Neural POS-Tagging with Julia

Hiroyuki Shindo

2015-12-19
Julia Tokyo

Why Julia?

- NLPの基礎解析（研究レベル）
  - ベクトル演算以外の計算も多い（探索など）
  - Java, Scala, C++

- Python-like な文法で，高速なプログラミング言語

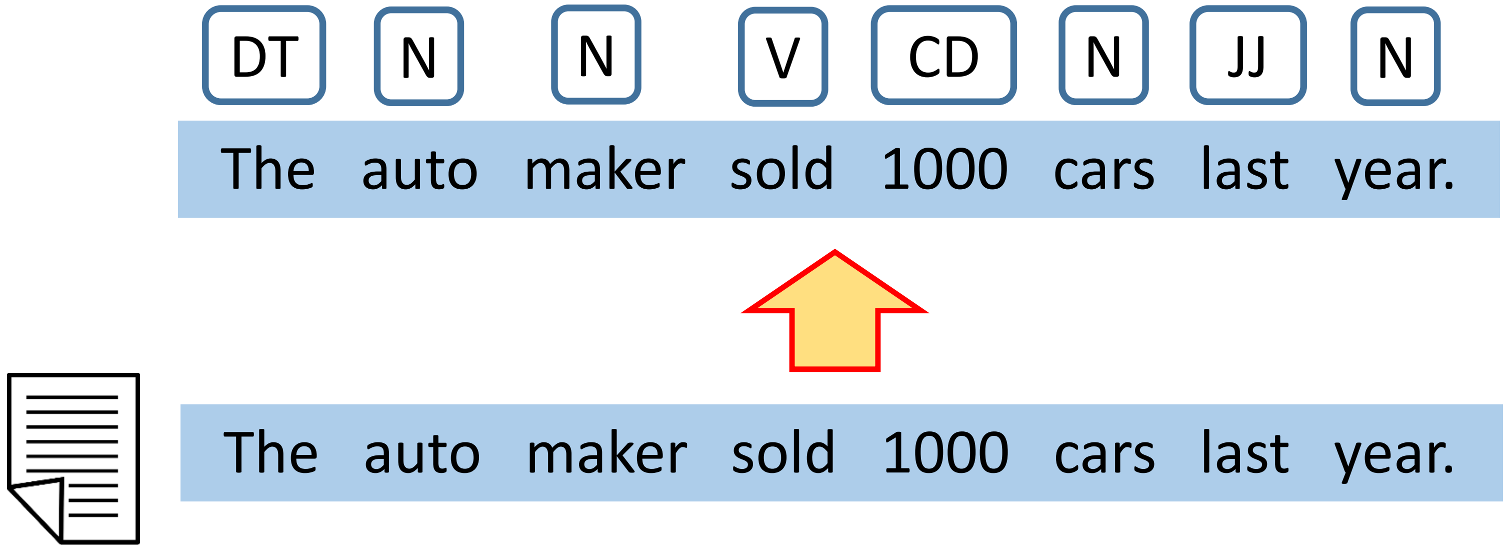
⇩

Julia, Nim, Crystal ?

Part-of-Speech (POS) Tagging

<u>品詞タグ（45種類）</u>

• DT:	限定詞 (the, a, an, ...)
• N:	名詞
• V:	動詞
• CD:	数字
• JJ:	形容詞

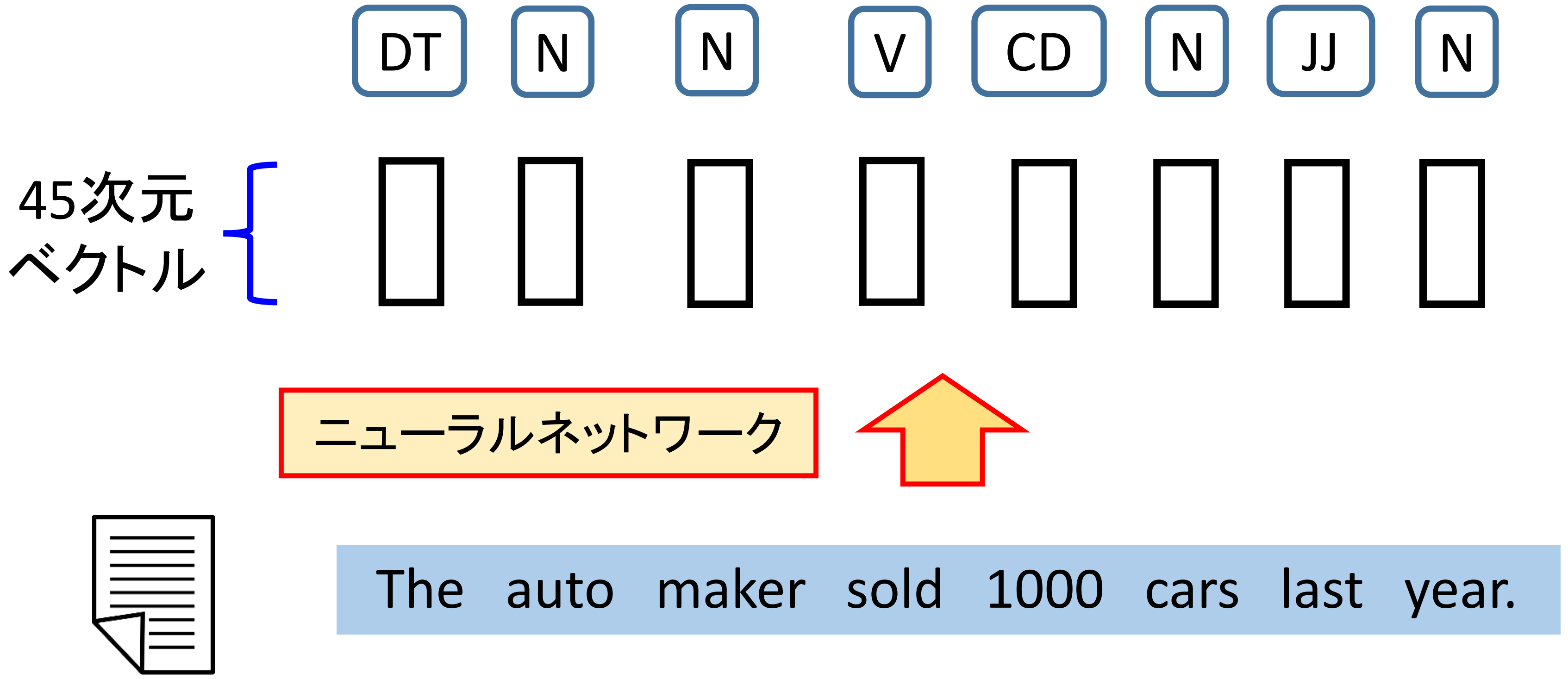

DT N N V CD N JJ N
The auto maker sold 1000 cars last year.
The auto maker sold 1000 cars last year.

List of POS-Tags for English

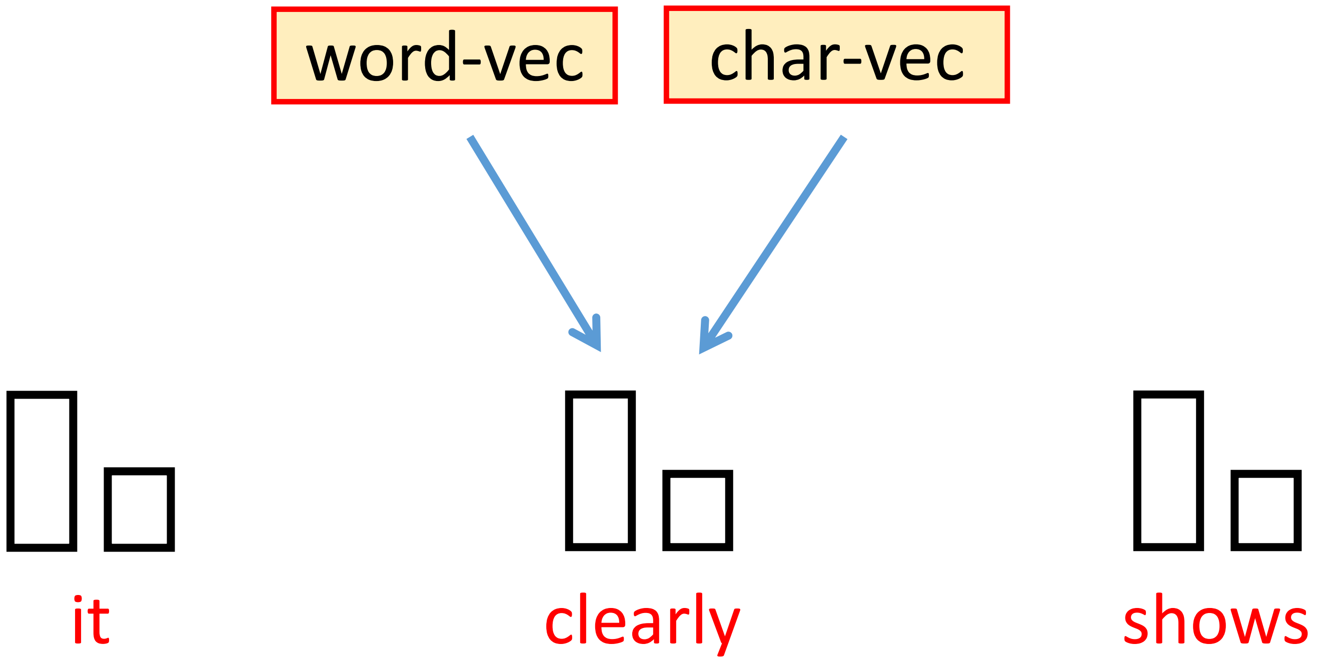
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Neural POS-Tagging

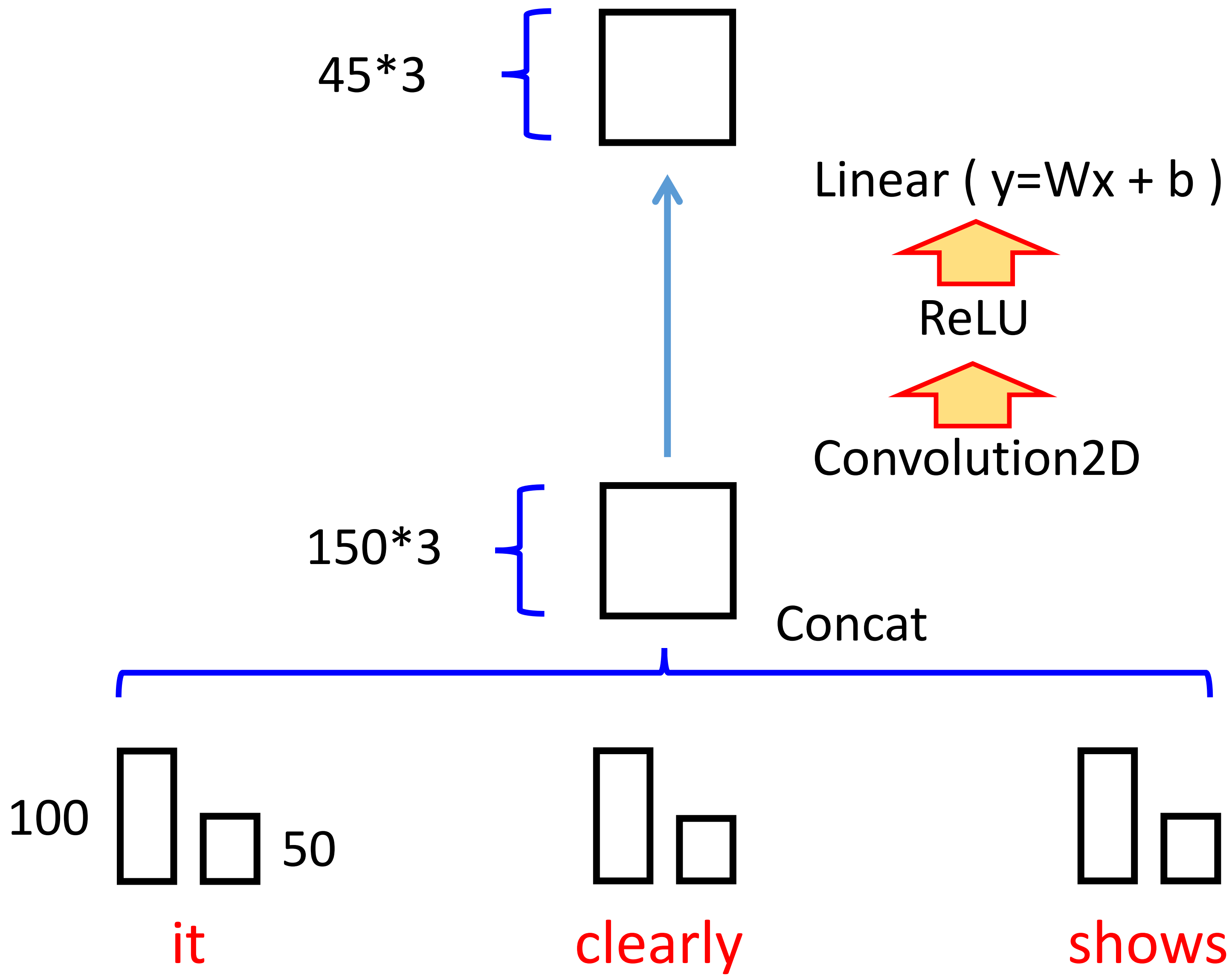
“Learning Character-level Representations for Part-of-Speech Tagging” [Santos+, ICML 2014]



Neural POS Tagger



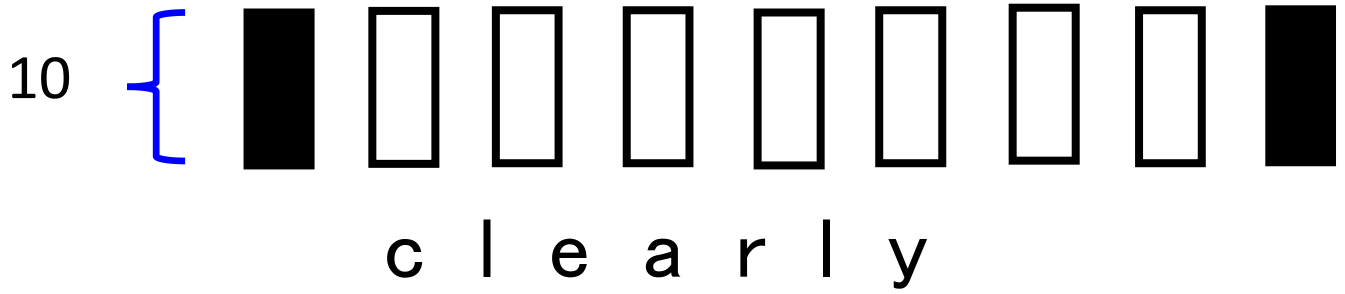
Neural POS Tagger



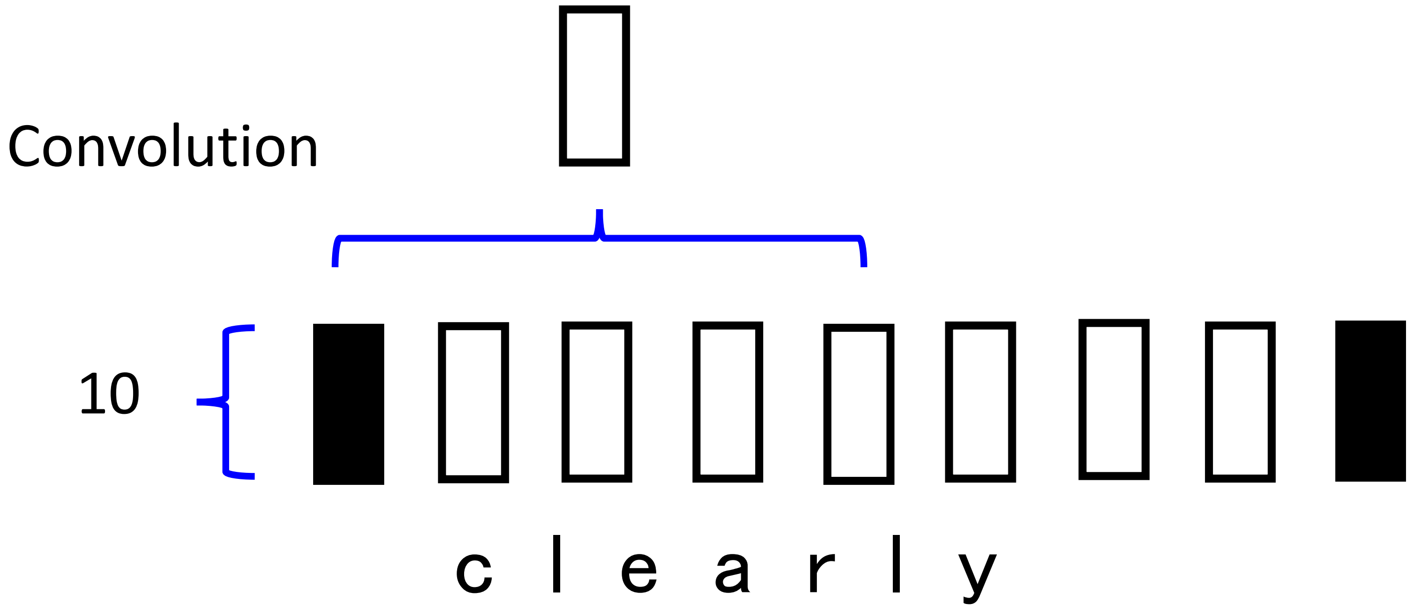
Character Vector

c l e a r l y

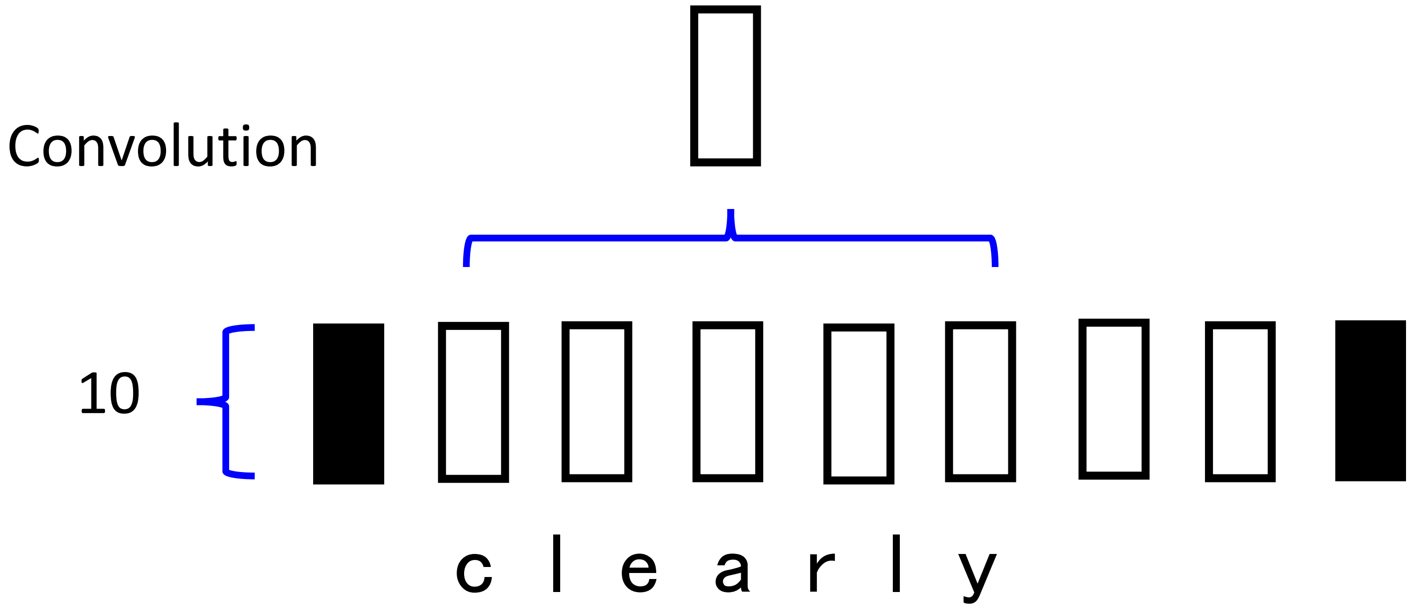
Character Vector



Character Vector

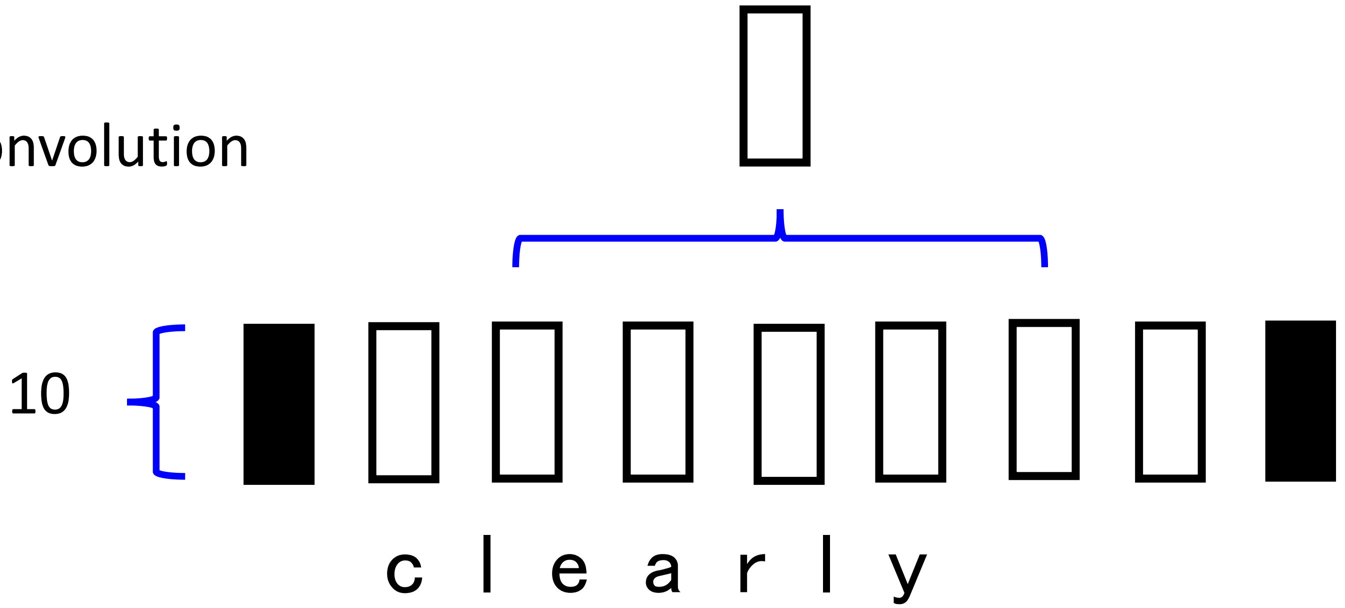


Character Vector

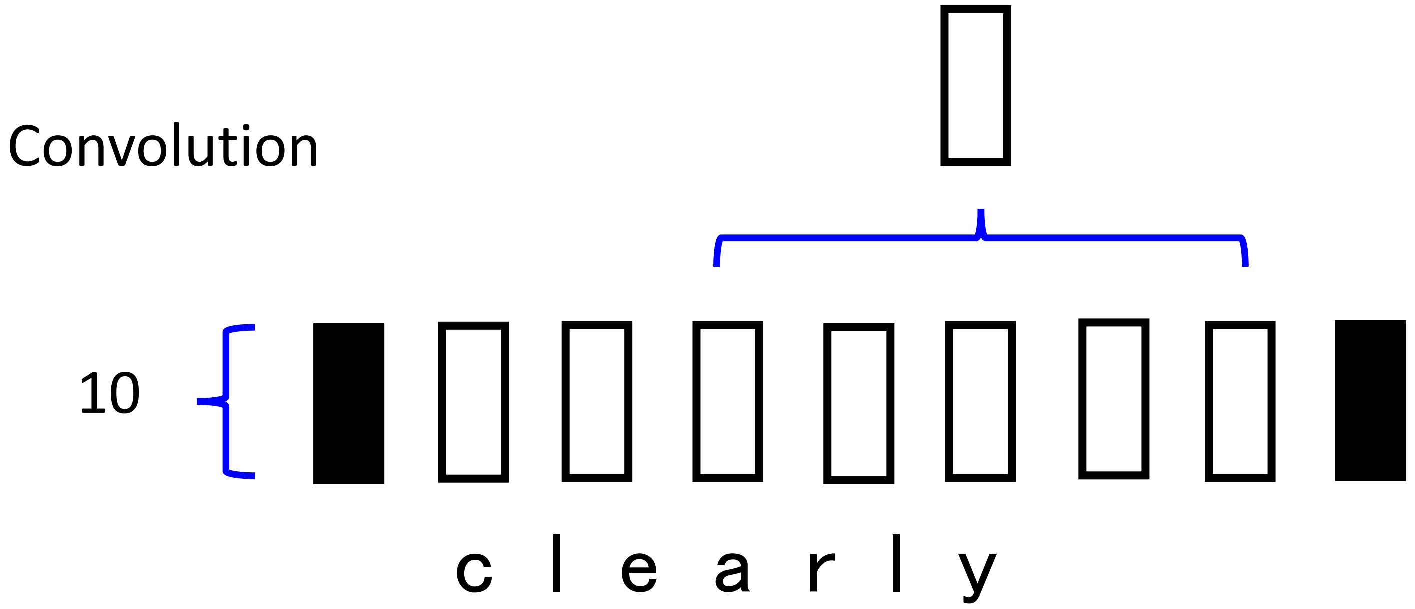


Character Vector

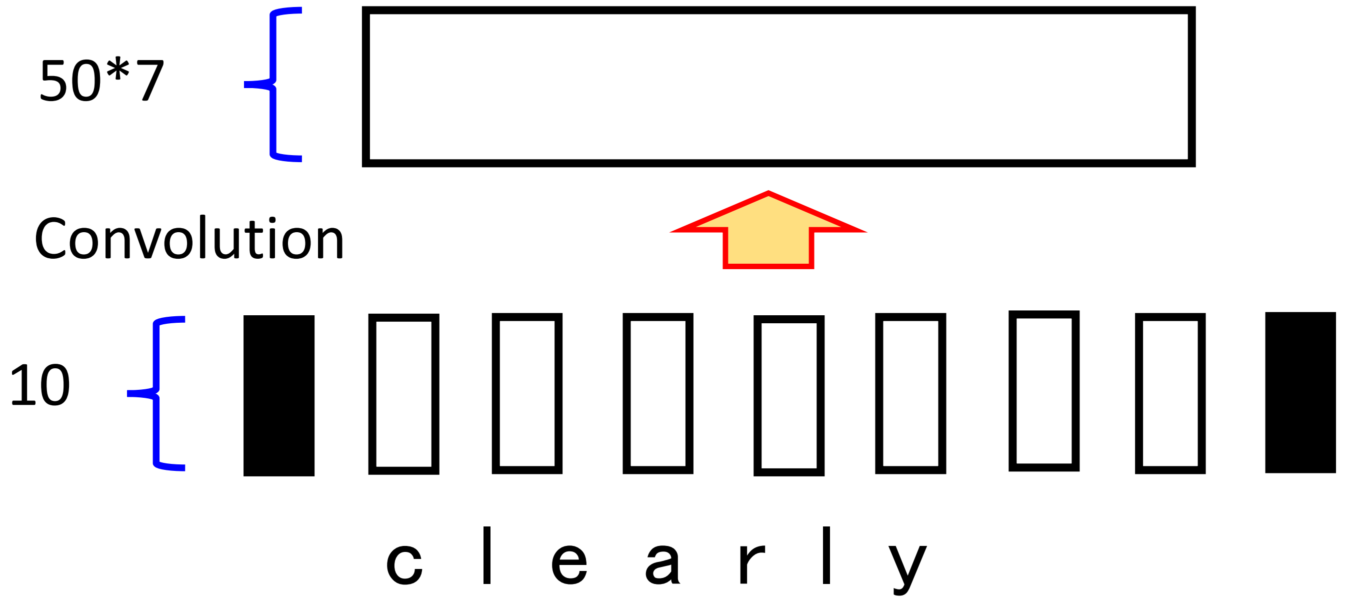
Convolution



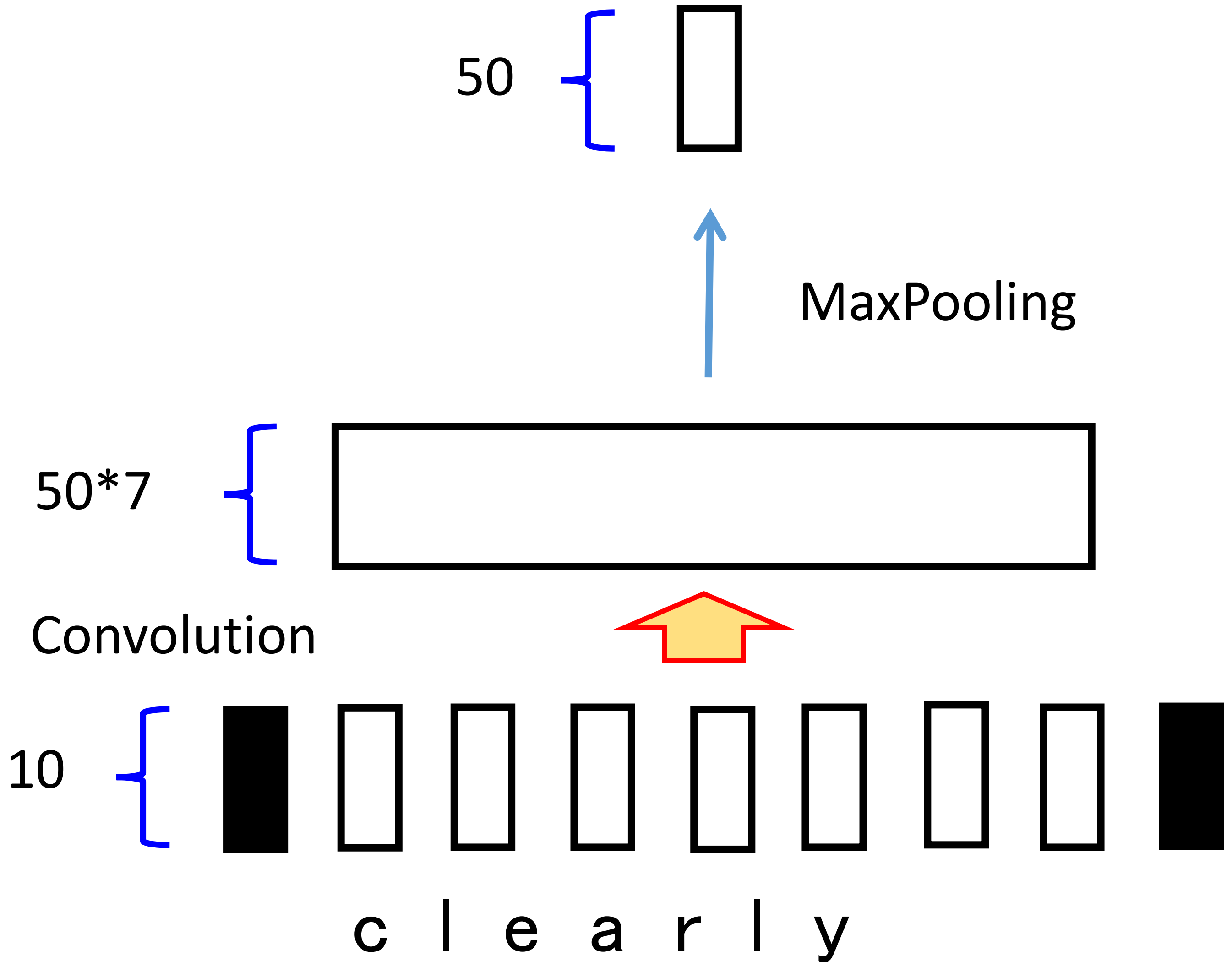
Character Vector



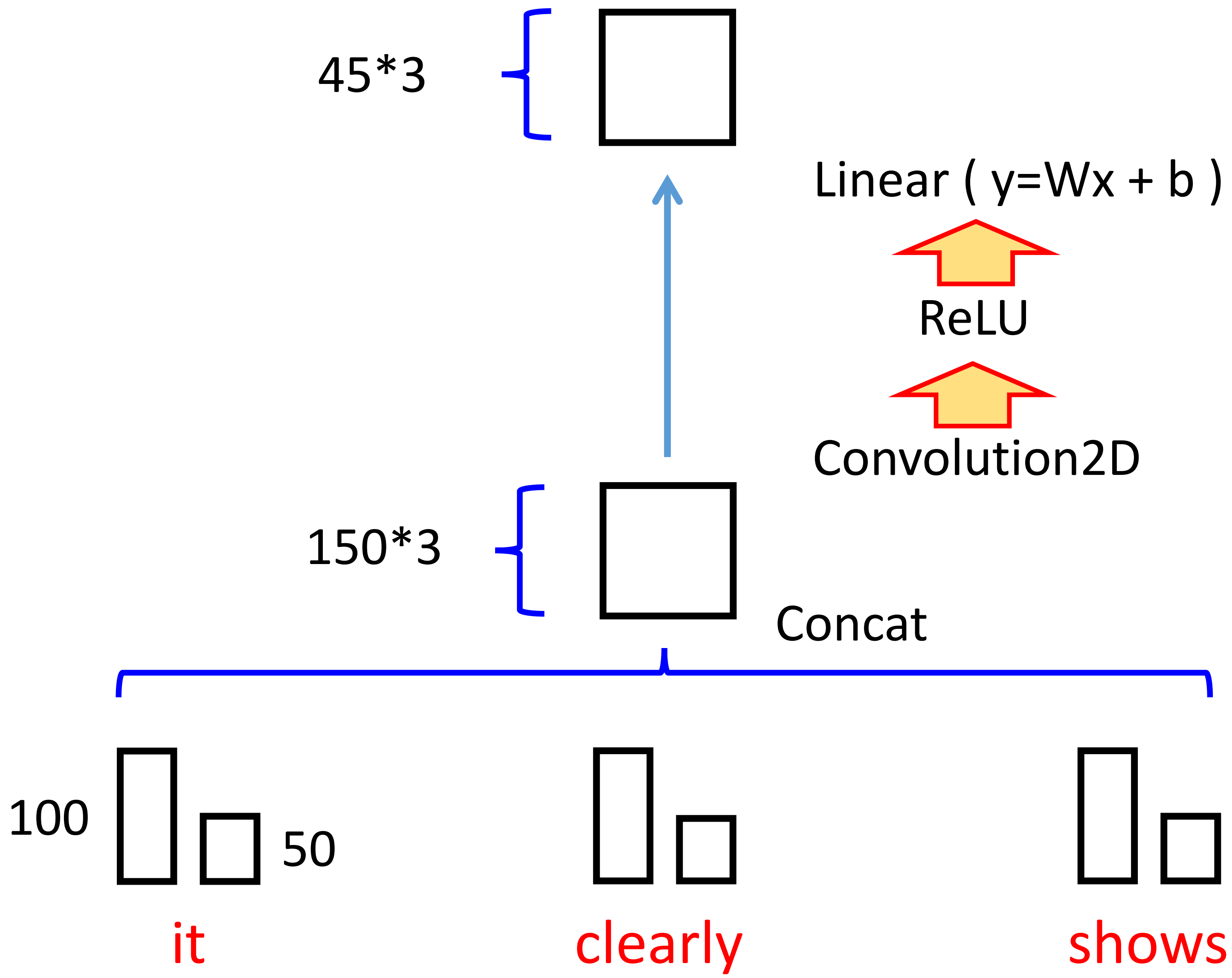
Character Vector



Character Vector



Neural POS Tagger



Neural Network Library for Julia

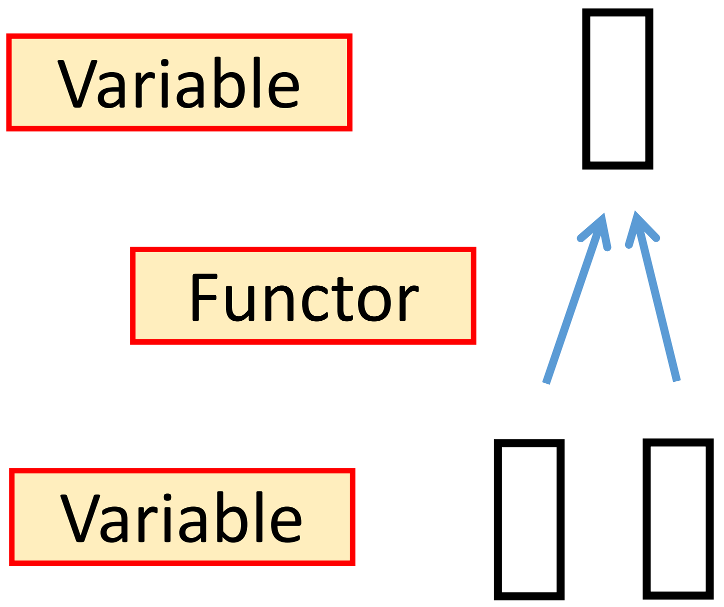
- Mocha.jl
- MXNet.jl

etc.

1. NLP用の minimal な library が欲しい
2. 言語解析では，動的ネットワークを書きやすいことが重要
   - ネットワーク構造が事前に決まらない（途中の計算結果に依存する）
   - GPU < CPUが起こりうる

Merlin.jl: Neural Network Library

- 実装： Julia,（一部）C++



Variable Type

```
type Variable
  value::Array
  grad::Array
  args::Union{Tuple{Variable},Vector{Variable}}
  diff::Function
  fixed::Bool
end
```

Functor Type

```
abstract Functor

type ReLU <: Functor
end

type Linear <: Functor
  weight::Variable
  bias::Variable
end

etc.
```

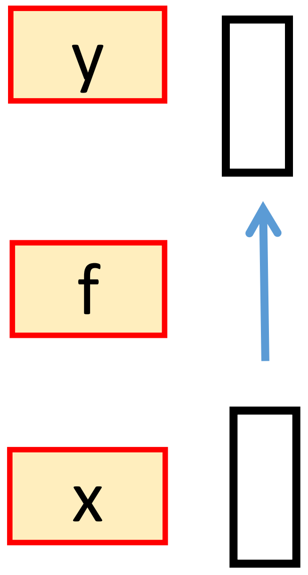
Functor Type

Functor は，apply, diff 関数を持つ

```
type ReLU <: Functor
end

function apply(f::ReLU, x::Array)
  y = similar(x)
  …
  y, gy -> diff(f, x, gy)
end

function diff(f::ReLU, x::Array, y::Array)
  …
end
```

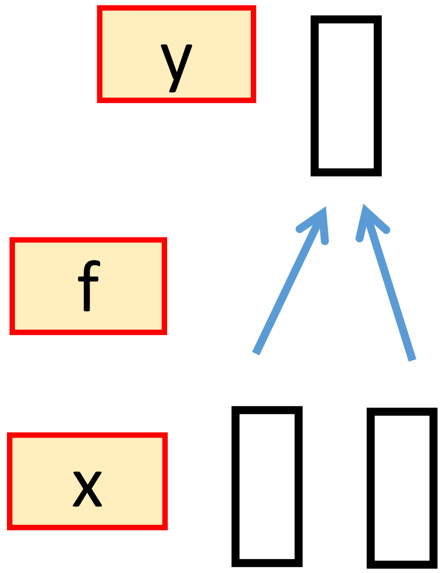
Merlin.jl: Imperative Mode

```
x = Variable(rand(Float32,100,1))
f = Linear(Float32, 100, 50)
y = x |> f

function call(f, x)
  y, diff = apply(f, x)
  Variable(y, diff, ...)
end
```



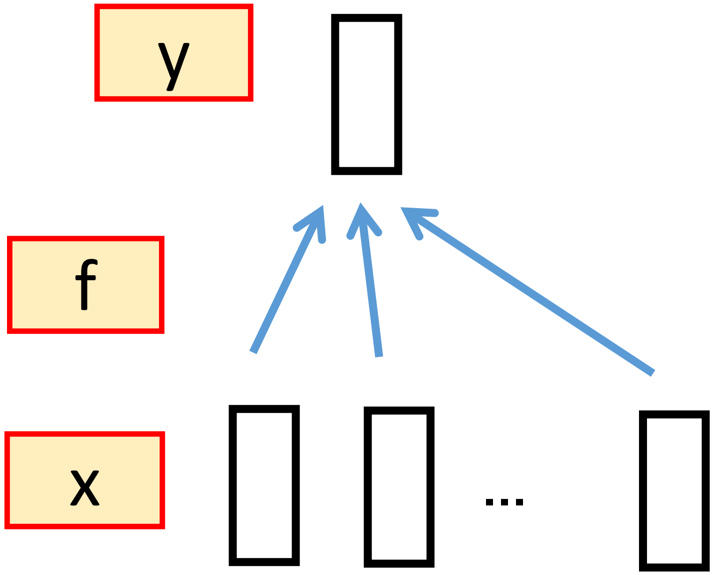
Merlin.jl: Imperative Mode

```
x1, x2 = Variable(…), Variable(…), …
y = (x1, x2) |> Add()
```



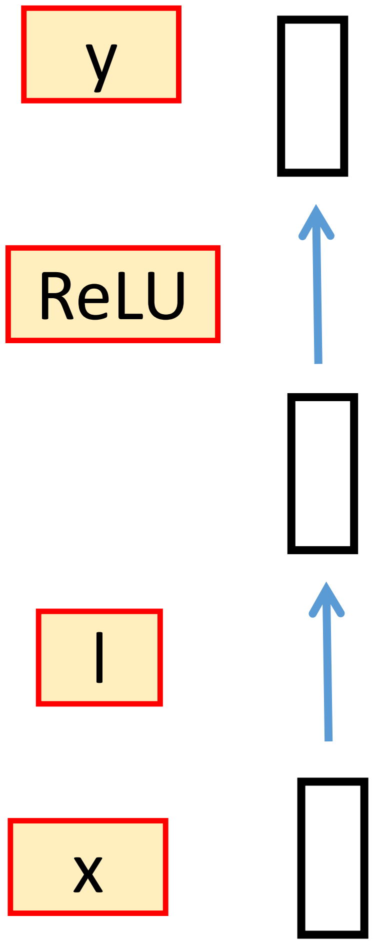
Merlin.jl: Imperative Mode

```
x1, x2, x3 = Variable(…), Variable(…), …
y = [x1, x2, x3] |> Concat(1)
```



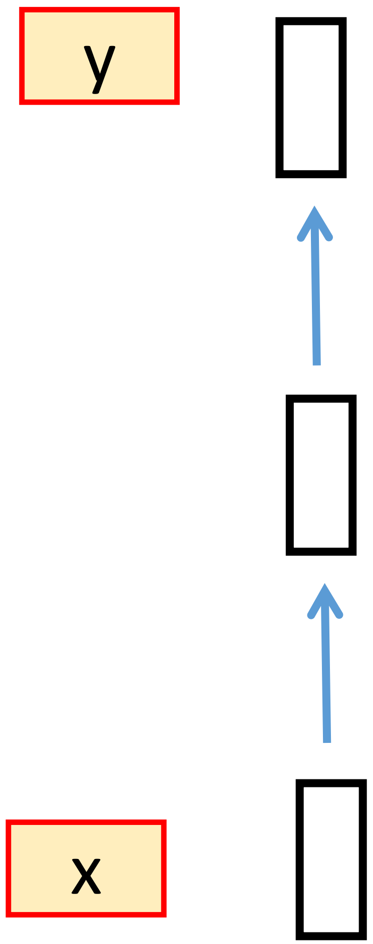
Merlin.jl: Imperative Mode

```
x = Variable(...)
l = Linear(Float32, 100, 50)

y = x |> l |> ReLU()
or
y = x |> [l, ReLU()]
```

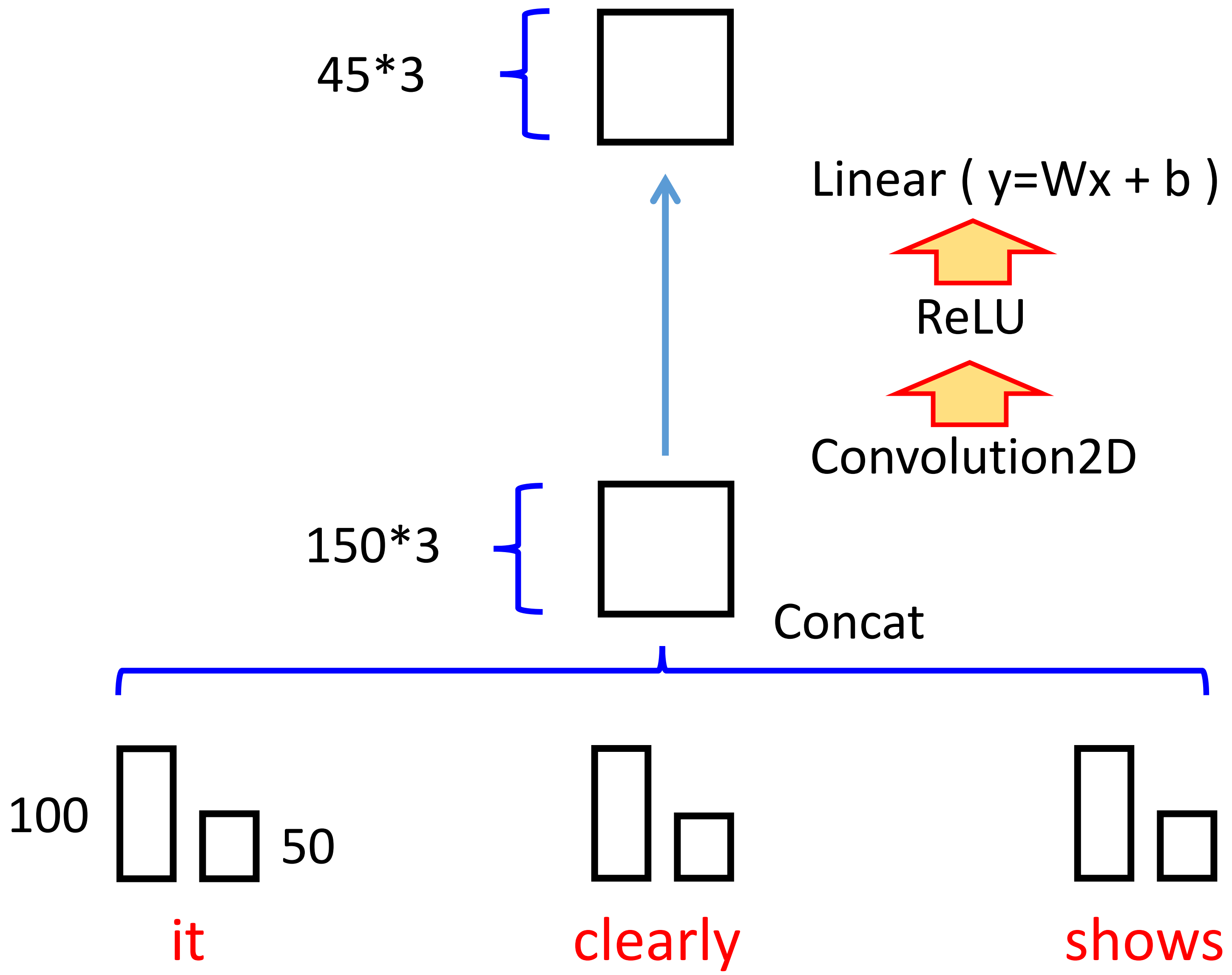


Merlin.jl: Symbolic Mode

```
x = Variable()
y = x |> f1 |> f2 |> f3
g = compile(y)
```



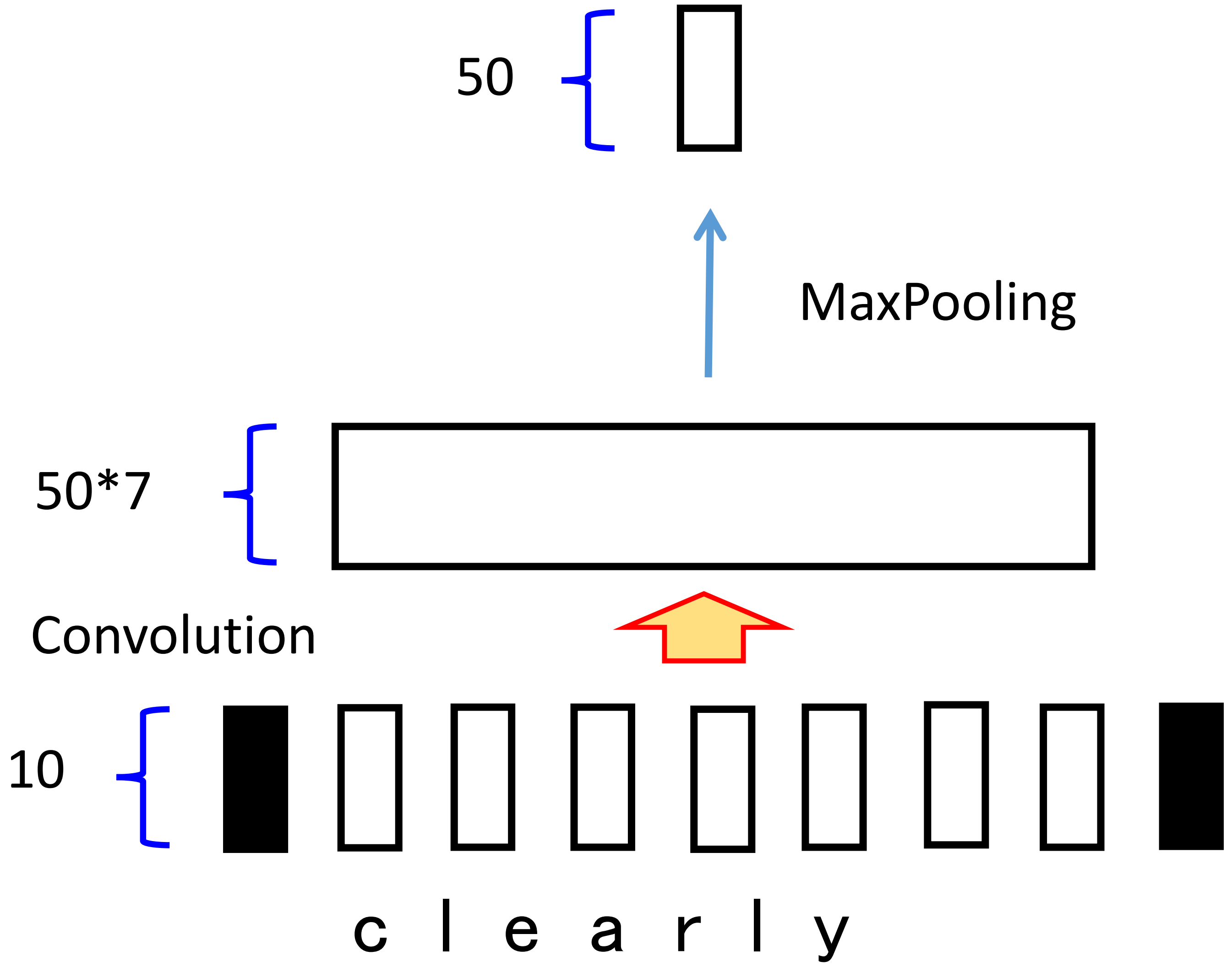
Sentence-level Forward



Sentence-level Forward

```
# functors definition
word_embed = Lookup(Float32, 100)
char_embed = Lookup(Float32, 10)
conv = Conv2d(...)
linear = Linear(Float32, 100, 45)

# forward
output = [vec1, vec2, vec3]
         |> Concat(2)
         |> [conv, ReLU(), linear]
loss = output |> CrossEntropy(...)

# backward
diff!(output, loss)

# update
optimize!(sgd, [word_embed, char_embed, conv, linear])
```

Character-level Forward



Character-level Forward

```
# functors definition
conv = Conv2d(…)
maxpool = MaxPool2d(…)

# forward
for i = 1:length(words)
  chars = words[i].chars
  output[i] = chars |> [conv, maxpool]
end
```

Merlin.jl: List of Functors

- Activation (ReLU, Tanh, Sigmoid)
- Concat
- Convolution2D
- Linear ($y = Wx + b$)
- Lookup Table
- Math Operators (Add, Multiply)
- MaxPooling2D
- Window2D

Experimental Results

- Training: Penn Treebank WSJ, sec. 00-18
- Testing: 〃, sec. 23
- Optimizer: SGD (learning rate = 0.0075)

Method	Accuracy
Neural POS-Tagger	97.28
Stanford NLP (MEMM)	97.24 ～ 97.32

Future Work

- GPU (cuDNN)
- Benchmark test
- Add more examples for NLP