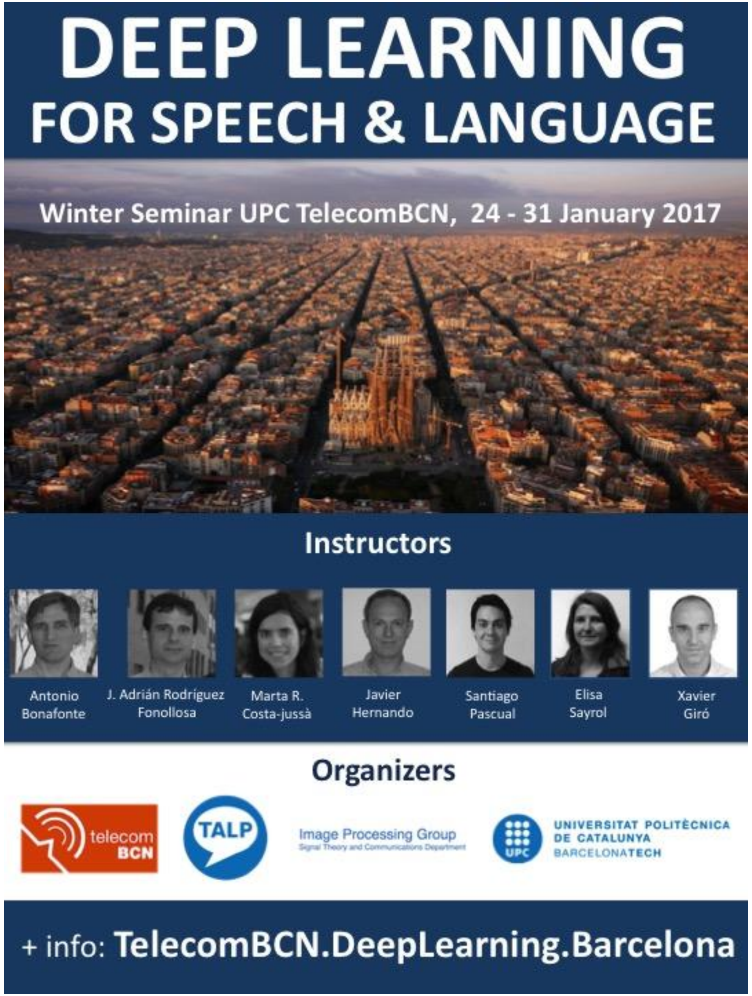


[course site]

Day 3 Lecture 1

Language Model

Marta R. Costa-jussà

Previous concepts from this course

- Word embeddings
- Feed-forward network and softmax
- Recurrent neural network (handle variable-length sequences)

What is the most probable sentence?

Two birds are flying

Two beards are flying

Probability of a sentence

- Suppose you record a database of one billion utterances in English.
- If the sentence “how's it going?” appears 76,413 times in that database, then we say
  - P(how's it going?) = 76,413/1,000,000,000

A language model finds the probability of a sentence

- Given a sentence (w1, w2, … wT),
- What is p(w1, w2, ... ,wT) =?

An n-gram language model

Chain rule probability and Markov simplifying assumption

p(w1, w2, . . . ,wT) = p(wT|w(T-1),w(T-2)...w1) p(w(T-1)|w(T-2),w(T-3)...w1) … p(w1)

Markov simplying assumption: The current word only depends on *n* previous words.

p(wt|w(t-1)w(t-2)..w1) ~ p(wt|w(t-1))

Objective

$$p(w_1, w_2, \ldots w_T) = \prod_{t=1}^{T} p(w_t | w_1 \ldots w_{t-1})$$

$$\approx \prod_{t=1}^{T} p(w_t | w_{t-1} \ldots w_{t-n})$$

An n-gram-based language model

- An n-word substring is called an n-gram.
- If n=1, we say unigram; if n=2, we say bigram; if n=3, we say trigram.
- P(<s> I like snakes that are not poisonous </s>) ~

b(I | <s>)*b(like | I) *b(snakes | like) *... *b(poisonous | not) *b(</s>| poisonous)

An n-gram-based language model

Unigram probabilities

$$p(w_1) = \frac{count(w_1)}{total\ words\ observed}$$

Bigram probabilities

$$p(w_2|w_1) = \frac{count(w_1 w_2)}{count(w_1)}$$

Trigram probabilities

$$p(w_3|w_1 w_2) = \frac{count(w_1 w_2 w_3)}{count(w_1 w_2)}$$

Any issues with the above model?

Some examples...

"<s> que la fuerza te acompañe </s>", = *may the force be with you*

bigrams and trigrams like:

fuerza te

la fuerza te

fuerza te acompañe

te acompañe </s>

do not appear in the big corpus of El Periodico (40 M words)

BUT PROBABILITY OF THE SENTENCE SHOULD NOT BE ZERO!!!!

Sparse counts are a big problem

- Backing off avoids zero probabilities

$$
\begin{aligned}
&.8 * p(w_3|w_1 w_2) \\
&\boxed{\begin{aligned}&+.15 * p(w_3|w_2) \\ &+0.049 * p(w_3)\end{aligned}} \\
&+.001
\end{aligned}
$$

Sparse counts are a big problem

- Smoothing avoids zero probabilities

$$.8 * p(w_3|w_1 w_2)$$
$$+.15 * p(w_3|w_2)$$
$$+0.049 * p(w_3)$$
$$\boxed{+.001}$$

Any other issue?

Lack of generalization

Mary buys two apples and two oranges in the market

three apples are for me

the tree has three oranges

A neural language model

To generalize to un-seen n-grams

A neural language model

Find a function that takes as input *n-1* words and returns a conditional probability of the next one

$$p(w_1, w_2, \dots w_T) = \prod_{t=1}^{T} p(w_t | w_1 \dots w_{t-1})$$

$$\approx \prod_{t=1}^{T} p(w_t | w_{t-1} \dots w_{t-n})$$

$$p(w_t | w_{t-n}, \dots w_{t-1}) = \boxed{f_{w_t}(w_{t-n}, \dots, w_{t-1})}$$

Architecture: neural language model

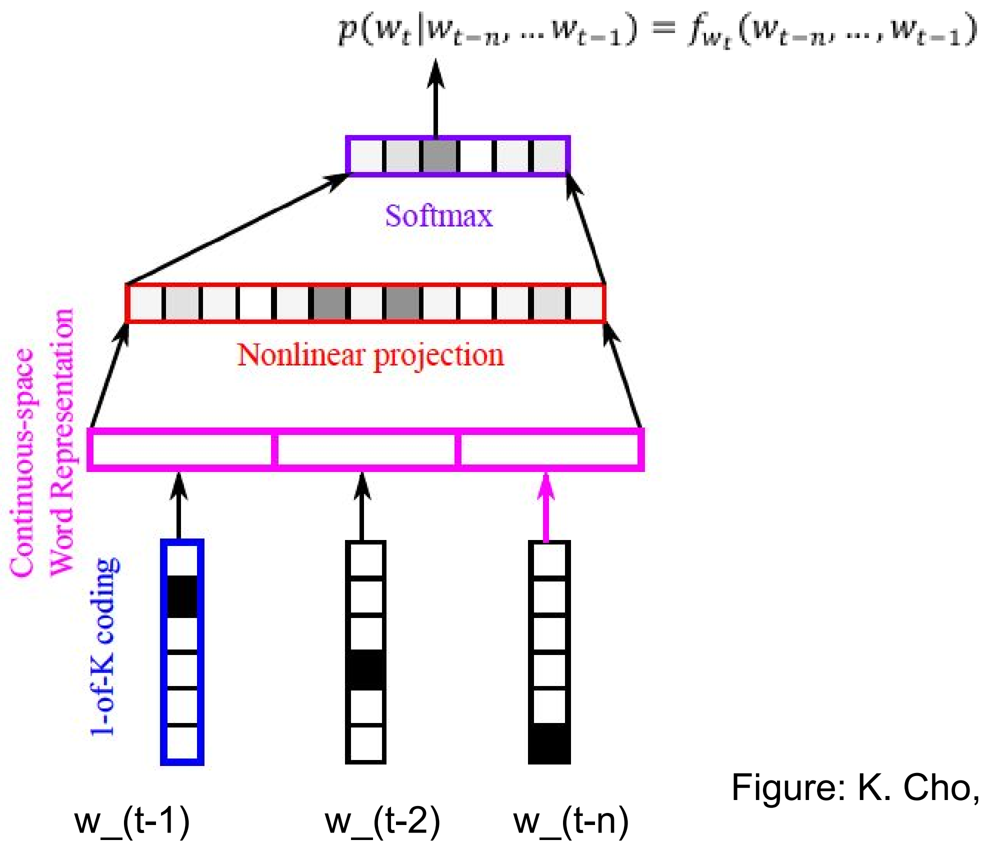
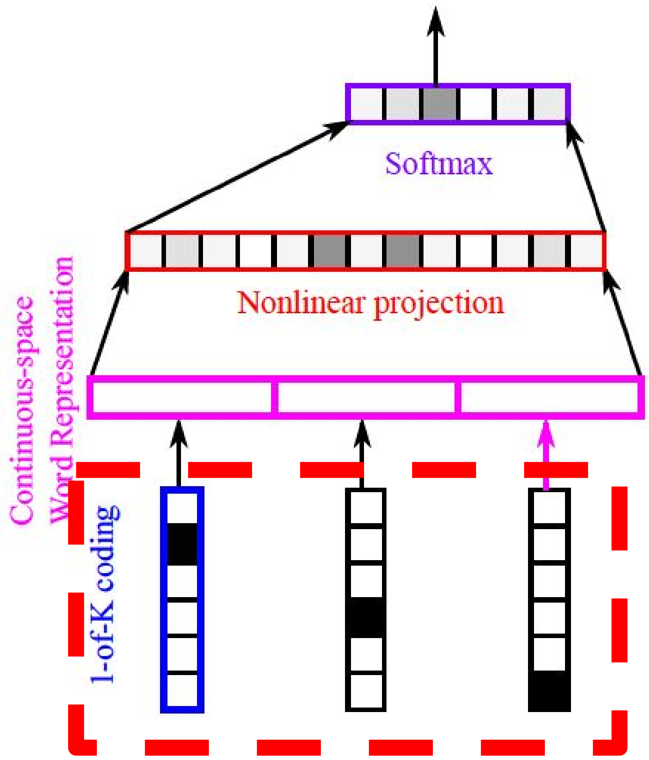


Figure: K. Cho, DL4MT course, 2015

Architecture: representation of input words



our goal is to put the least amount of prior knowledge

Figure: K. Cho, DL4MT course, 2015

Step 1: One-hot encoding

From previous lectures

Natural language words can be one-hot encoded on a vector of dimensionality equal to the size of the dictionary (K=|V|).

Word	One-hot encoding
economic	000010...
growth	001000...
has	100000...
slowed	000001...

Architecture: continuous word representation

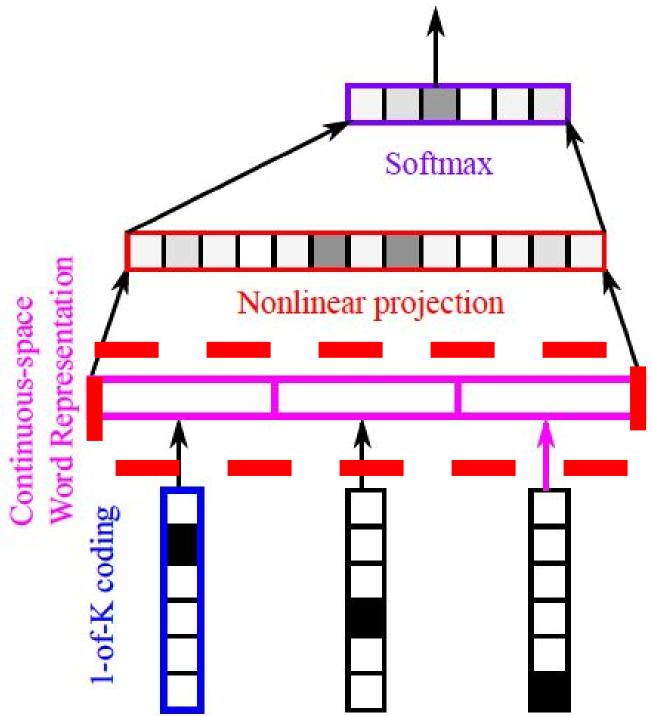


Figure: K. Cho, DL4MT course, 2015

input vectors are multiplied by the weight matrix (E), to obtain continuous vectors

this weight matrix (E) is also called word embedding and should reflect the meaning of a word

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \vdots \\ \mathbf{e}_{|V|} \end{bmatrix}, \quad \mathbf{e}_i \in \mathbb{R}^d.$$

$$\mathbf{E}^\top \mathbf{w}_i = \mathbf{e}_i.$$

$$\mathbf{E} \in \mathbb{R}^{|V| \times d}.$$

Architecture: context vector

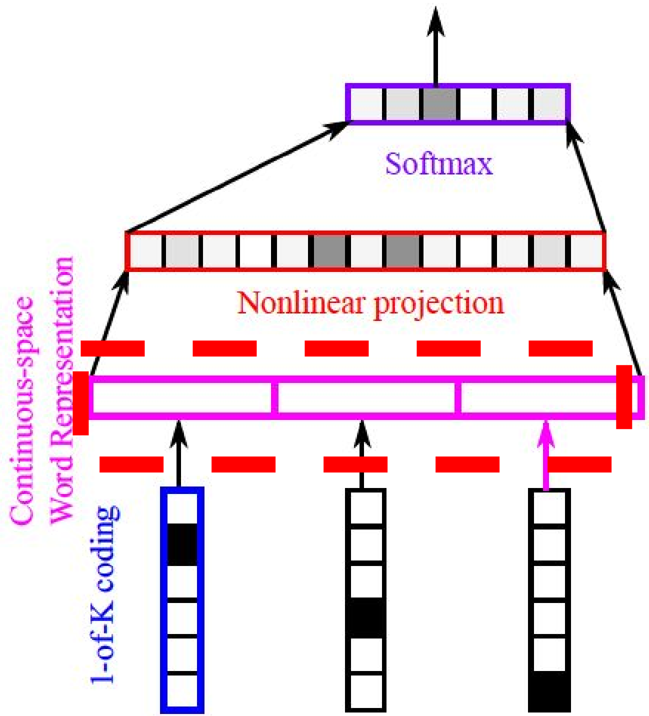


Figure: K. Cho, DL4MT course, 2015

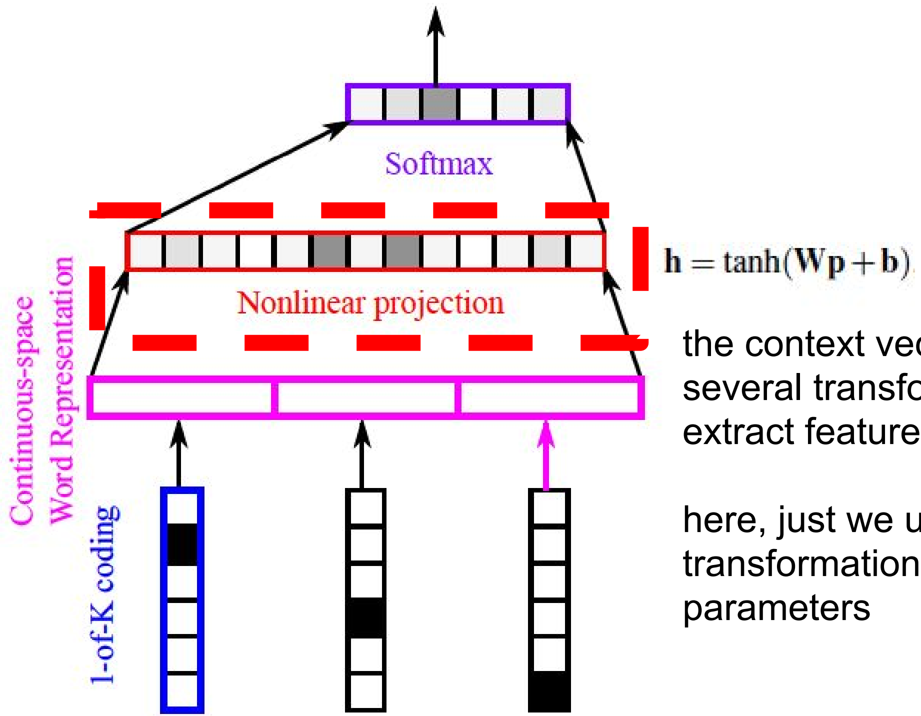
we get a sequence of continuous vectors, by concatenating the continuous representations of the input words

$$\mathbf{p}^j = \mathbf{E}^\top \mathbf{w}^j$$

$$\mathbf{p} = \left[\mathbf{p}^1; \mathbf{p}^2; \ldots; \mathbf{p}^{n-1}\right]^\top$$

context vector

Architecture: nonlinear projection



the context vector is fed through one or several transformation layers (which extract features),

here, just we used a simple transformation layer, with W and b as parameters

Figure: K. Cho, DL4MT course, 2015

Architecture: output probability distribution

$$p(w_t = k | w_{t-n}, \dots w_{t-1}) = \mu_k = \frac{\exp(u_k^T h)}{\sum_{k'}^{|V|} \exp(u_{k'}^T h)}$$

$$\mathbf{u}_k \in \mathbb{R}^{\dim(\mathbf{h})}.$$

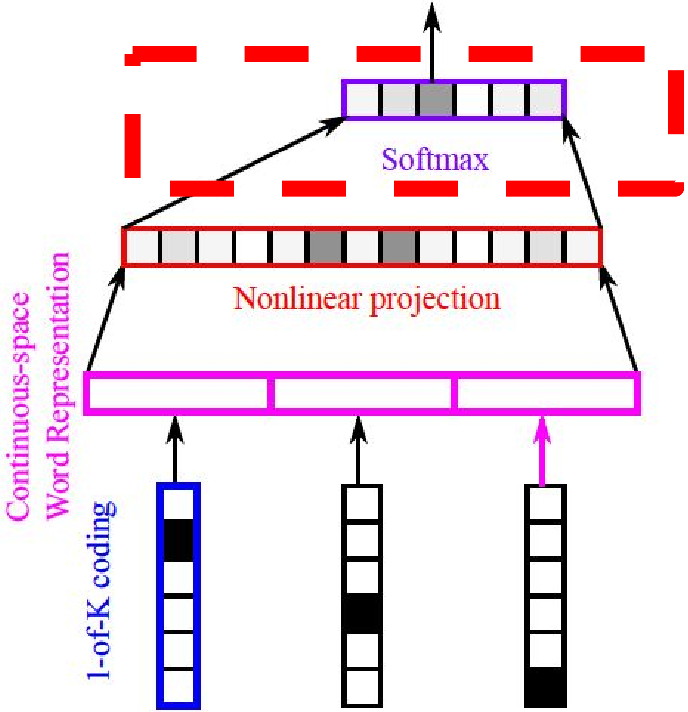


Figure: K. Cho, DL4MT course, 2015

LM has a categorical distribution, where the probability of the k-th event happening is denoted as μ_k

$$\sum_{k=1}^{K} \mu_k = 1.$$

the function needs to return a K-dimensional vector $[\mu_1, \mu_2, \dots, \mu_K]$ $K = |V|$ and μ_i corresponds to the probability of the i-th word in the vocabulary for the next word

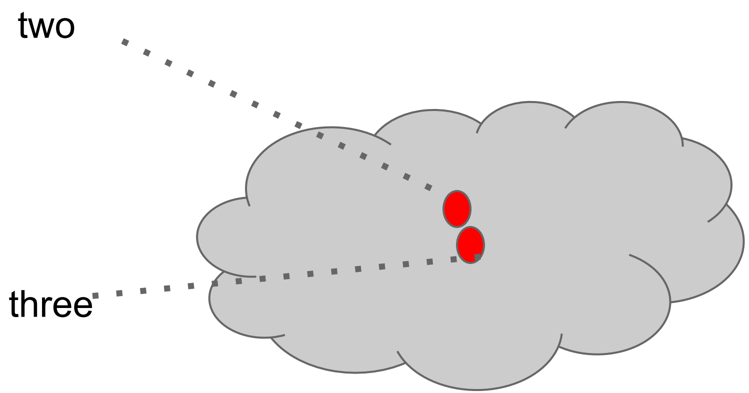
Why this model is generalizing to unseen events?

Further generalization comes from embeddings

Mary buys two apples and two oranges in the market

three apples are for me

the tree has three oranges



Recurrent Language Model

To further generalize to un-seen n-grams

A neural language model

Still assumes the n-th order Markov property

it looks only as n-1 past words

In France , there are around 66 million people and they speak French.

How we can modelate variable-length input?

$$p(w_1, w_2, \dots w_T) = \prod_{t=1}^{T} p(w_t | w_1 \dots w_{t-1})$$

How we can modelate variable-length input?

$$p(w_1, w_2, \dots w_T) = \prod_{t=1}^{T} p(w_t | w_1 \dots w_{t-1})$$

We directly model the original conditional probabilities

Via Recursion:

$h_0 = 0$ Initial condition

$h_t = f(w_{t-1}, h_{t-1})$ Recursion

summarizes the history from w1 to w(t-1)

How we can modelate variable-length input?

$$p(w_1, w_2, \dots w_T) = \prod_{t=1}^{T} p(w_t | w_1 \dots w_{t-1})$$

We directly model the original conditional probabilities

Via Recursion:

$h_0 = 0$ Initial condition

$h_t = f(w_{t-1}, h_{t-1})$ Recursion

summarizes the history from w1 to w(t-1)

The RNN is capable of summarizing a variable-length input sequence (w) into a memory state (h)

Example

p(Mary, buys, the,apple)

(1) Intialization: h0=0 -> p(Mary)=g(h0)
(2) Recursion
    (a) h1=f(h0,Mary) ->p(buys|Mary)=g(h1)
    (b) h2=f(h1,buys) -> p(the|Mary,buys)=g(h2)
    (c) h3=f(h2,the) -> p(apple|Mary,buys,the)=g(h3)
(3) Output: p(Mary,buys,the,apple)=g(h0)g(h1)g(h2)g(h3)

It works for any number of context words

READ, UPDATE, PREDICT

A recurrent neural language model

Can modelate the probability of a variable-length input sequence

$$p(w_1, w_2, \ldots w_T) = \prod_{t=1}^{T} \underbrace{p(w_t | w_1 \ldots w_{t-1})}$$

conditional probability that we want to compute

A recurrent neural language model

what we need

(1) Transition function $h_t = f(w_{t-1}, h_{t-1})$

(2) Output function $p(w_t = k | w_1 \dots w_{t-1})$

(Naive) Transition function

Inputs:

one-hot vector $w_{(t-1)} \in \{0,1\}^{|V|}$

\+ hidden state $h_{(t-1)} \in \mathbb{R}^{d}$

Parameters:

input weight matrix $E \in \mathbb{R}^{d\,x\,|V|}$

\+ transition weight matrix $U \in \mathbb{R}^{d\,x\,d}$

\+ bias vector (b)

(Naive) Transition function

Linear Transformation of the Previous Hidden State

nonlinear transformation

$$h_t = tanh(E^T w_{t-1} + U h_{t-1} + b)$$

Continuous-space Representation of word

Output function

Input:

hidden state (ht)

Parameters:

output matrix $V \in \mathbb{R}^{|V| x d}$

bias vector (c)

Output function

This summary vector is affine-transformed followed by a softmax nonlinear function to compute the conditional probability of each word.

$$\mu = softmax\left(Vh_t + c\right)$$

Measure to evaluate

Perplexity

Perplexity: a measure to evaluate language modeling

Perplexity measures how high a probability the language model assigns to correct next words in the test corpus “on average”. A better language model is the one with a lower perplexity.

Perplexity measures as well how complex is a task equals size of vocabulary (V)

PP=V

Comparing language models

LM	Hidden Layers	PPL
n-gram-based		131.2
+feed-forward	600	112.5
+RNN	600	108.1
+LSTM	600	92.0

Results from Sundermeyer et al, 2015

Applications of language modeling

Speech recognition

Machine translation

Handwriting recognition

Information retrieval

...

A bad language model

Summary

- Language modeling consists in assigning a probability to a sequence of words.
- We can model a sequence of words with n-grams, feed -forward networks and recurrent networks.
- Feed-forward networks are able to generalise unseen contexts
- RNN are able to use variable contexts

Learn more

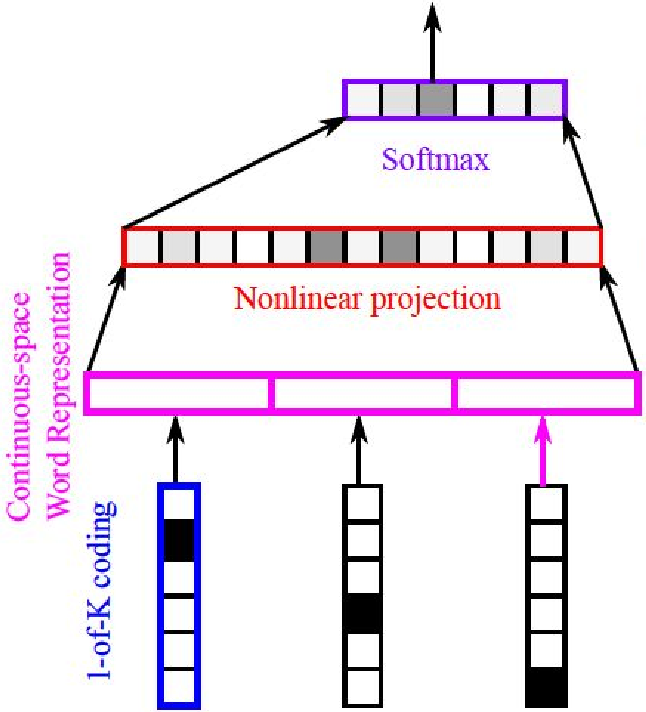
Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3 (March 2003), 1137-1155.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. 2015. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 23, 3 (March 2015), 517-529. DOI=http://dx.doi.org/10.1109/TASLP.2015.2400218

Thanks ! Q&A ?

https://www.costa-jussa.com

Architecture: neural language model



$$p(w_n = k | w_1, w_2, \ldots, w_{n-1}) = \mu_k = \frac{\exp(\mathbf{u}_k^\top \mathbf{h} + c_k)}{\sum_{k'=1}^{|V|} \exp(\mathbf{u}_{k'}^\top \mathbf{h} + c_{k'})}, \quad \mathbf{u}_k \in \mathbb{R}^{\dim(\mathbf{h})}$$

$$\sum_{k=1}^{K} \mu_k = 1.$$

$$\mathbf{h} = \tanh(\mathbf{W}\mathbf{p} + \mathbf{b})$$

$$\mathbf{E}^\top \mathbf{w}_i = \mathbf{e}_i. \qquad \mathbf{E} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \vdots \\ \mathbf{e}_{|V|} \end{bmatrix}, \qquad \mathbf{p}^j = \mathbf{E}^\top \mathbf{w}^j \qquad \mathbf{E} \in \mathbb{R}^{|V| \times d}.$$

$$\mathbf{w}_i = [0, 0, \ldots, \underbrace{1}_{i\text{-th element}}, \ldots, 0]^\top \in \{0, 1\}^{|V|}$$