A Neural Probabilistic Language Model

By: Yoshua Bengio, Rejean Ducharme and Pascal Vincent

Introduction: Statistical Language Models

- Goal: Model distribution of natural language.
  Joint probability for a sequence of words: $P(w_0, w_1, \dots, w_n)$

- Predict the next word in a sequence given the words that precede it:

$$P(w_0, w_1, \dots, w_n)$$
$$= P(w_0) * P(w_1|w_0) \; * \; \dots \; * P(w_n|\, w_0, w_1, \dots, w_{n-1})$$

- Problem: Curse of dimensionality: 100.000 word vocabulary, 10 word context: 100.000^{10} possibilities

Previous Approach: *n-Gram Models*

n-Gram = series of n words (or just syllables or letters)

Paper refers to **Tri**gram models (n = 3)

Context matters! - Previous words

Generate tables of conditional probabilities for next word

```
THE CAT IS
CAT IS WALKING
IS WALKING IN
WALKING IN THE
IN THE BEDROOM
```

Limitations of *n-Gram Models*

Context limited to n – 1 previous words

„Similar" words not taken into consideration

⇨ Authors experiment with a new strategy!

A neural network approach

1. Distributed feature vector for each word in the vocabulary (real-valued vector in $\mathbf{R}^m$) -> Similarity between words.

2. Express joint probability function of word sequences in terms of the feature vectors in the sequence -> Using information of the sequence.

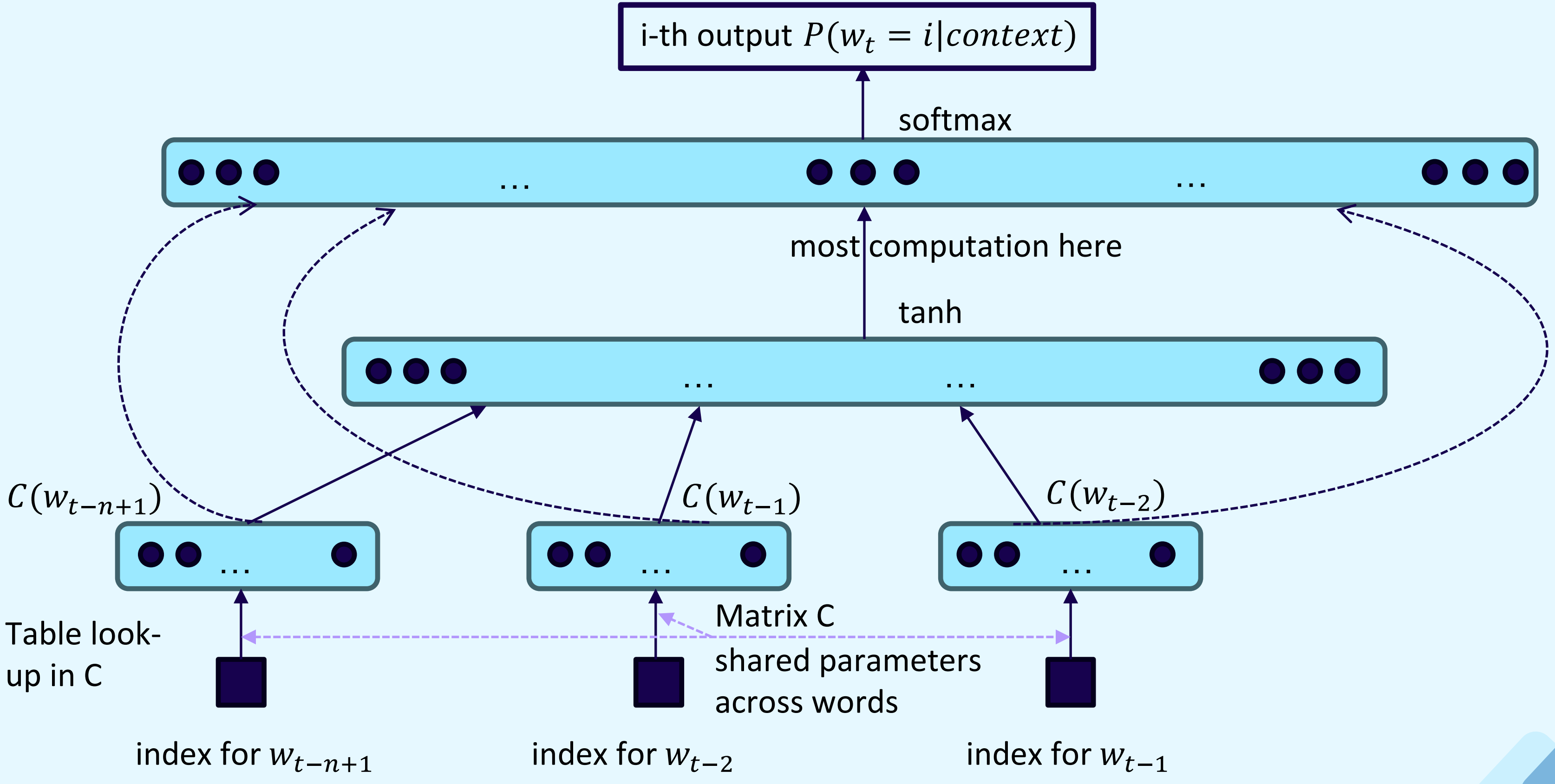
3. Learn simultaneously the word feature vectors and the parameters of the function.

Architecture of the Neural Network

$V = \{w_0, w_1, \ldots\}$

$C = |\mathrm{V}| \times m$, whose row i is the feature vector $C(i)$ for word i

V	Vocabulary
w	Word
t	Index of the word in the sequence (n-gram)
C	Look-up Table
m	Size of the feature vector
i	Index of a word in the Look-up Table
T	Length of word sequence



i-th output $P(w_t = i|context)$
softmax
most computation here
tanh
$C(w_{t-n+1})$
$C(w_{t-1})$
$C(w_{t-2})$
Table look-up in C
Matrix C
shared parameters across words
index for w_{t-n+1}
index for w_{t-2}
index for w_{t-1}

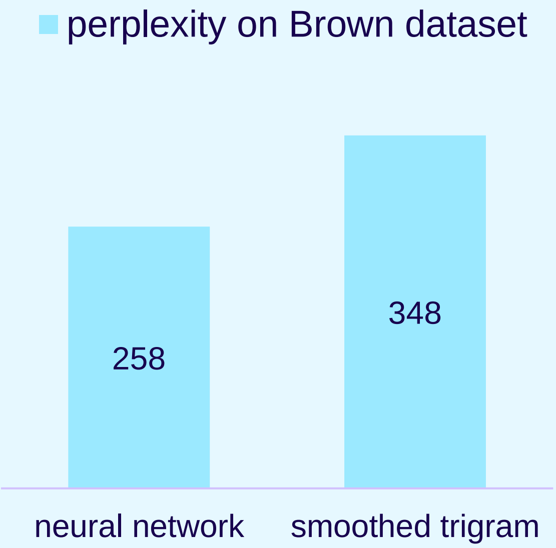
Why does it work?

- Possibility to generalize word context, which not appeared in the training set
  - Example $p(\text{"cat"} \mid \text{"the"}, \text{"small"})$
  - Trigram $[\text{"the"}, \text{"small"}, \text{"cat"}]$ is not in the training corpus
  - But $[\text{"the"}, \text{"small"}, \text{"dog"}]$ is
- Word representation (feature vector) of cat and dog is similar
  → NN is able to generalize $[\text{"cat"}]$
- NN could have learned similar trigrams like
  - $[\text{"a"},\text{"little"},\text{"dog"}]$
  - $[\text{"a"},\text{"little"},\text{"cat"}]$

Evaluation

Test results

- Up to 35% better prediction than smoothed trigram
- Better results by abstracting the words
- A larger context improves the result significantly
- Hidden layers improve the result



Evaluation

Problems / Improvements

- Performance
  - Training
    - stochastic gradient descent
    - early stopping
  - Execution
    - lookup table
    - short list
- No use of known grammatical structures