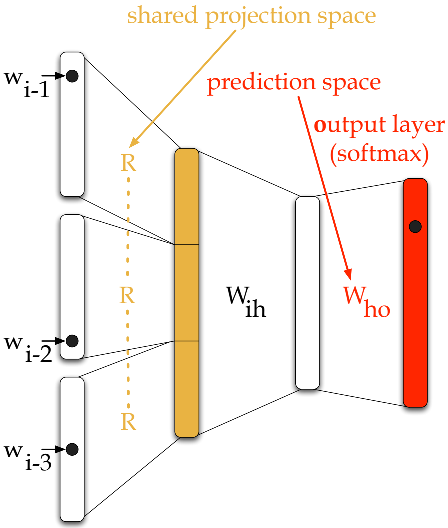
## Stochastic Gradient Ascent using back-propagation

One epoch of training = for each training example:

- Present the example (1 *n*-gram)
- Forward propagation
- Back-propagation of the error to update each matrix.

## In practice

- A billion of training *n*-grams
  $\Rightarrow$ a billion of inferences
- Many epochs $\Rightarrow$ many billions of inferences.



# 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.

# 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

# A kind of class-based language model

## Main ideas

As proposed by [Mnih and Hinton, 2008]:

- Represent the vocabulary as a clustering tree [Brown et al., 1992].
- Predict the path in this clustering tree.

## Word clustering

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

# A kind of class-based language model

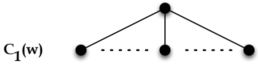
## Main ideas

As proposed by [Mnih and Hinton, 2008]:

- Represent the vocabulary as a clustering tree [Brown et al., 1992].
- Predict the path in this clustering tree.

## Word clustering

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



# A kind of class-based language model

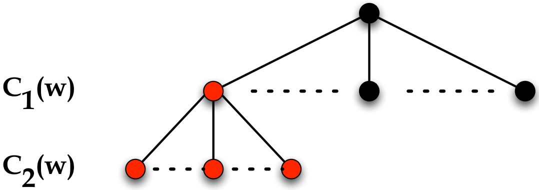
## Main ideas

As proposed by [Mnih and Hinton, 2008]:

- Represent the vocabulary as a clustering tree [Brown et al., 1992].
- Predict the path in this clustering tree.

## Word clustering

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



# A kind of class-based language model

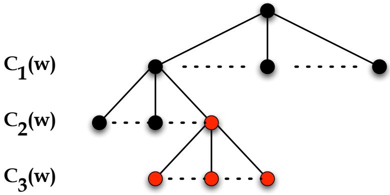
## Main ideas

As proposed by [Mnih and Hinton, 2008]:

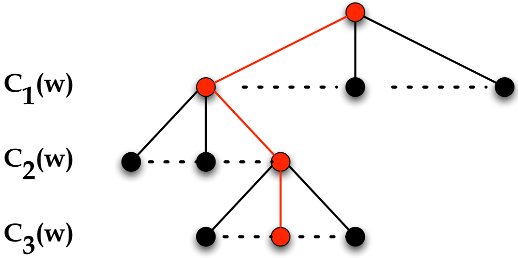
- Represent the vocabulary as a clustering tree [Brown et al., 1992].
- Predict the path in this clustering tree.

## Word clustering

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



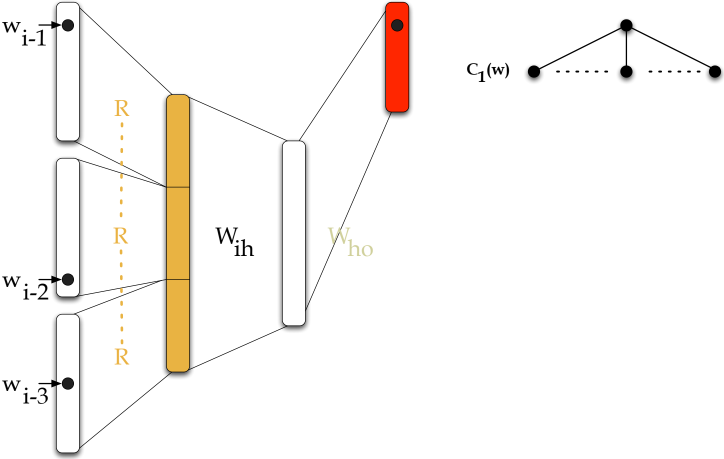
# Word probability



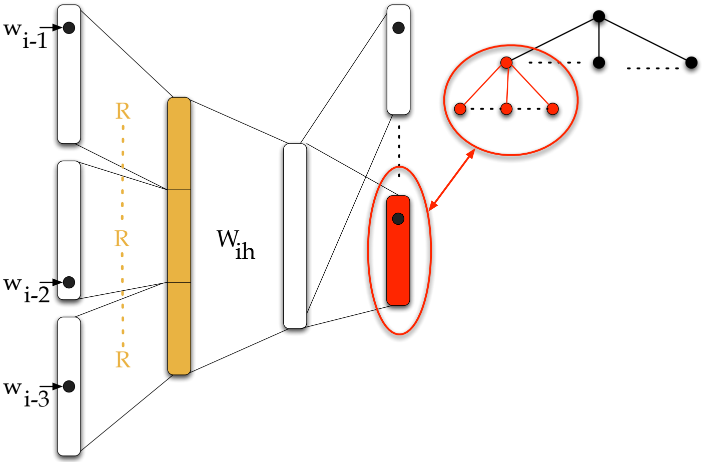

$$P(w_i|h) = P(c_1(w_i)|h) \prod_{d=2}^{D} P(c_d(w_i)|h, c_{1:d-1})$$

- $c_{1:D}(w_i) = c_1, \ldots, c_D$ : 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



# 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.

# 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.

# Training algorithm

**Step 1:**

Train a standard NNLM model with the short-list as an output
(3 epochs and a short-list 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 short-list).

**Step 4:**

Train the whole model

# Training algorithm

**Step 1:**

Train a standard NNLM model with the short-list as an output
(3 epochs and a short-list 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 short-list).

**Step 4:**

Train the whole model

# Training algorithm

### Step 1:

Train a standard NNLM model with the short-list as an output
(3 epochs and a short-list 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 short-list).

### Step 4:

Train the whole model

# Training algorithm

### Step 1:

Train a standard NNLM model with the short-list as an output
(3 epochs and a short-list 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 short-list).

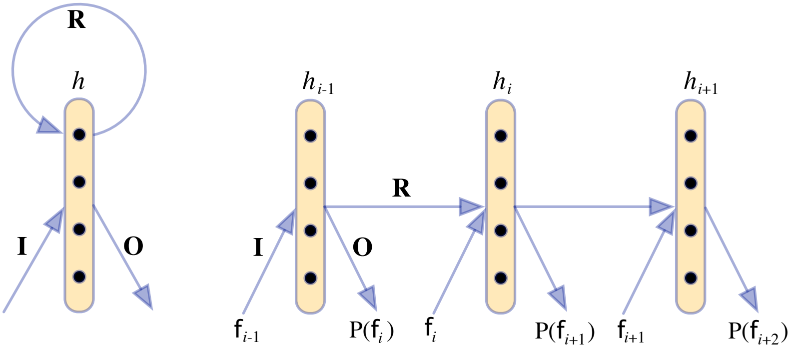
### Step 4:

Train the whole model

# Summary

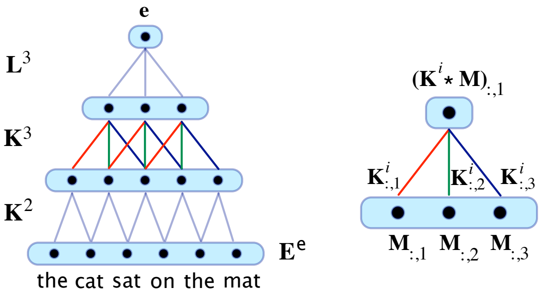

- SOUL LM combines two techniques: neural network and class-based language models.
- SOUL LM is the first complete large-scale continuous space language model.
- Within a large-scale task, significant improvement is achieved.
- When increasing context length, SOUL LM improves the performance.

# Future/Other work - Recurrent model

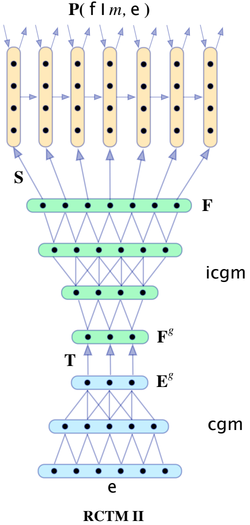



[Kalchbrenner and Blunsom, 2013b]

# Future/Other work - Sentence model

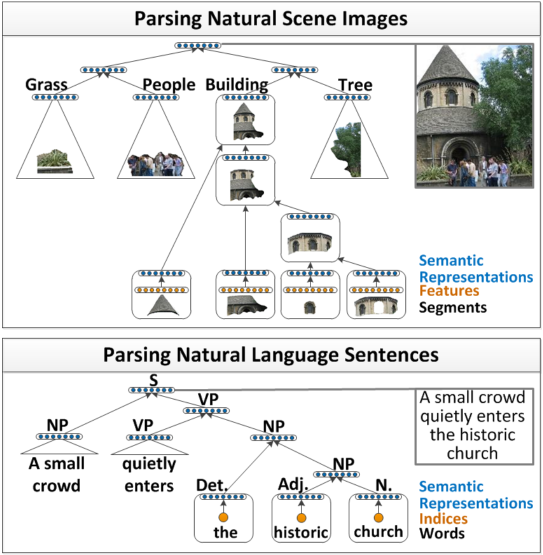


# Future/Other work - Towards translation models




[Kalchbrenner and Blunsom, 2013a]

# Future/Other work - Linguistic structure




[Socher et al., 2011]

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

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003).
A neural probabilistic language model.
*JMLR*, 3:1137–1155.

Brown, P., de Souza, P., Mercer, R., Della Pietra, V., and Lai, J. (1992).
Class-based n-gram models of natural language.
*Computational Linguistics*, 18(4):467–479.

Kalchbrenner, N. and Blunsom, P. (2013a).
Recurrent continuous translation models.
In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Kalchbrenner, N. and Blunsom, P. (2013b).
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. Association for Computational Linguistics.

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

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

Socher, R., Lin, C. C.-Y., Ng, A. Y., and Manning, C. D. (2011).
Parsing Natural Scenes and Natural Language with Recursive Neural Networks.
In *Proceedings of the 26th International Conference on Machine Learning (ICML)*.