Sequence modeling

Alex Allauzen

2017-11-28

Outline

Plan

Sequence tagging

$$\mathbf{w} = w_1^L = w_1, w_2, ..., w_L$$
$$\mathbf{t} = t_1^L = t_1, t_2, ..., t_L$$

Example : Part-of-Speech (POS) tagging

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
.	$.

Language model / Generative sequence model

Applications

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

The goal

Estimate the **non-zero** probability of a word sequence given a vocabulary

$$P(w_1^L) = P(w_1, w_2, ...w_L) = \prod_{i=1}^{L} P(w_i|w_1^{i-1}), \quad \forall i, w_i \in \mathcal{V}$$

with the n**-gram assumption**:

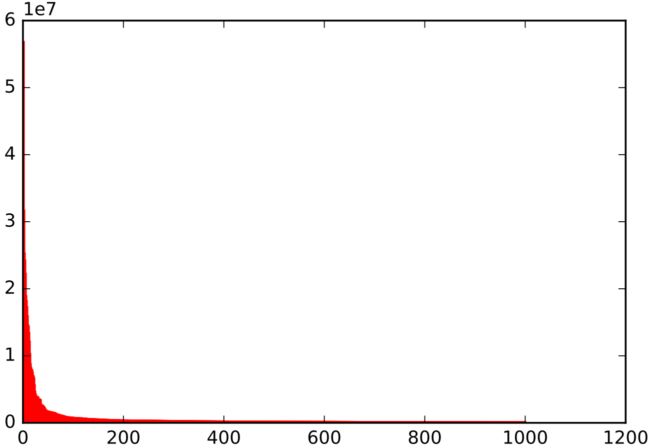
$$P(w_1^L) = \prod_{i=1}^{L} P(w_i|w_{i-n+1}^{i-1}), \quad \forall i, w_i \in \mathcal{V},$$

in the **recurrent** way

$$P(w_i|w_1^{i-1})$$

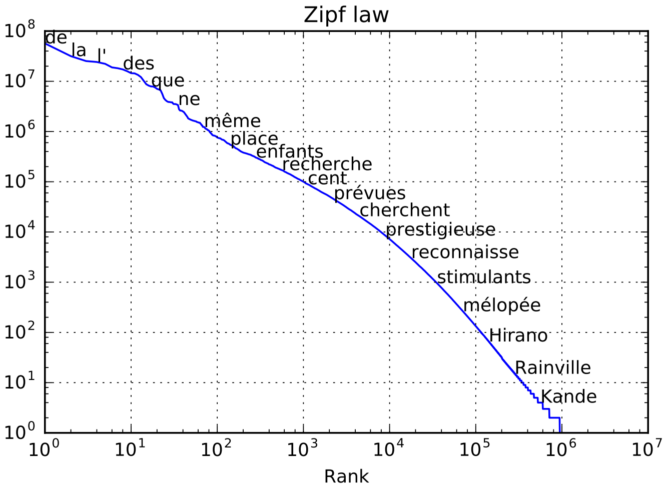
The Zipf law (for French)

$$frequency \propto 1/rank$$



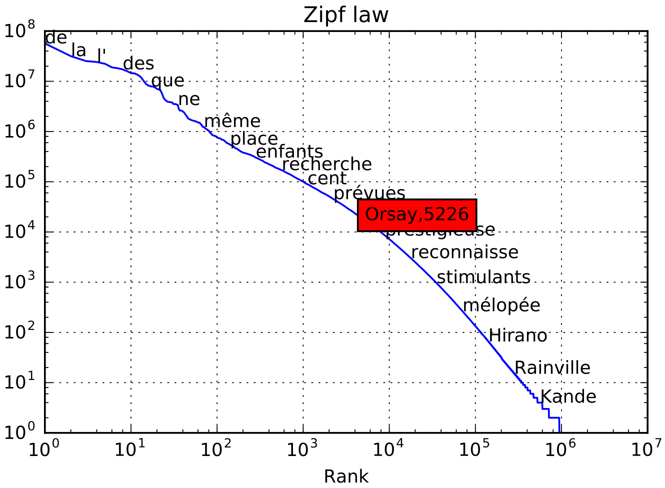
The Zipf law (for French)

$$frequency \propto 1/rank$$



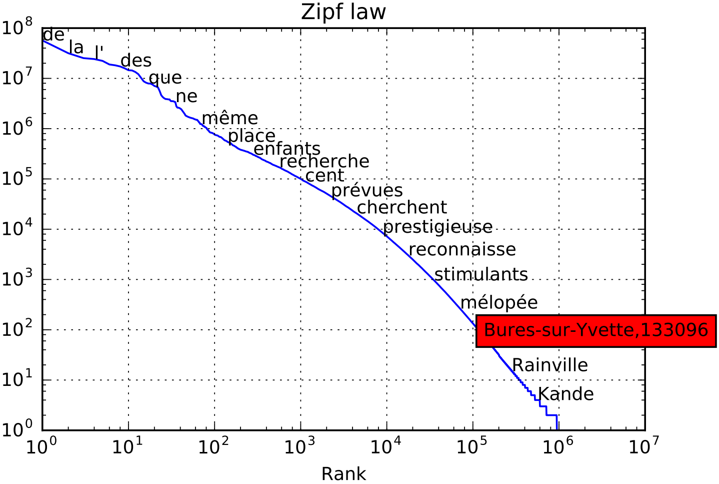
The Zipf law (for French)

$$frequency \propto 1/rank$$



The Zipf law (for French)

$$frequency \propto 1/rank$$



Plan

Count-based language model

n-gram LM

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

$$P_{ML}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}$$

- Very inefficient !
- How to deal with zero counts in the denominator ?
- In the numerator ?
- Zero counts are the most frequent case.

→ smoothing

Count-based language model, smoothed version

n-gram LM

$$P_S(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) + \alpha}{c(w_{i-n+1}^{i-1}) + \beta}$$

$$P(w_i|w_{i-n+1}^{i-1}) = \lambda P_{ML}(w_i|w_{i-n+1}^{i-1}) + (1-\lambda)P(w_i|w_{i-n+2}^{i-1})$$

- Methods differ in how to estimate α, β, λ.
- See *An empirical study of smoothing techniques for language modeling*, Chen and Goodman, 1998/1999
- *Modified Kneser-Ney* is the best.

Evaluation of language model (bad, old, but still used)

- Log-Likelihood

$$LL(\mathcal{D}_{test}) = \sum_{w} \log P(w|h)$$

- Log-Likelihood per word

$$WLL(\mathcal{D}_{test}) = \frac{1}{|\mathcal{D}_{test}|} \sum_{w} \log P(w|h)$$

- Cross-Entropy per word

$$H(\mathcal{D}_{test}) = \frac{1}{|\mathcal{D}_{test}|} \sum_{w} -\log_2 P(w|h)$$

- Perplexity

$$PPL(\mathcal{D}_{test}) = 2^{H(\mathcal{D}_{test})} = e^{-WWL(\mathcal{D}_{test})}$$

Usage of language models

- Scoring hypothesis (ASR, MT)

time goes by so slowly $\rightarrow$ good
so slowly goes by time $\rightarrow$ bad

- Generate sentence

  $w = \#s\#$, $h = w$
  while w not $\#/s\#$
    Compute $P(w|h)$
    Sample $w \sim P(w|h)$
    $h = h + w$

- Both of them

Challenges

- Share knowledge among similar words

you go to London	you go to Montelimar
you went to London	you went to Montelimar

- Keep/skip what is meaningful/meaningless

Dr. Janet Smith talks

- Long distance dependancies

to greatly play **chess** he wants to have a nice **board**
to greatly play **music** he wants to buy a new **keyboard**

Challenges

- Share knowledge among similar words
- Keep/skip what is meaningful/meaningless
- Long distance dependancies

Plan

Log-linear n-gram model

$$n = 3 \rightarrow P(w_i | w_{i-2}, w_{i-1})$$
$$ex. : P(w_i | \text{give, me})$$

Log-linear n-gram model

$$n = 3 \rightarrow P(w_i | w_{i-2}, w_{i-1})$$
$$ex. : P(w_i | \text{give, me})$$

$$\begin{matrix} a \\ the \\ hand \\ five \\ \dots \\ w_i \end{matrix} \rightarrow \quad \underbrace{\mathbf{b} = \begin{pmatrix} 4.1 \\ 5.4 \\ 1 \\ -2.1 \\ \dots \end{pmatrix}}_{\text{prior score of } w_i}, \quad \underbrace{\boldsymbol{\theta}_{1,me} = \begin{pmatrix} -2.9 \\ 0.2 \\ -3.1 \\ 2.4 \\ \dots \end{pmatrix}}_{w_i|(w_{i-1} = \text{me})}, \quad \underbrace{\boldsymbol{\theta}_{2,give} = \begin{pmatrix} 1.3 \\ 2.2 \\ -1.1 \\ 0 \\ \dots \end{pmatrix}}_{w_i|(w_{i-2} = \text{give})}$$

Log-linear n-gram model

$$n = 3 \rightarrow P(w_i | w_{i-2}, w_{i-1})$$
$$ex. : P(w_i | \text{give, me})$$

$$\begin{array}{c} a \\ the \\ hand \\ five \\ \ldots \\ w_i \end{array} \rightarrow \quad \underset{\text{prior score of } w_i}{\mathbf{b} = \begin{pmatrix} 4.1 \\ 5.4 \\ 1 \\ -2.1 \\ \ldots \end{pmatrix}}, \quad \underset{w_i|(w_{i-1} = \text{me})}{\boldsymbol{\theta}_{1,me} = \begin{pmatrix} -2.9 \\ 0.2 \\ -3.1 \\ 2.4 \\ \ldots \end{pmatrix}}, \quad \underset{w_i|(w_{i-2} = \text{give})}{\boldsymbol{\theta}_{2,give} = \begin{pmatrix} 1.3 \\ 2.2 \\ -1.1 \\ 0 \\ \ldots \end{pmatrix}}$$

$$\mathbf{s} = \mathbf{b} + \boldsymbol{\theta}_{1,give} + \boldsymbol{\theta}_{2,me}$$
$$P(w_i | \text{give, me}) = \text{softmax}(\mathbf{s})$$

Log-linear n-gram model : parametrization

One classifier per context

- Given a context made 2 words: w_{i-2}, w_{i-1}
- Gather the associated parameters: $\boldsymbol{\theta}_{1,give} + \boldsymbol{\theta}_{2,me}$
- Add the bias term $\approx w_i$ frequency/importance
- Combine everything with a sum

Parameters

For each word in a context position : one real value per possible w_i

Challenges

Share knowledge among similar words	$\square$
Keep/skip what is meaningful/meaningless	$\boxtimes$
Long distance dependancies	$\square$

Generalization

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.

Generalization

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.

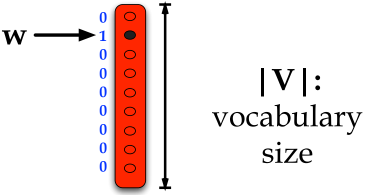
Why should it work ?

- "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 words into a continuous space

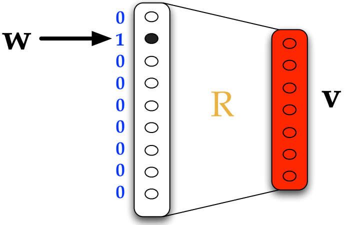
- The vocabulary is a neural network layer



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

Project a words into a continuous space

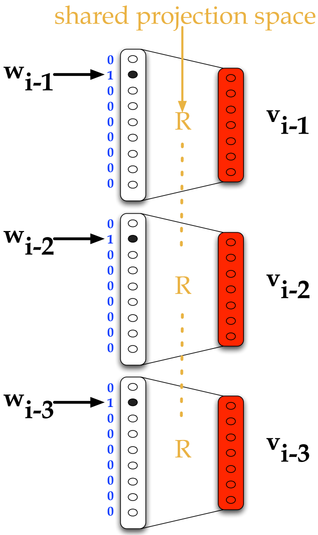
- The vocabulary is a neural network layer
- Word continuous representation:
  add a second layer fully connected



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

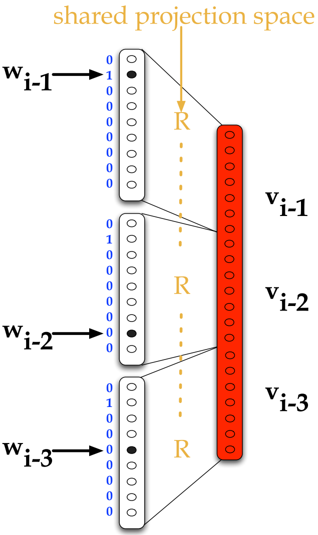
Project a words into a continuous space

- The vocabulary is a neural network layer
- Word continuous representation: add a second layer fully connected
- For a 4-gram, the history is a sequence of 3 words



Project a words into a continuous space

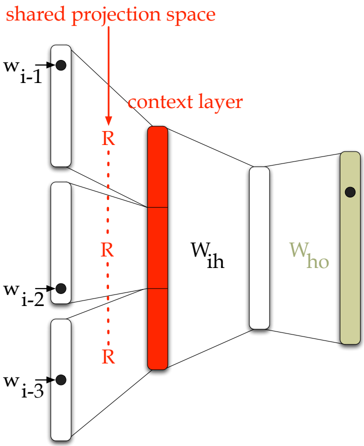
- The vocabulary is a neural network layer
- Word continuous representation: add a second layer fully connected
- For a 4-gram, the history is a sequence of 3 words
- Merge these three vectors to derive a single vector for the history



Estimate the n-gram probability

The program

- Given the history expressed as a feature vector : $\mathbf{v}$

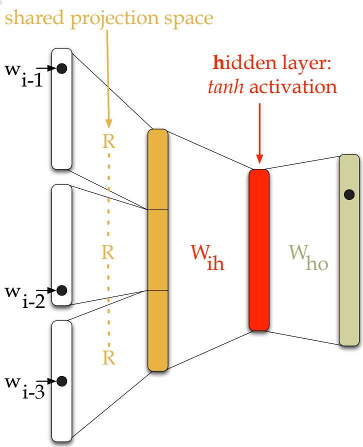


Estimate the n-gram probability

The program

- Given the history expressed as a feature vector : $\mathbf{v}$
- Create a feature vector for the word to be predicted:

$$\mathbf{h} = f(\mathbf{W}_{vh}\mathbf{v})$$



Estimate the n-gram probability

The program

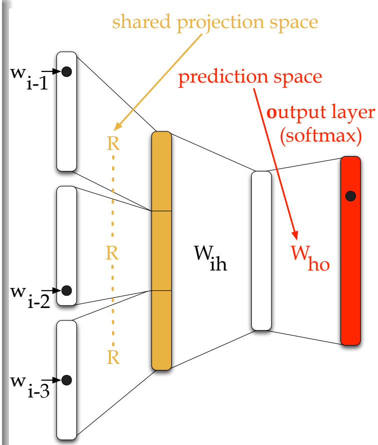
- Given the history expressed as a feature vector : $\mathbf{v}$
- Create a feature vector for the word to be predicted:

$$\mathbf{h} = f(\mathbf{W}_{vh}\mathbf{v})$$

- Estimate probabilities for all words given the history:

$$\mathbf{o} = f(\mathbf{W}_{ho}\mathbf{h})$$

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{\exp(o_{w_i})}{\sum_{w\in\mathcal{V}}\exp(o_{w_i})}$$



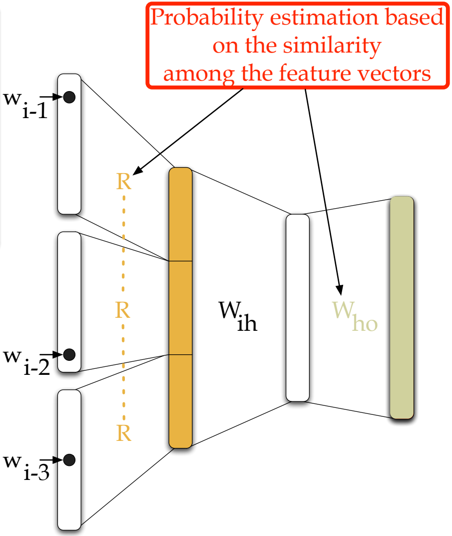
Assessment

Key points

- The projection in continuous spaces

$\rightarrow$ reduces the sparsity issues

- Learn simultaneously the projection and the prediction: $(\mathbf{R}, \mathbf{W_{vh}}, \mathbf{W_{ho}})$



Assessment

Key points

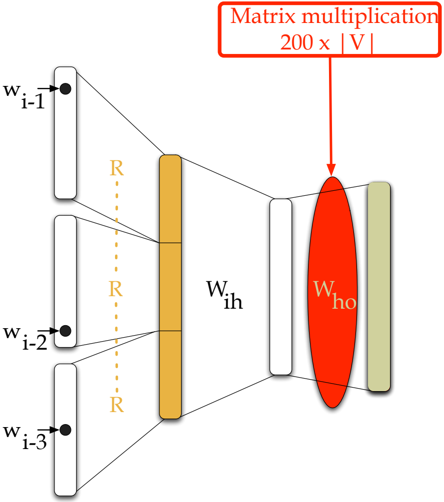
- The projection in continuous spaces

→ reduces the sparsity issues

- Learn simultaneously the projection and the prediction: $(\mathbf{R}, \mathbf{W_{vh}}, \mathbf{W_{ho}})$

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.

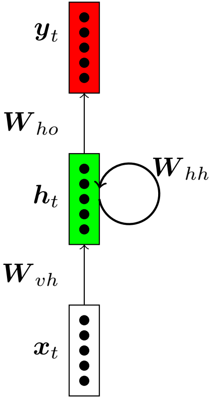


Challenges

Share knowledge among similar words	☒
Keep/skip what is meaningful/meaningless	☒/☐
Long distance dependancies	☒/☐

Plan

Recurrent network



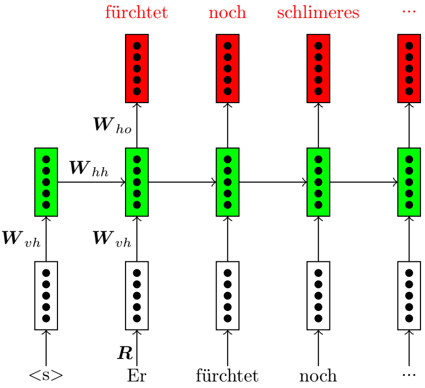
A dynamic system, at time t:

- 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 ($\boldsymbol{h}_t$)
- $\boldsymbol{x}_t$ comes from word embeddings

The same parameter set is shared across time steps

A recurrent network unfolded

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 $\boldsymbol{h}_t = f(\boldsymbol{W}_{vh}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hh}}\boldsymbol{h}_{\boldsymbol{t-1}})$
- The prediction at t from $\boldsymbol{h}_t$:

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

- g is the softmax function

Training recurrent model

Training algorithm

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

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

Known issues

- Vanishing/exploding gradient (Pascanu et al.2013) $\rightarrow$ LSTM, Gradient clipping
- Long-term memory $\rightarrow$ Bi-recurrent network

Gradient clipping

A simple and efficient trick

Given a threshold γ, before each update:

- Compute the norm of the gradient (at each time step) : $||\nabla_{\boldsymbol{\theta}}||$
- If $||\nabla_{\boldsymbol{\theta}}|| > \gamma$:

$$\nabla_{\boldsymbol{\theta}} \leftarrow \frac{\gamma}{||\nabla_{\boldsymbol{\theta}}||} \nabla_{\boldsymbol{\theta}}$$

Challenges

Share knowledge among similar words	☒
Keep/skip what is meaningful/meaningless	☒/☐
Long distance dependancies	☒/☐

Plan

Sequence modeling

n-gram models

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

- A sliding window of fixed size $(n-1)$
- n can be wide
- A kind of 1-d convolution

Recurrent models

$$P(w_i | w_1^{i-1})$$

- The hidden state accumulates the memory of the past
- Difficult to optimize : exploding/vanishing gradient
- Long range dependancies are still an issue

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, Réjean Ducharme, Pascal Vincent, and Christian Janvin.
2003.
A neural probabilistic language model.
Journal of Machine Learning Research, 3:1137–1155.

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.

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.

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.

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.