Let the AI do the Talk

Adventures with Natural Language Generation

@MarcoBonzanini

PyParis 2018

PyData
London

NUMFOCUS
OPEN CODE = BETTER SCIENCE

PyData London Conference

12-14 July 2019

`@PyDataLondon`

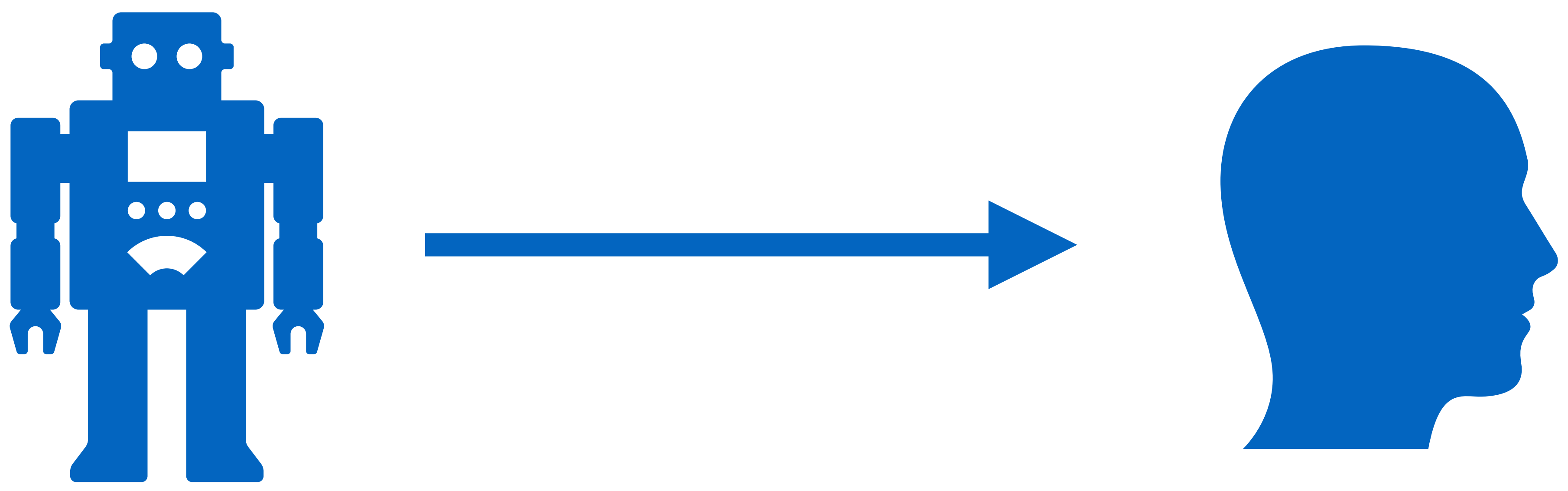
NATURAL LANGUAGE GENERATION

Natural Language Processing

Natural Language Processing

Natural Language Understanding

Natural Language Generation



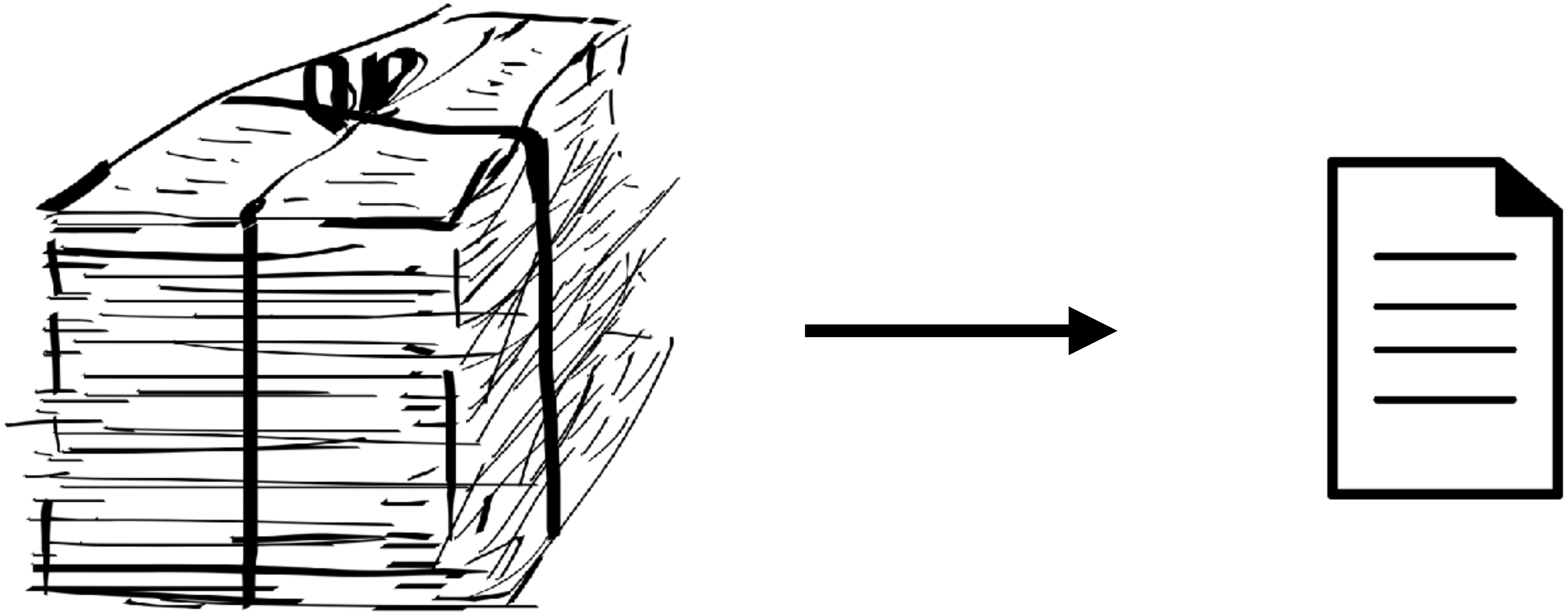
Natural Language Generation

Natural Language Generation

The task of generating
Natural Language
from a machine representation

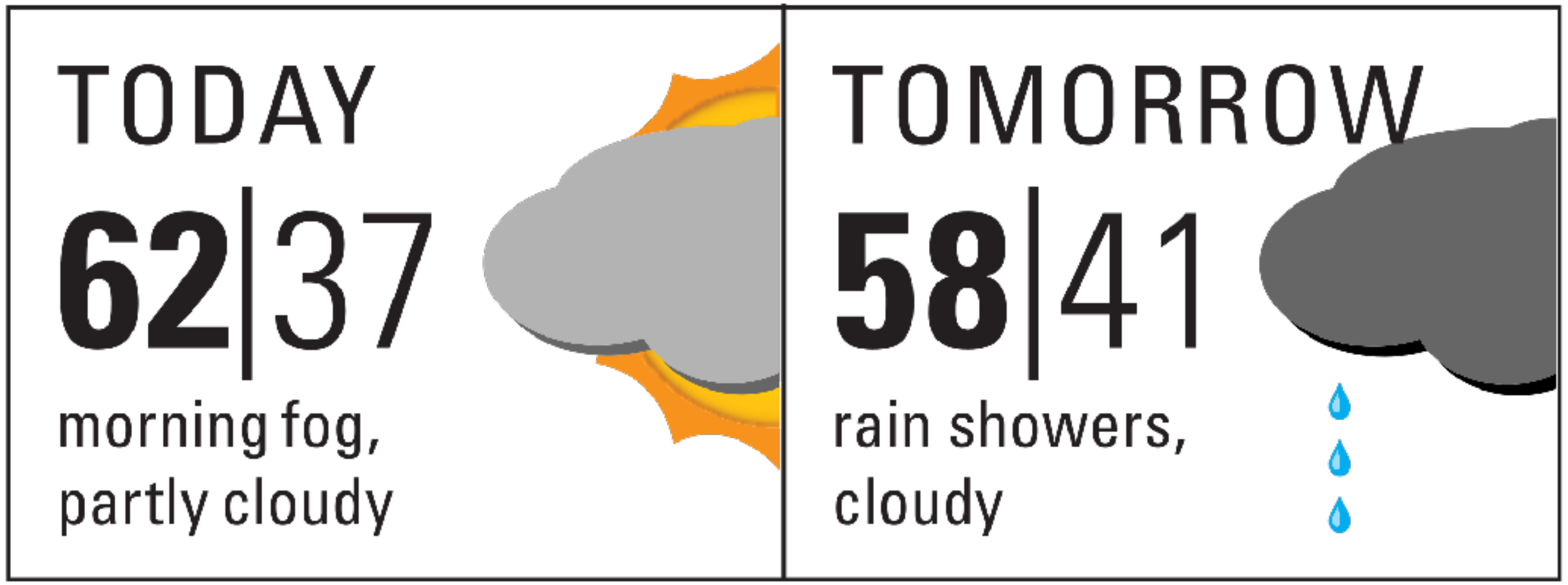
Applications of NLG

Applications of NLG



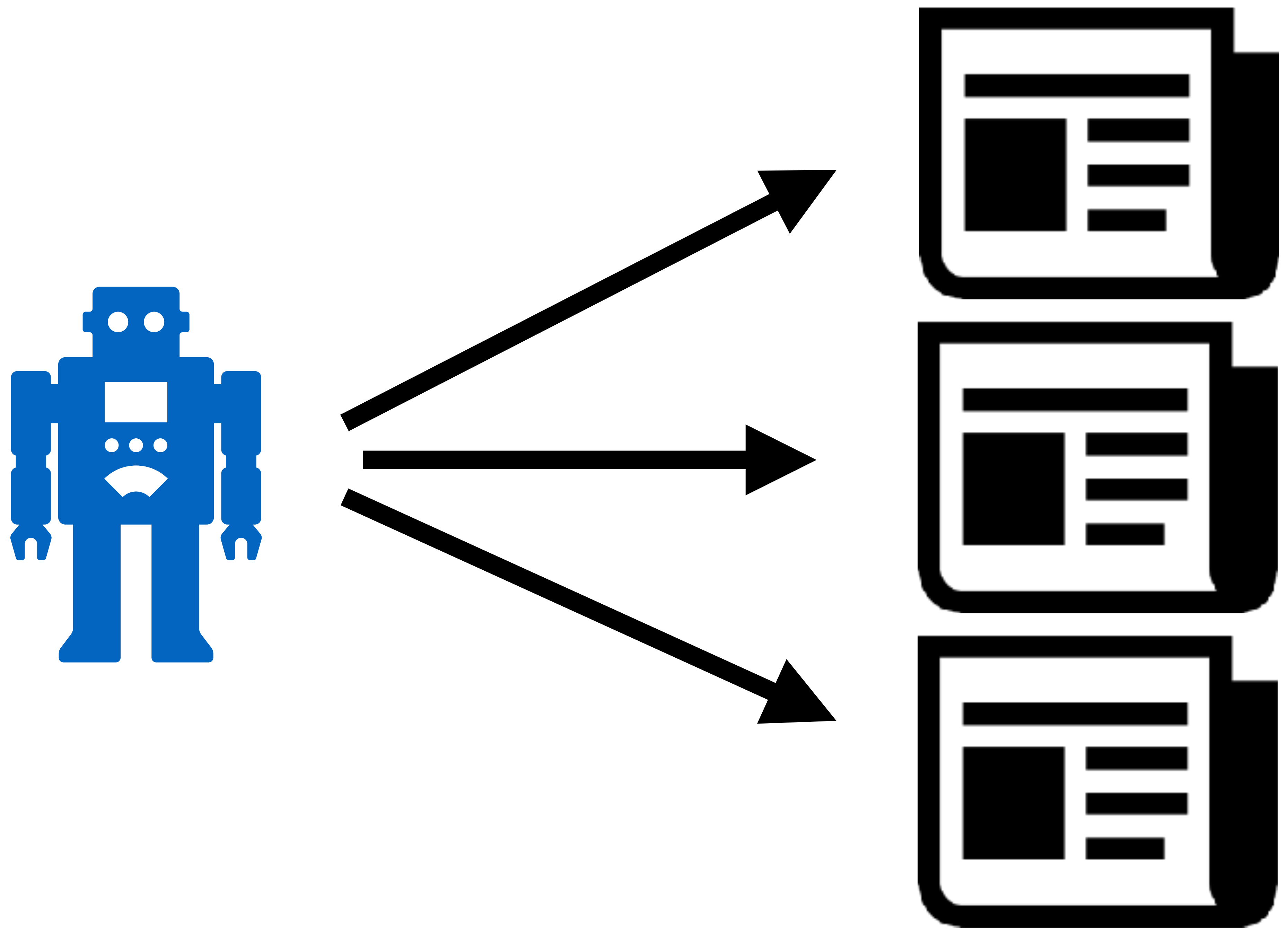
Summary Generation

Applications of NLG



Weather Report Generation

Applications of NLG



Automatic Journalism

Applications of NLG

Virtual Assistants / Chatbots

LANGUAGE MODELLING

Language Model

Language Model

A model that gives you
the probability of
a sequence of words

Language Model

P(I’m going home)

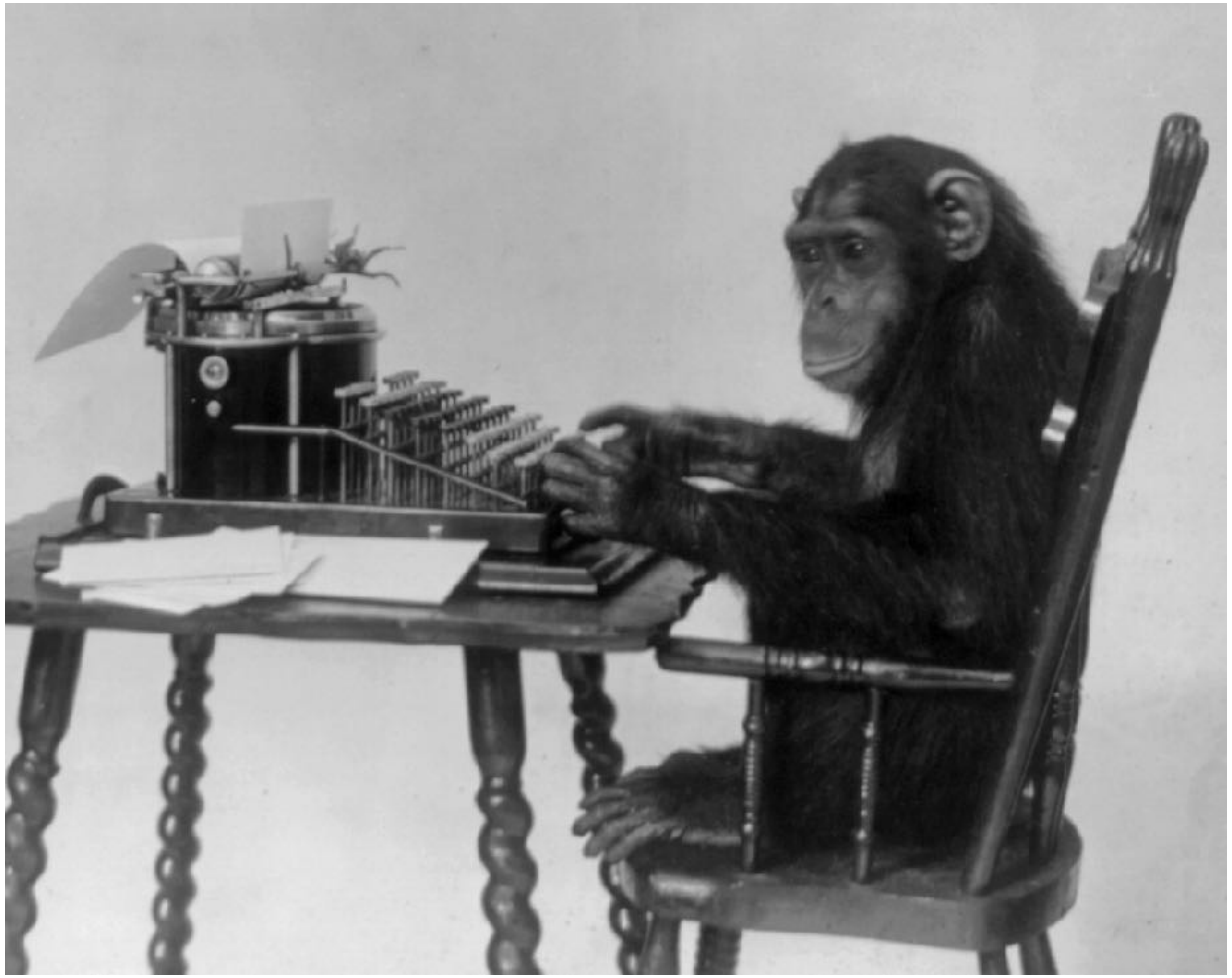
>

P(Home I’m going)

Language Model

P(I’m going home)
>
P(I’m going house)

Infinite Monkey Theorem



https://en.wikipedia.org/wiki/Infinite_monkey_theorem

Infinite Monkey Theorem

```
from random import choice
from string import printable

def monkey_hits_keyboard(n):
    output = [choice(printable) for _ in range(n)]
    print("The monkey typed:")
    print(''.join(output))
```

Infinite Monkey Theorem

```
>>> monkey_hits_keyboard(30)
The monkey typed:
%
a9AK^YKx    OkVG)u3.cQ,31("!ac%

>>> monkey_hits_keyboard(30)
The monkey typed:
fWE,ou)cxmV2IZ  l}jSV'XxQ**9'|
```

n-grams

n-grams

Sequence on N items
from a given sample of text

n-grams

```
>>> from nltk import ngrams
>>> list(ngrams("pizza", 3))
```

n-grams

```
>>> from nltk import ngrams
>>> list(ngrams("pizza", 3))
[('p', 'i', 'z'), ('i', 'z', 'z'),
('z', 'z', `a')]
```

n-grams

```
>>> from nltk import ngrams
>>> list(ngrams("pizza", 3))
[('p', 'i', 'z'), ('i', 'z', 'z'),
('z', 'z', `a')]
```

character-based trigrams

n-grams

```
>>> s = "The quick brown fox".split()
>>> list(ngrams(s, 2))
```

n-grams

```
>>> s = "The quick brown fox".split()
>>> list(ngrams(s, 2))
[('The', 'quick'), ('quick', 'brown'),
('brown', 'fox')]
```

n-grams

```
>>> s = "The quick brown fox".split()
>>> list(ngrams(s, 2))
[('The', 'quick'), ('quick', 'brown'),
('brown', 'fox')]
```

word-based bigrams

From n-grams to Language Model

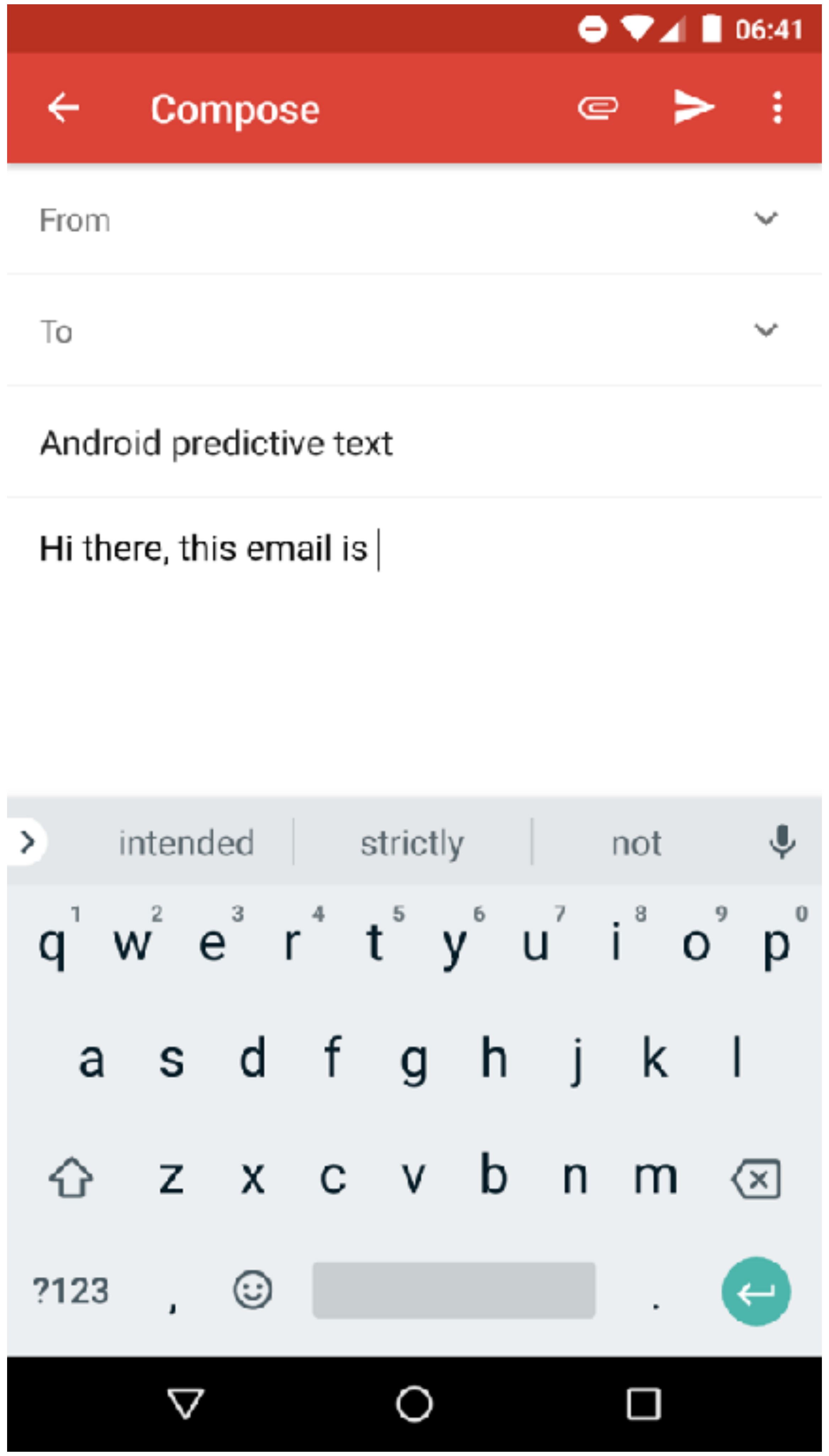
From n-grams to Language Model

- Given a large dataset of text
- Find all the n-grams
- Compute probabilities, e.g. count bigrams:

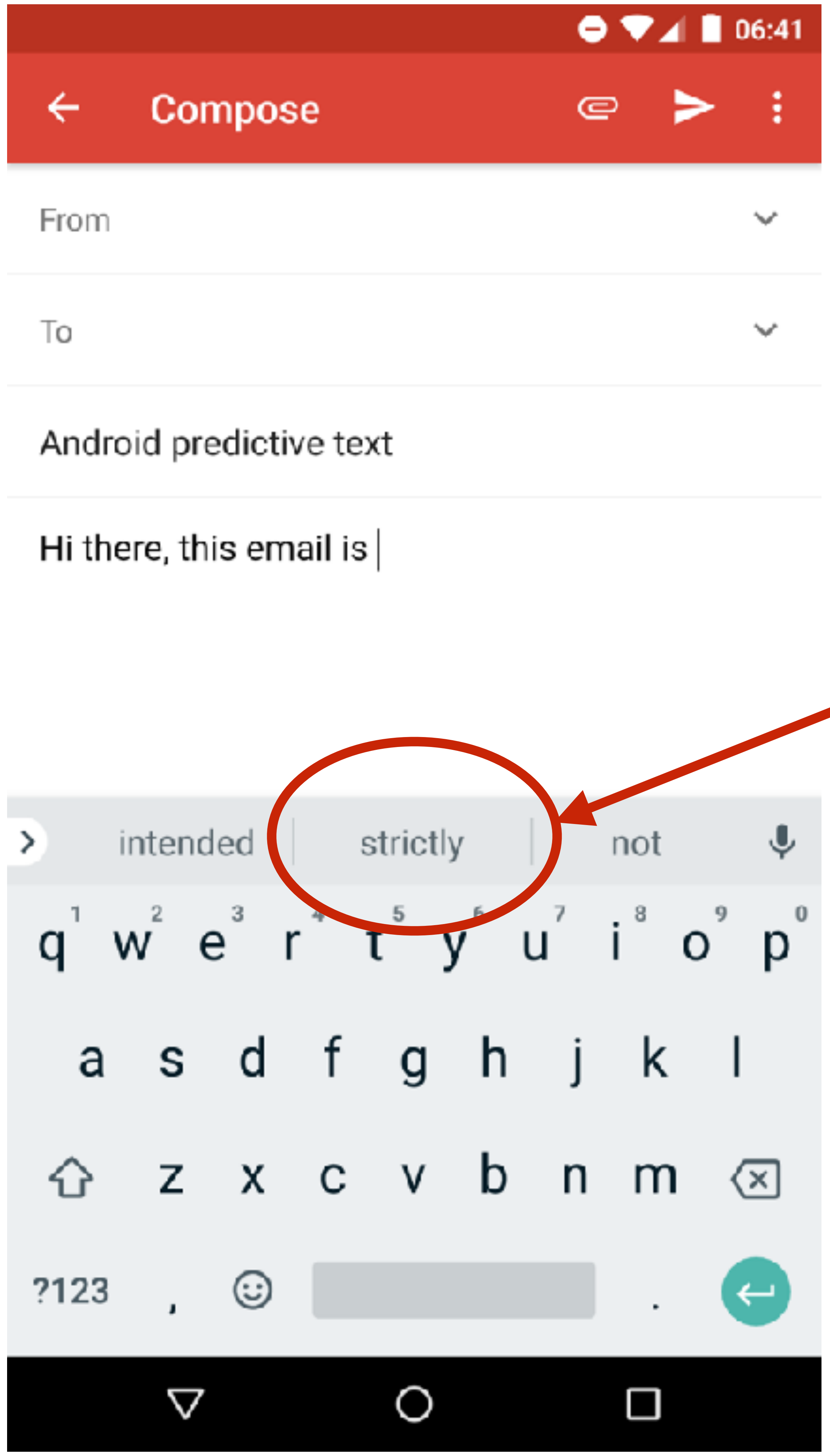
$$P(\text{fox}|\text{brown}) = \frac{P(\text{brown}, \text{fox})}{P(\text{brown})}$$

Example: Predictive Text in Mobile

Example: Predictive Text in Mobile

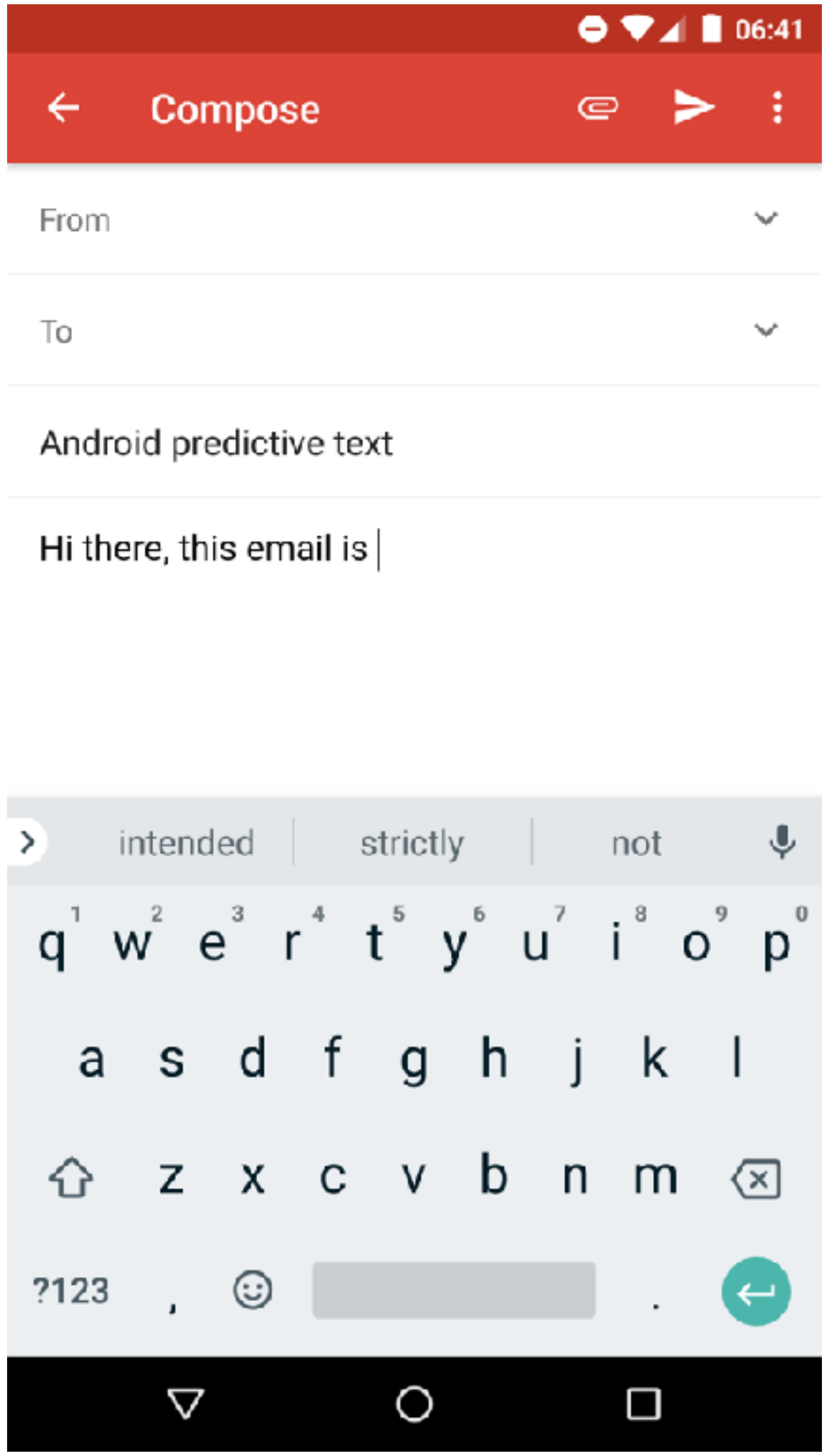


Example: Predictive Text in Mobile



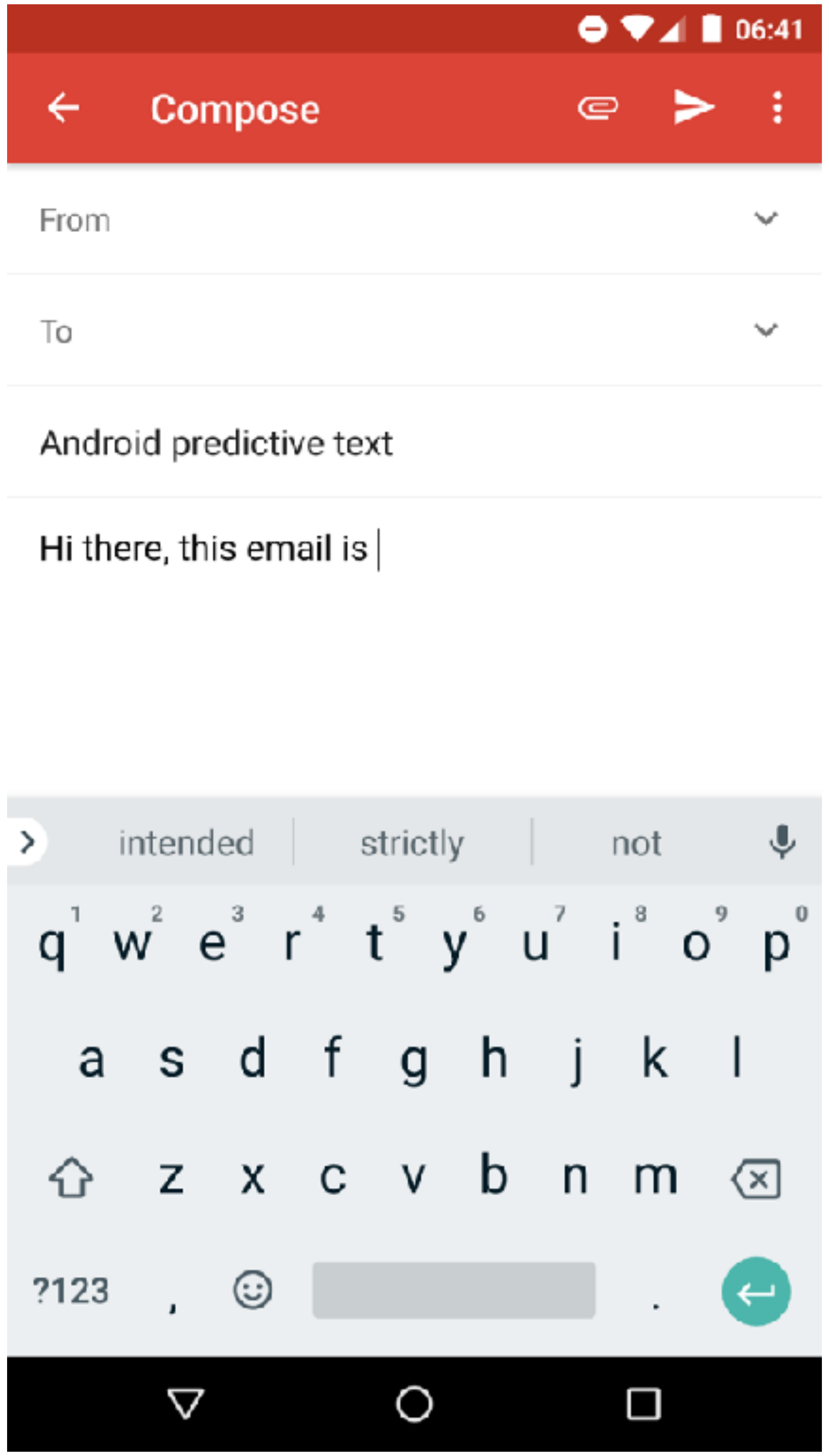
most likely next word

Example: Predictive Text in Mobile



Marco is …

Example: Predictive Text in Mobile



Marco is a good time to get the latest flash player is required for video playback is unavailable right now because this video is not sure if you have a great day.

Limitations of LM so far

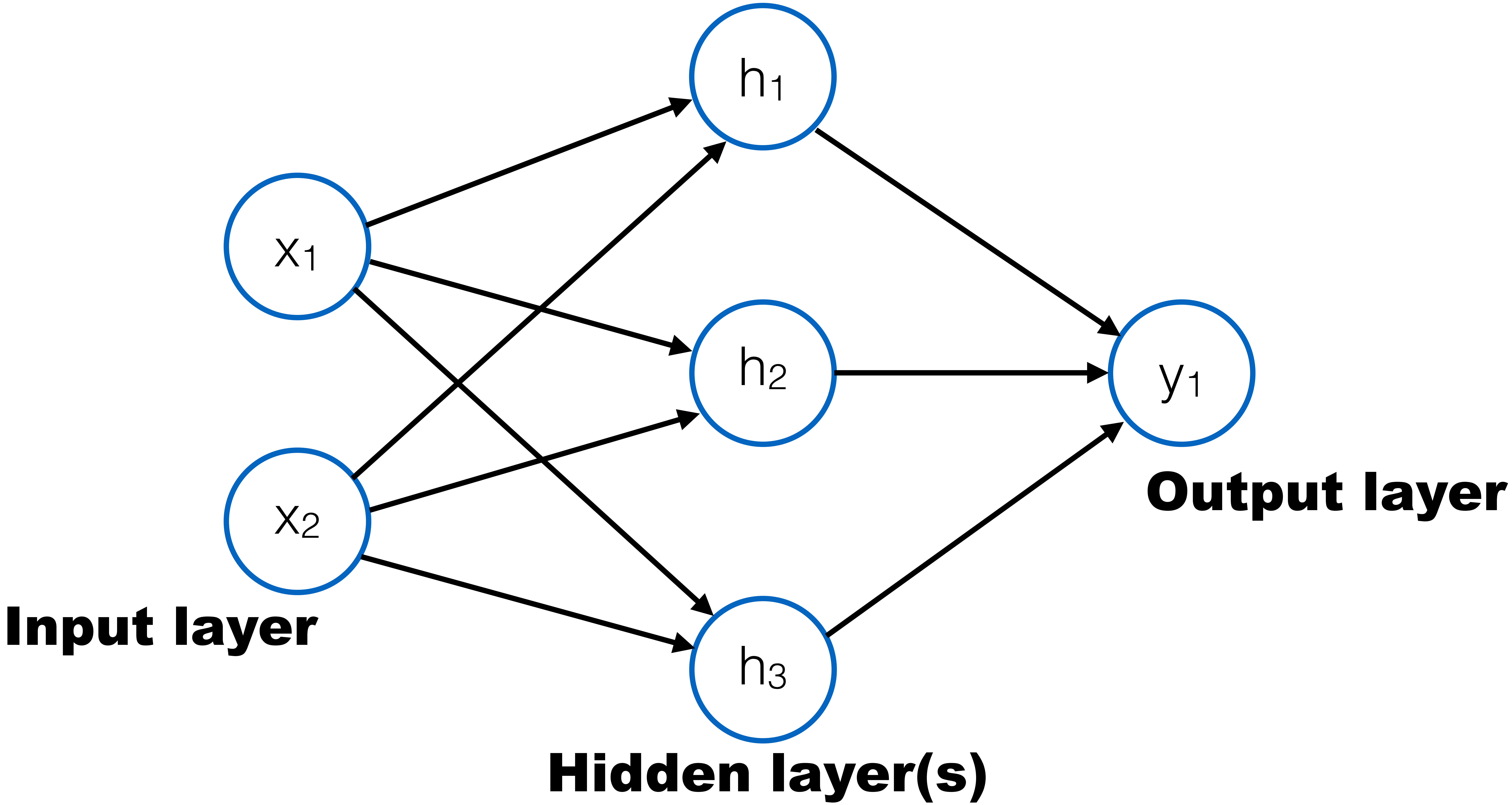
Limitations of LM so far

- P(word | full history) is too expensive
- P(word | previous few words) is feasible
- … Local context only! Lack of global context

QUICK INTRO TO NEURAL NETWORKS

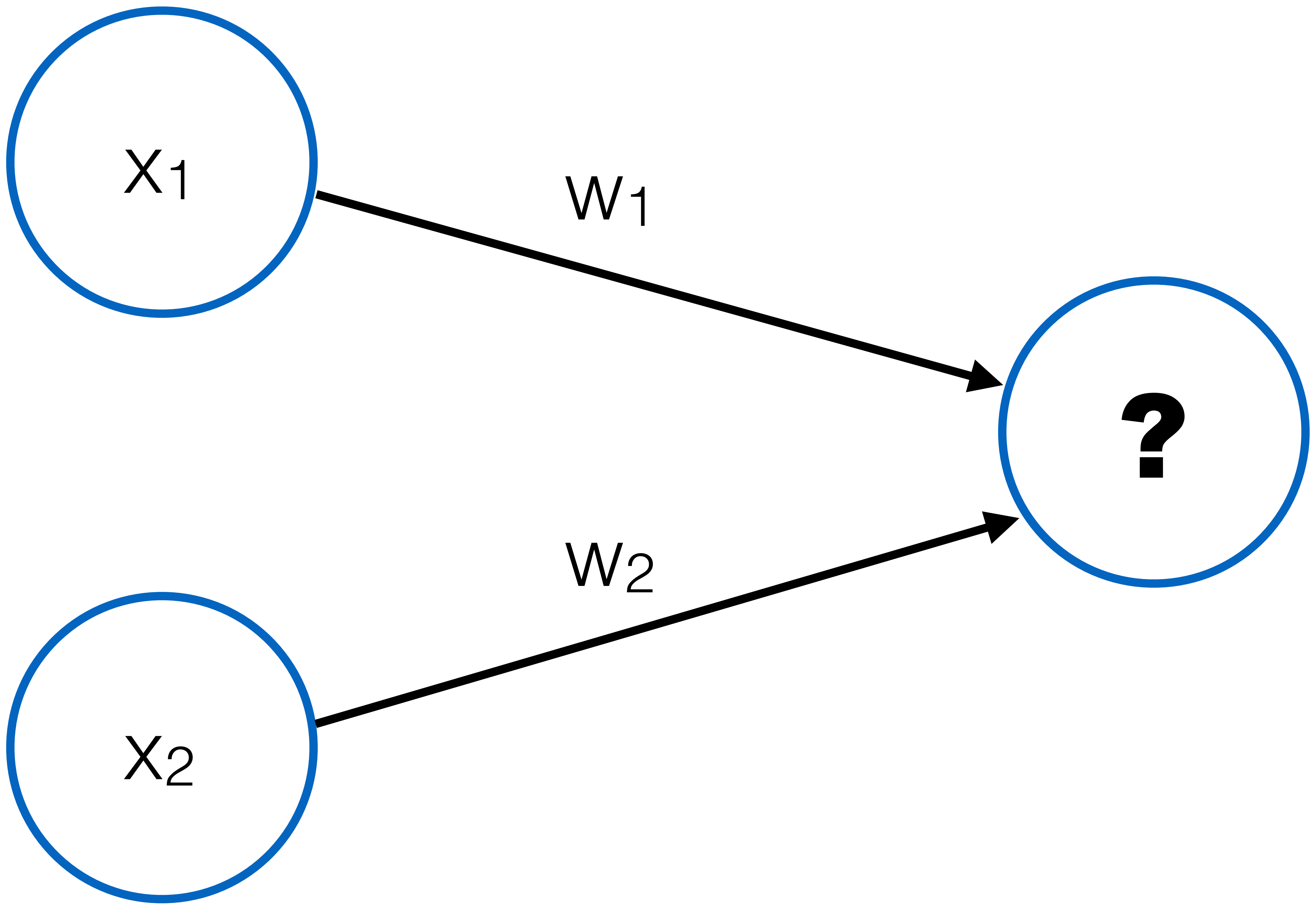
Neural Networks

Neural Networks

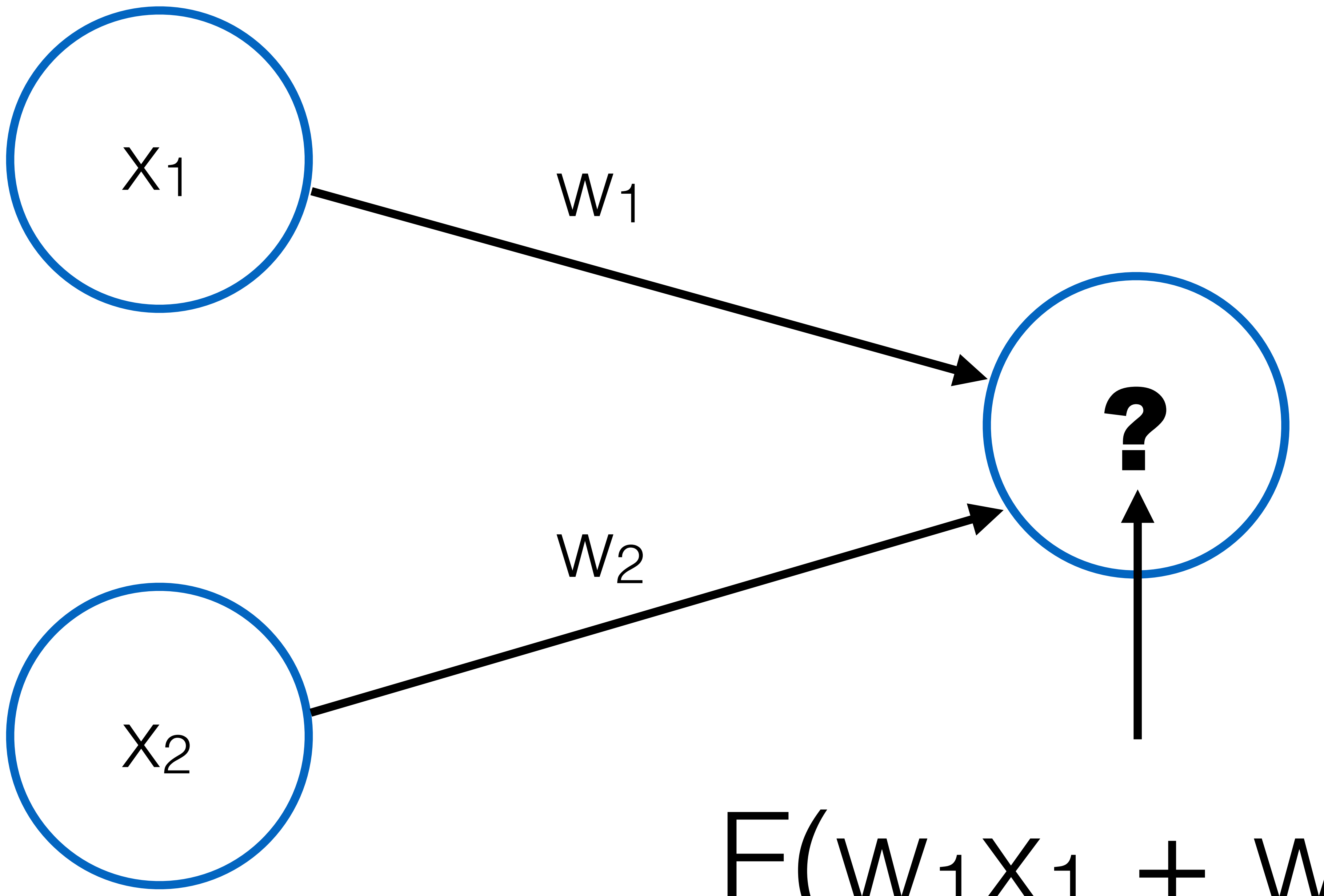


Neurone Example

Neurone Example



Neurone Example

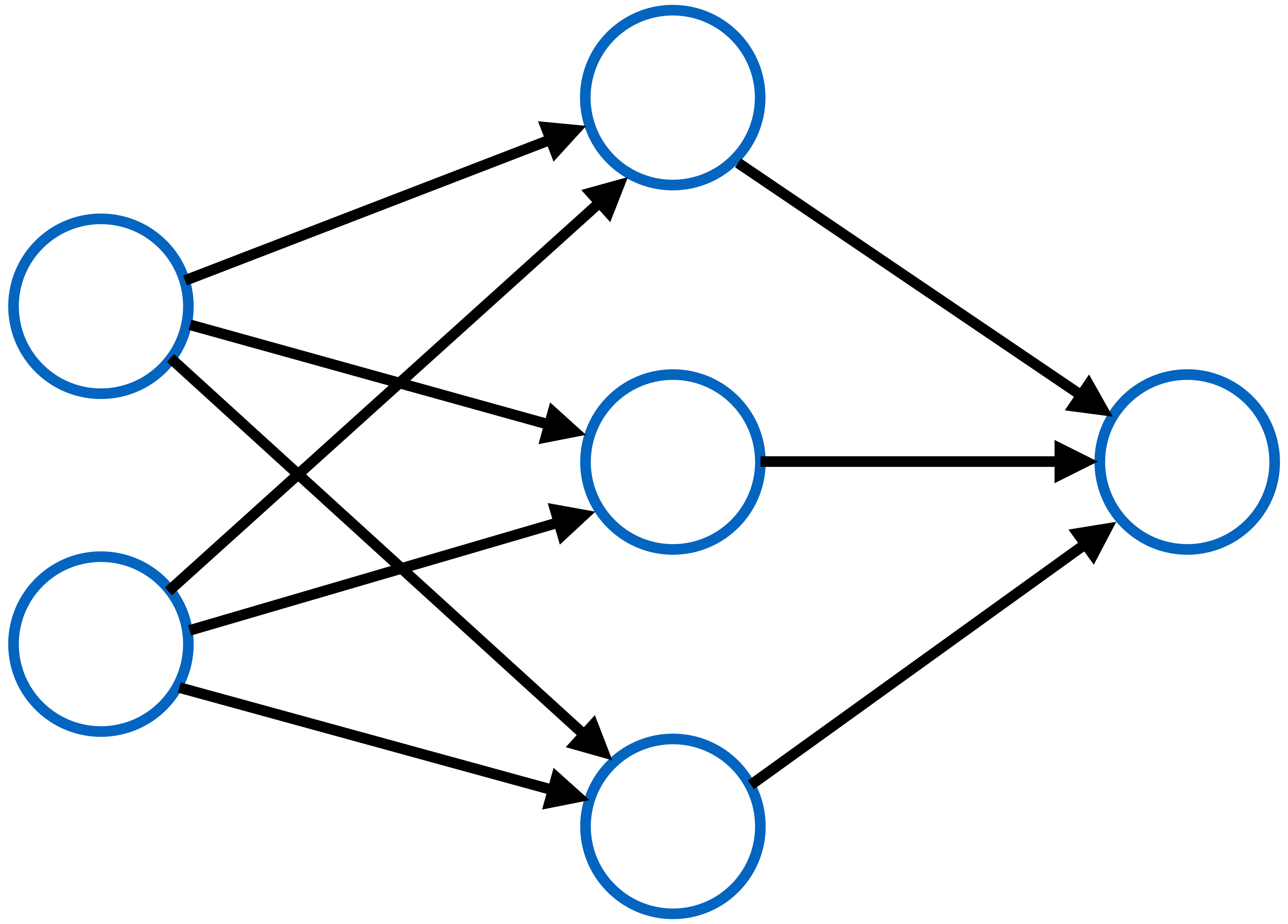


$$F(w_1x_1 + w_2x_2)$$

Training the Network

Training the Network

- Random weight init
- Run input through the network
- Compute error
  (loss function)
- Use error to adjust weights
  (gradient descent + back-propagation)



More on Training

More on Training

- Batch size
- Iterations and Epochs
- e.g. 1,000 data points, if batch size = 100 we need 10 iterations to complete 1 epoch

RECURRENT NEURAL NETWORKS

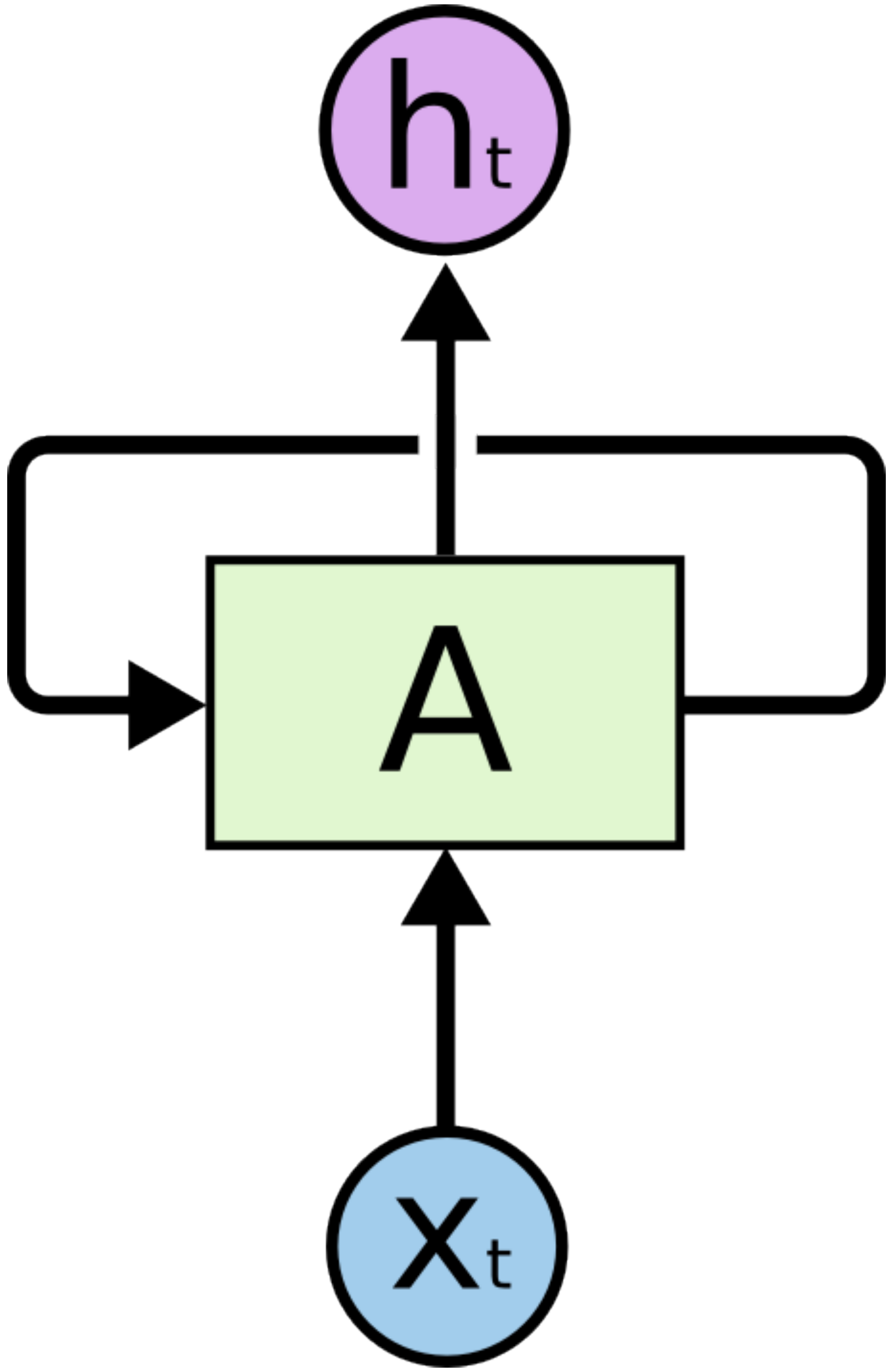
Limitation of FFNN

Limitation of FFNN

Input and output
of fixed size

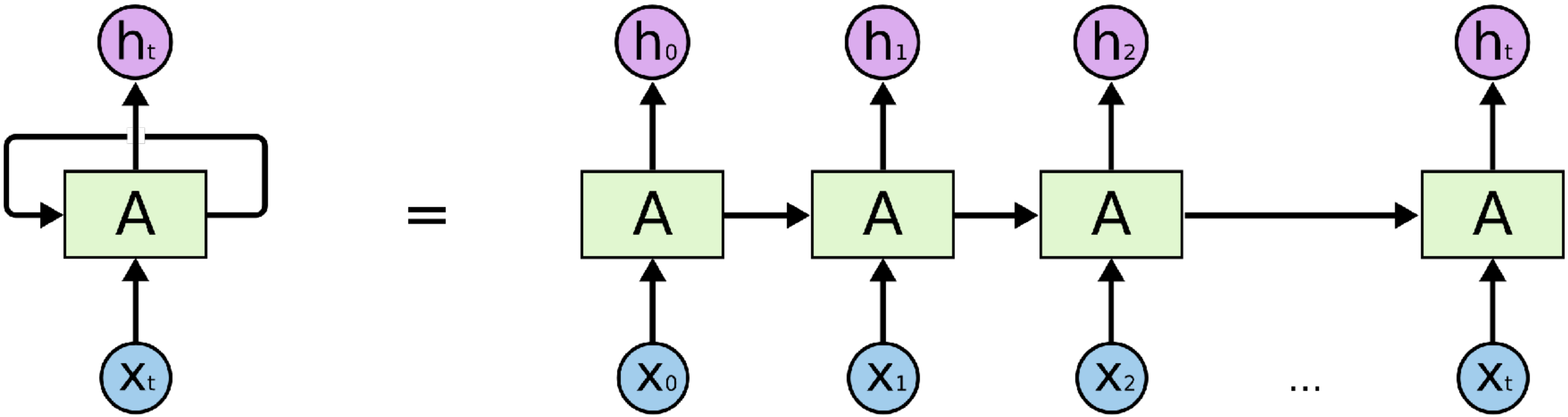
Recurrent Neural Networks

Recurrent Neural Networks



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Limitation of RNN

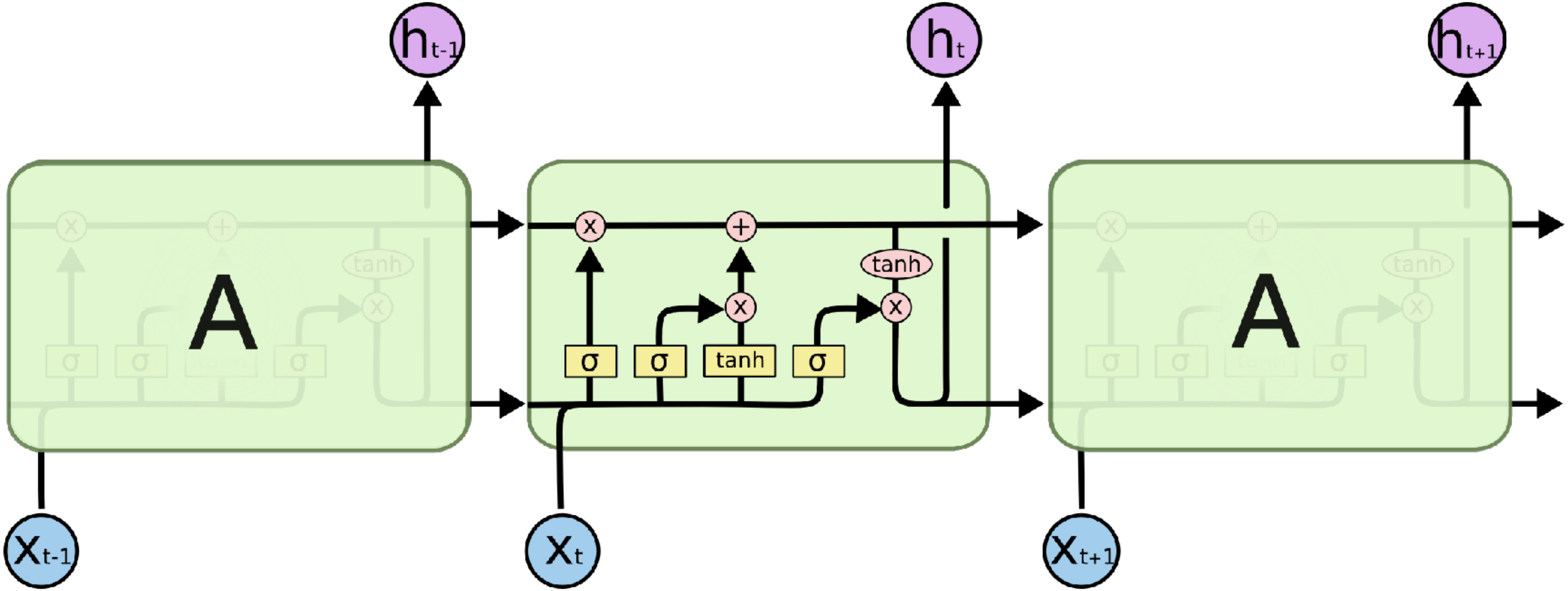
Limitation of RNN

“Vanishing gradient”

Cannot “remember”
what happened long ago

Long Short Term Memory

Long Short Term Memory



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM with a forget gate [edit]

The compact forms of the equations for the forward pass of an LSTM unit with a forget gate are:[1][2]

$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}$$

where the initial values are $c_0 = 0$ and $h_0 = 0$ and the operator $\circ$ denotes the Hadamard product (element-wise product). The subscript t indexes the time step.

Variables [edit]

- $x_t \in \mathbb{R}^d$: input vector to the LSTM unit
- $f_t \in \mathbb{R}^h$: forget gate's activation vector
- $i_t \in \mathbb{R}^h$: input gate's activation vector
- $o_t \in \mathbb{R}^h$: output gate's activation vector
- $h_t \in \mathbb{R}^h$: output vector of the LSTM unit
- $c_t \in \mathbb{R}^h$: cell state vector
- $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: weight matrices and bias vector parameters which need to be learned during training

where the superscripts d and h refer to the number of input features and number of hidden units, respectively.

Activation functions [edit]

- σ_g: sigmoid function.
- σ_c: hyperbolic tangent function.
- σ_h: hyperbolic tangent function or, as the peephole LSTM paper[which?] suggests, $\sigma_h(x) = x$.[17][18]

https://en.wikipedia.org/wiki/Long_short-term_memory

A BIT OF PRACTICE

Deep Learning in Python

Deep Learning in Python

- Some NN support in scikit-learn
- Many low-level frameworks: Theano, PyTorch, TensorFlow
- … Keras!
- Probably more

Keras

Keras

- Simple, high-level API
- Uses TensorFlow, Theano or CNTK as backend
- Runs seamlessly on GPU
- Easier to start with

LSTM Example

LSTM Example

Define the network

```
model = Sequential()
model.add(
    LSTM(
        128,
        input_shape=(maxlen,len(chars))
    )
)
model.add(Dense(len(chars), activation='softmax'))
```

LSTM Example

Configure the network

```
optimizer = RMSprop(lr=0.01)
model.compile(
    loss='categorical_crossentropy',
    optimizer=optimizer
)
```

LSTM Example

Train the network

```
model.fit(x, y,
          batch_size=128,
          epochs=60,
          callbacks=[print_callback])

model.save('char_model.h5')
```

LSTM Example

Generate text

```
for i in range(output_size):
    ...

    preds = model.predict(x_pred, verbose=0)[0]
    next_index = sample(preds, diversity)
    next_char = indices_char[next_index]

    generated += next_char
```

LSTM Example

Seed text

```
for i in range(output_size):
    ...

    preds = model.predict(x_pred, verbose=0)[0]
    next_index = sample(preds, diversity)
    next_char = indices_char[next_index]

    generated += next_char
```

Sample Output

Sample Output

Seed text **After 1 epoch**

```
 are the glories it included.
Now am I lrA to r    ,d?ot praki ynhh
kpHu ndst -h ahh
umk,hrfheleuloluprffuamdaedospe
aeooasak sh frxpaphrNumlpAryoaho (…)
```

Sample Output

After ~5 epochs

```
I go from thee:
Bear me forthwitht wh, t
che f uf ld,hhorfAs c c ff.h
scfylhle, rigrya p s lee
rmoy, tofhryg dd?ofr hl t y
ftrhoodfe- r  Py (...)
```

Sample Output

After 20+ epochs

```
a wild-goose flies,
Unclaim'd of any manwecddeelc uavekeMw
gh  whacelcwiiaeh  xcacwiDac  w
fioarw ewoc h feicucra
 h,h,  :ewh utiqitilweWy ha.h  pc'hr,
lagfh
eIwislw ofiridete w
laecheefb .ics,aicpaweteh fiw?egp t? (...)
```

Tuning

Tuning

- More layers?
- More hidden nodes? or less?
- More data?
- A combination?

Tuning

After 1 epoch

```
Wyr feirm hat. meancucd kreukk?
, foremee shiciarplle. My,
Bnyivlaunef sough bus:
Wad vomietlhas nteos thun. lore
orain, Ty thee I Boe,
I rue. niat
```

Tuning

Much later

```
to Dover, where inshipp'd
Commit them to plean me than stand and the
woul came the wife marn to the groat pery me
Which that the senvose in the sen in the poor
The death is and the calperits the should
```

FINAL REMARKS

A Couple of Tips

A Couple of Tips

- You’ll need a GPU
- Develop locally on very small dataset
  then run on cloud on real data
- At least 1M characters in input,
  at least 20 epochs for training
- **`model.save()`** !!!

Summary

- Natural Language Generation is fun
- Simple models vs. Neural Networks
- Keras makes your life easier
- A lot of trial-and-error!

THANK YOU

`@MarcoBonzanini`

`speakerdeck.com/marcobonzanini`

Readings & Credits

- Brandon Rohrer on "Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)": `https://www.youtube.com/watch?v=WCUNPb-5EYI`
- Chris Olah on Understanding LSTM Networks: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`
- Andrej Karpathy on "The Unreasonable Effectiveness of Recurrent Neural Networks": `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

Pics:
- Weather forecast icon: https://commons.wikimedia.org/wiki/File:Newspaper_weather_forecast_-_today_and_tomorrow.svg
- Stack of papers icon: https://commons.wikimedia.org/wiki/File:Stack_of_papers_tied.svg
- Document icon: https://commons.wikimedia.org/wiki/File:Document_icon_(the_Noun_Project_27904).svg
- News icon: https://commons.wikimedia.org/wiki/File:PICOL_icon_News.svg
- Cortana icon: https://upload.wikimedia.org/wikipedia/commons/thumb/8/89/Microsoft_Cortana_light.svg/1024px-Microsoft_Cortana_light.svg.png
- Siri icon: https://commons.wikimedia.org/wiki/File:Siri_icon.svg
- Google assistant icon: https://commons.wikimedia.org/wiki/File:Google_mic.svg