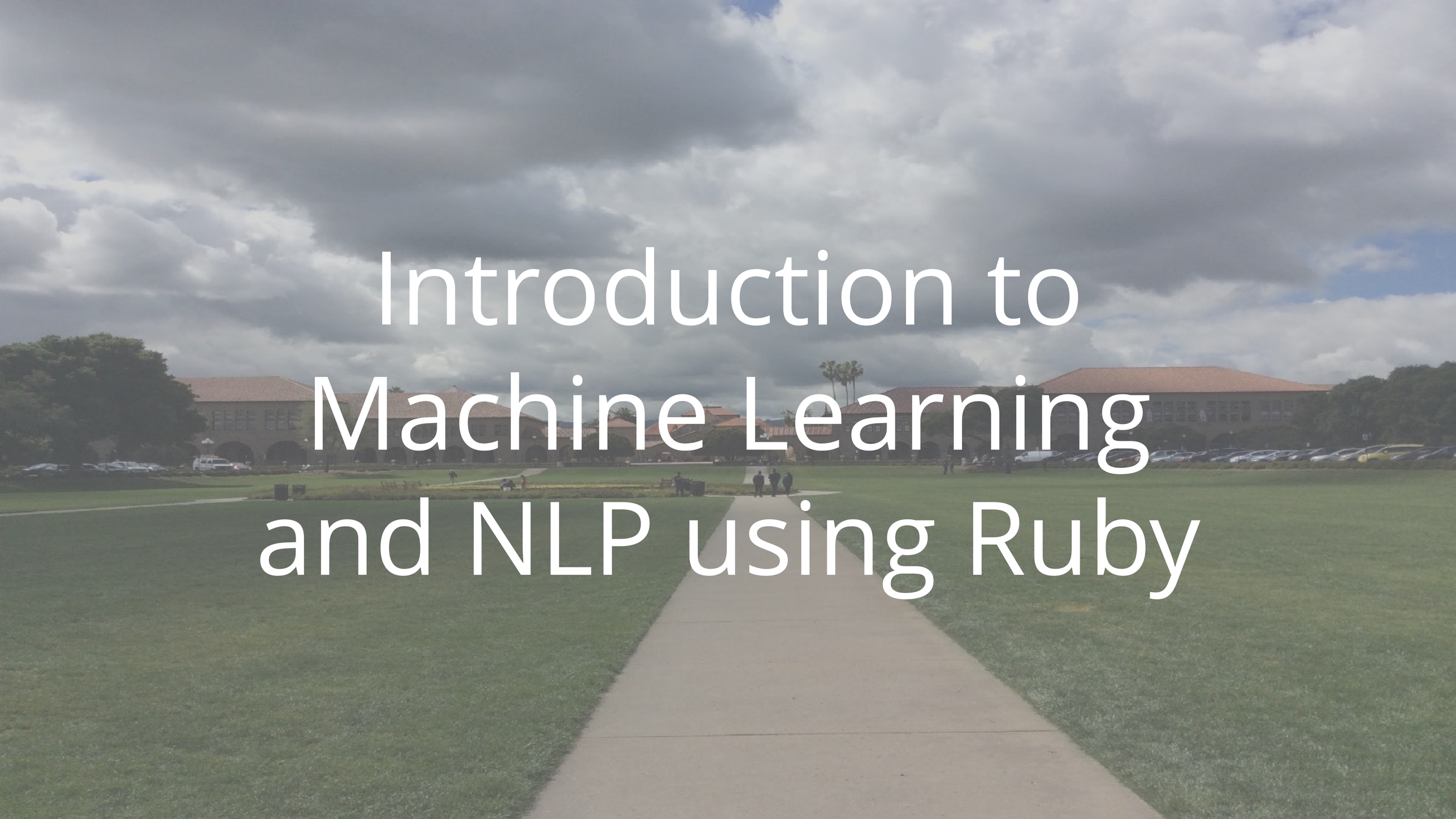


Introduction to Machine Learning and NLP using Ruby

Thanks Athens!

Erik Mathiesen

Octavia.ai | mancml.io | erik@octavia.ai

Outline

- Part 1: Machine Learning
  - Octavia.ai
  - Machine Learning
    - History
    - Now!
  - Deep Learning
    - Neural Nets
    - Word Embeddings
    - Deep Neural Nets
- Part 2: Ruby
  - Our Tech Stack
  - Ruby and Machine Learning
  - Ruby C-API
    - Creating a gem
    - C-extension gem
  - Tips & Tricks
  - Jupiter

Part I: Machine Learning

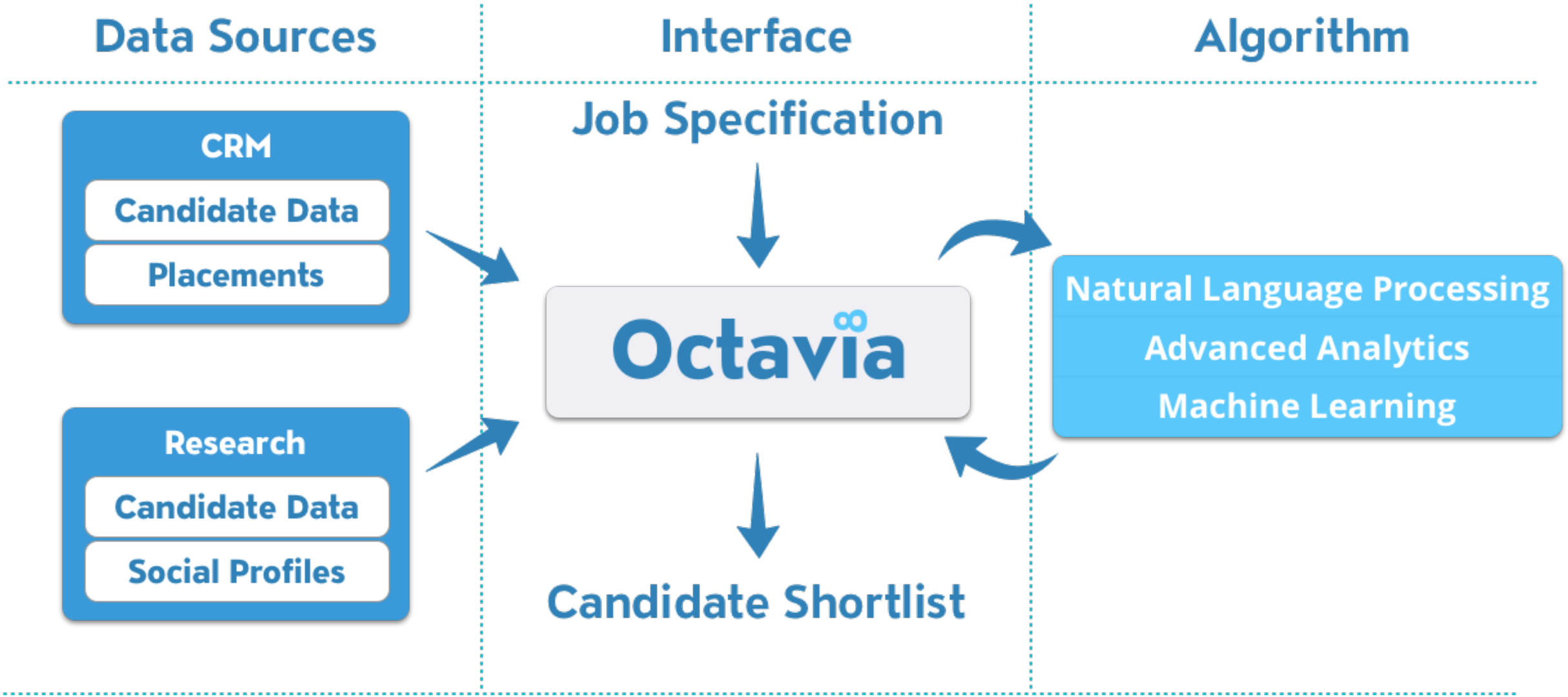

Octavia

Faster, Smarter Recruitment

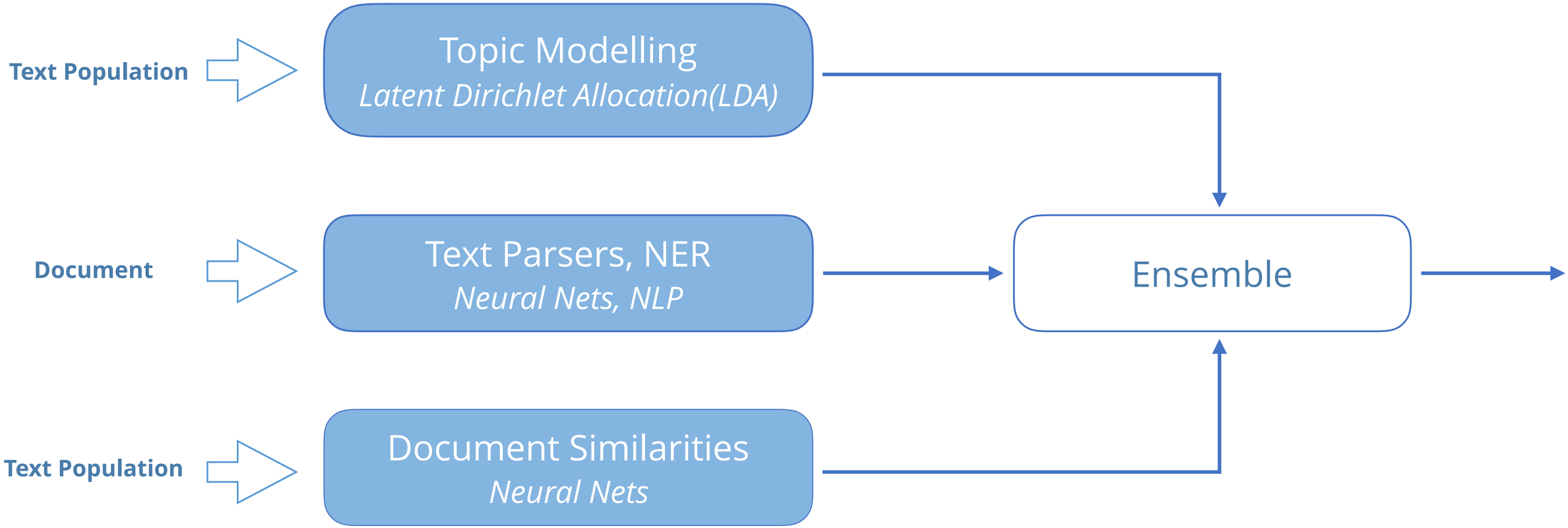
Problem: Finding the best candidates for a job in the fastest time possible

Aim: '*Solve*' this using NLP and Machine Learning

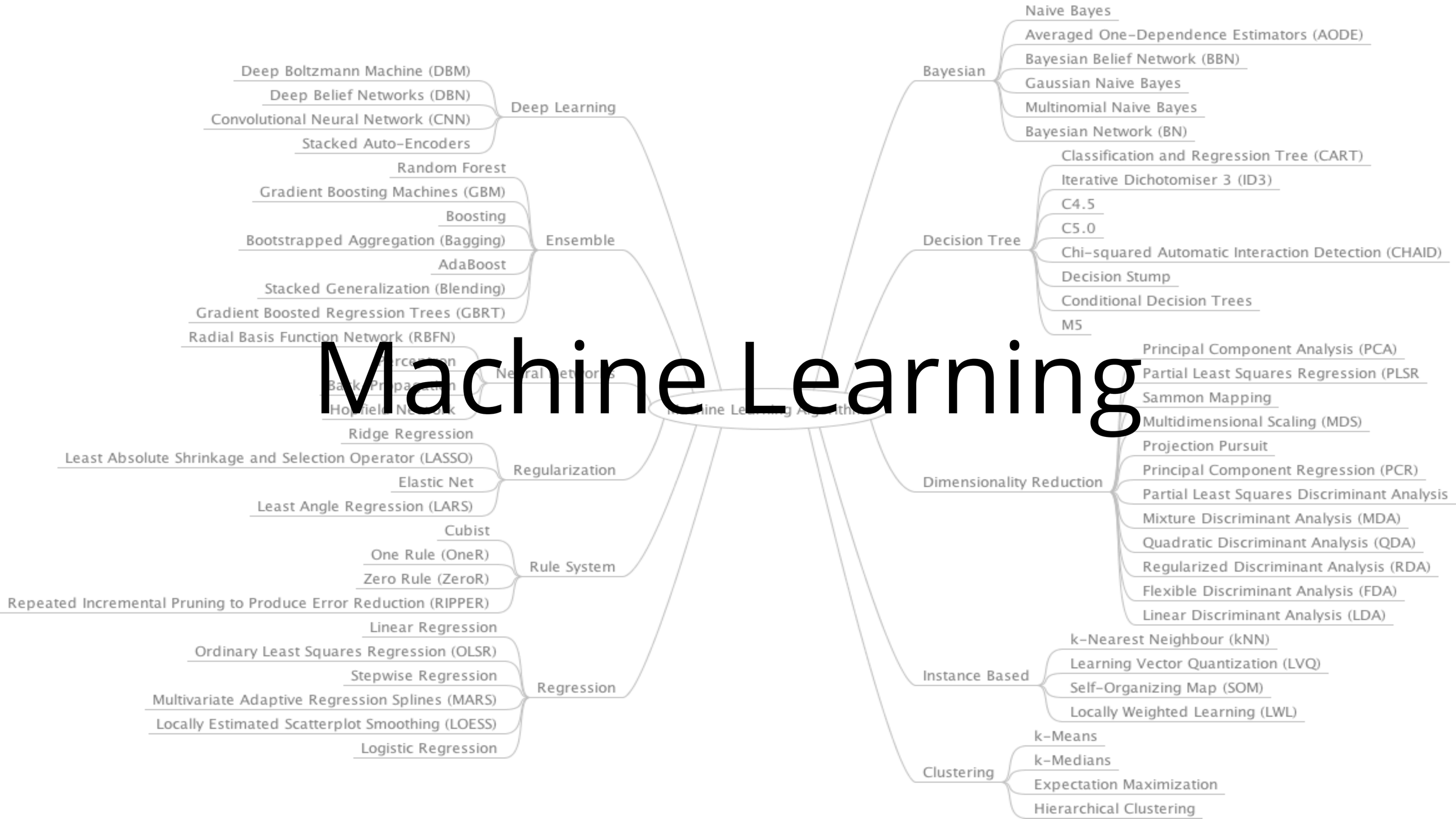
How does it work?



Machine Learning Part



Machine Learning



Machine Learning

"...the programming of a digital computer to behave in a way which, if done by human beings or animals, would be described as involving the process of learning."

Arthur Samuel

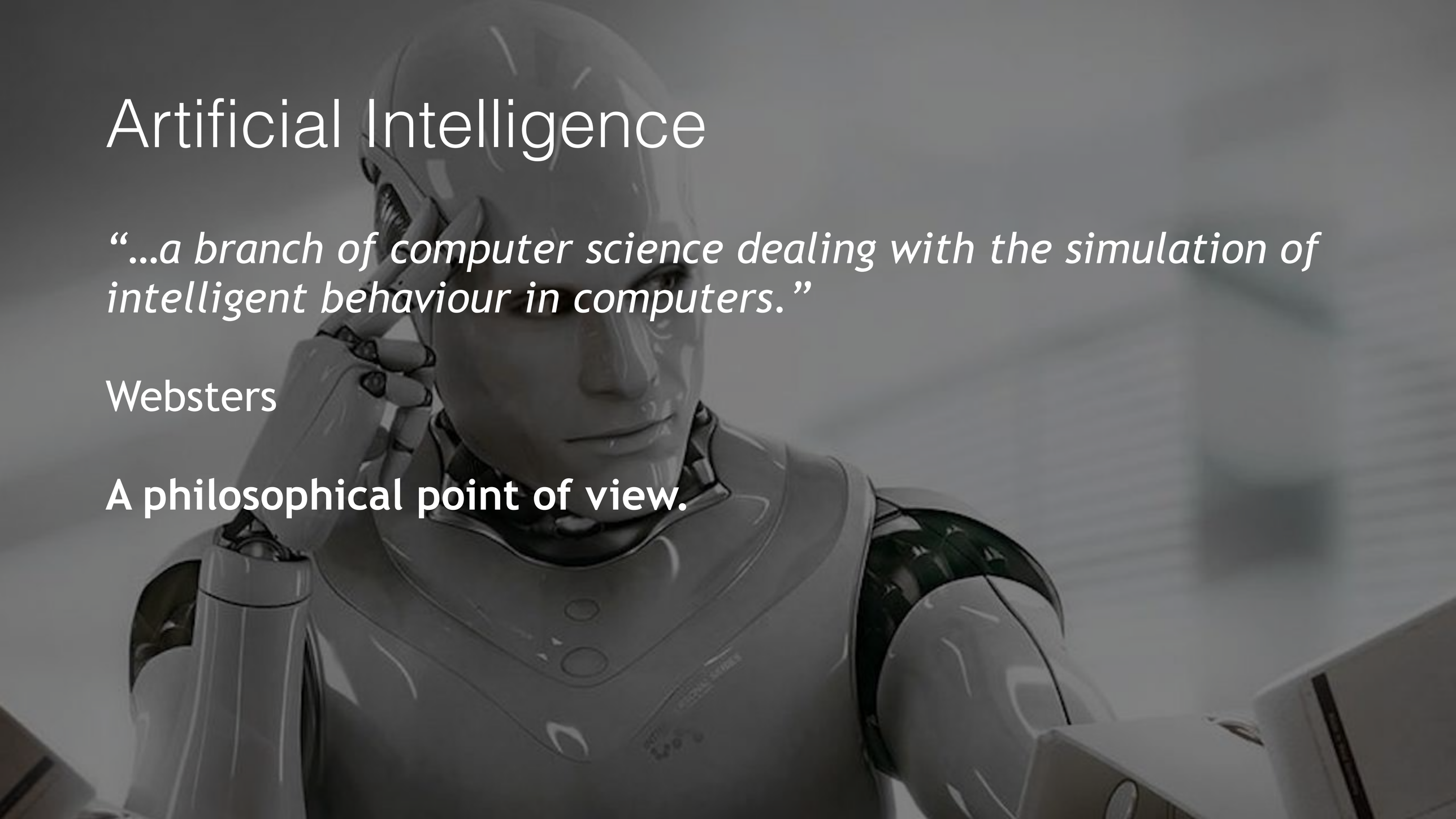
A set of techniques.

Artificial Intelligence

“…a branch of computer science dealing with the simulation of intelligent behaviour in computers.”

Websters

A philosophical point of view.

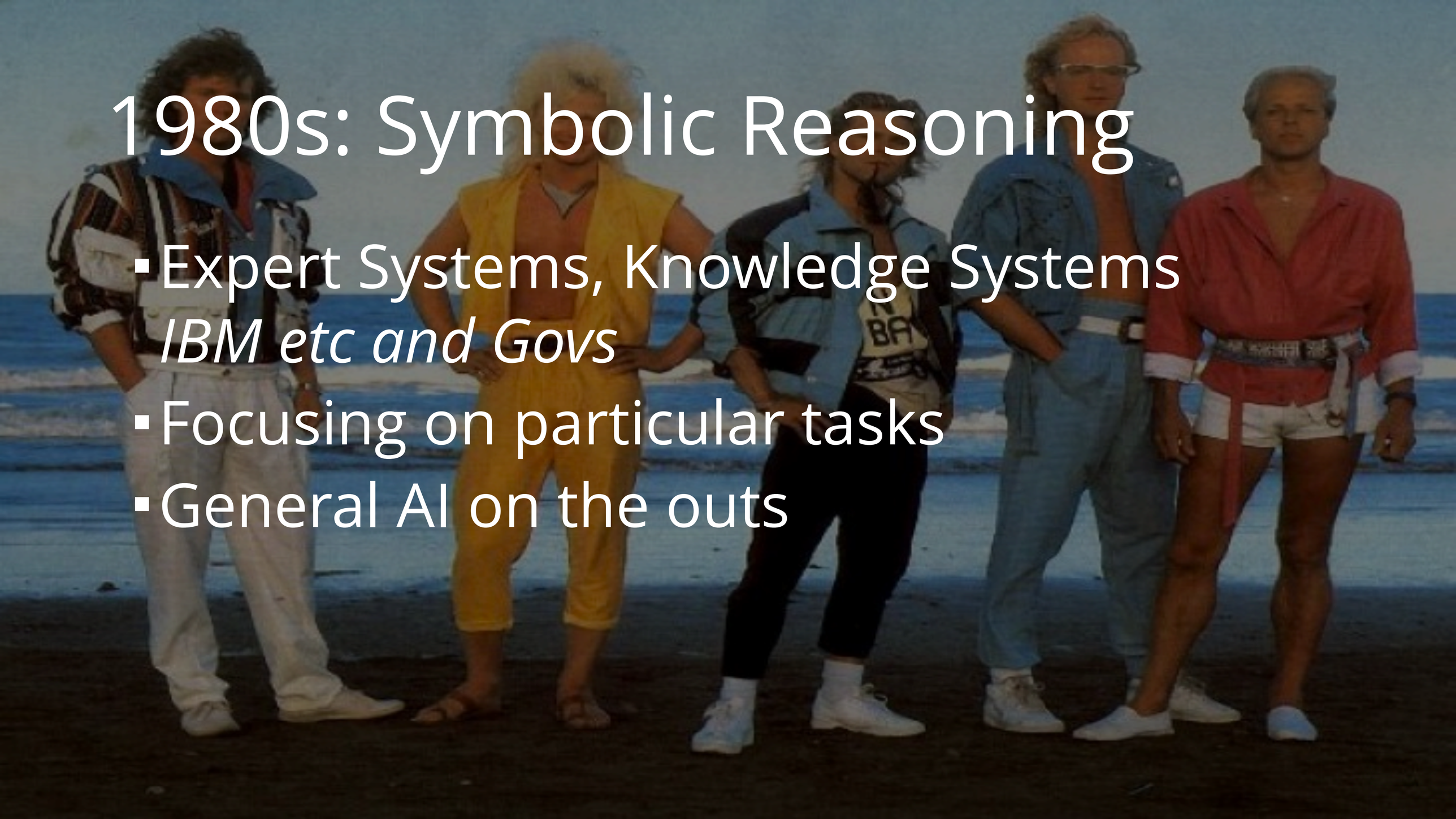



History

1950s: The Beginning

- First neural network & the perceptron
  Frank Rosenblatt
- Computers can play checkers
  Christopher Strachey, Arthur Samuel
- Machine translation
  DARPA, UK GOV

1970s: First AI Winter



1980s: Symbolic Reasoning

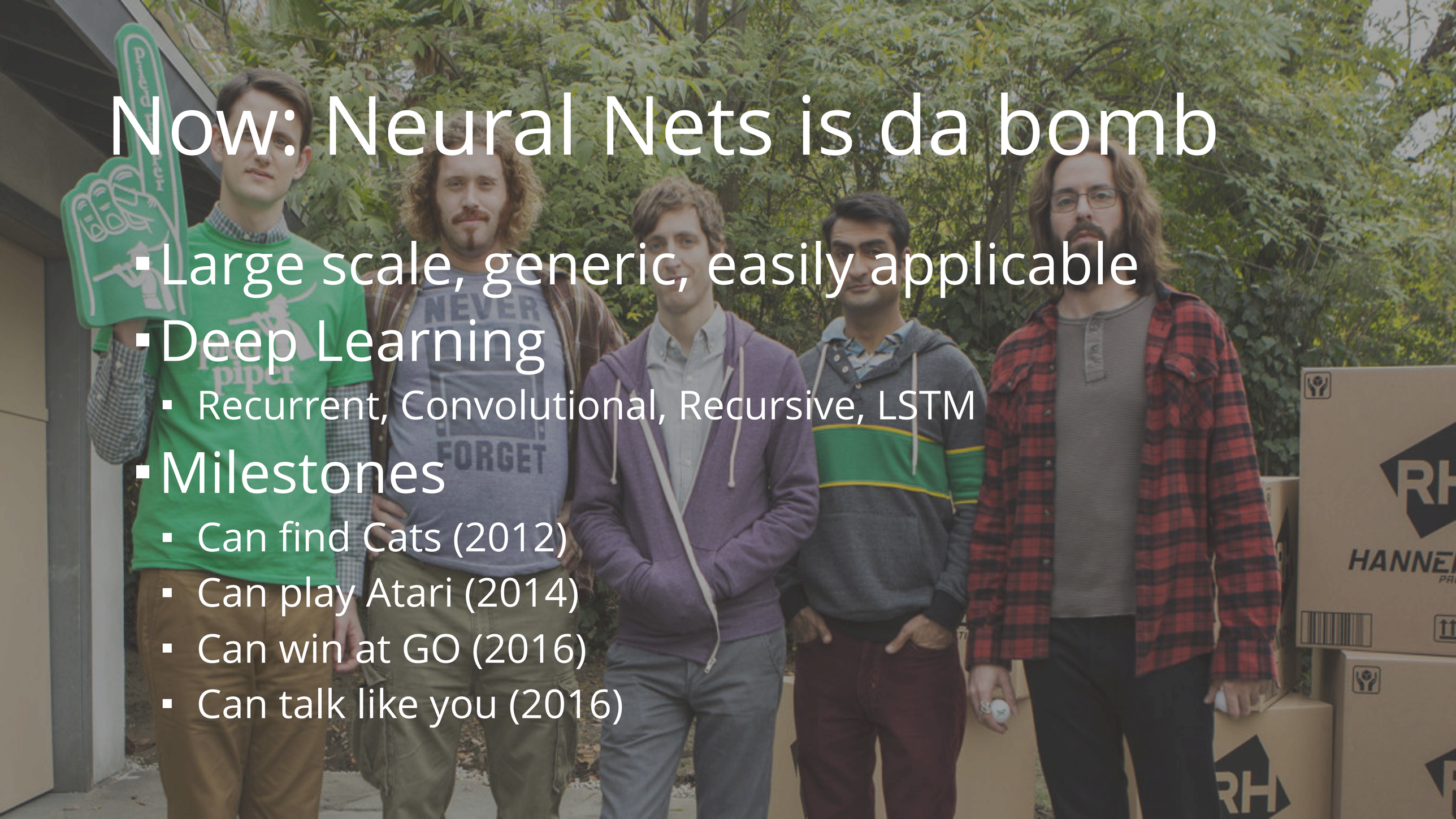
- Expert Systems, Knowledge Systems
  IBM etc and Govs
- Focusing on particular tasks
- General AI on the outs

Second AI Winter

The 1990s: It is summer again

- Markov is big: Statistical machine learning
- Supervised Classification Techniques
  - Support Vector Machines
  - Random Forests
- Unsupervised Techniques
  - Clustering
  - Latent Variable Models

Now: Neural Nets is da bomb



- Large scale, generic, easily applicable
- Deep Learning
  - Recurrent, Convolutional, Recursive, LSTM
- Milestones
  - Can find Cats (2012)
  - Can play Atari (2014)
  - Can win at GO (2016)
  - Can talk like you (2016)

The Money

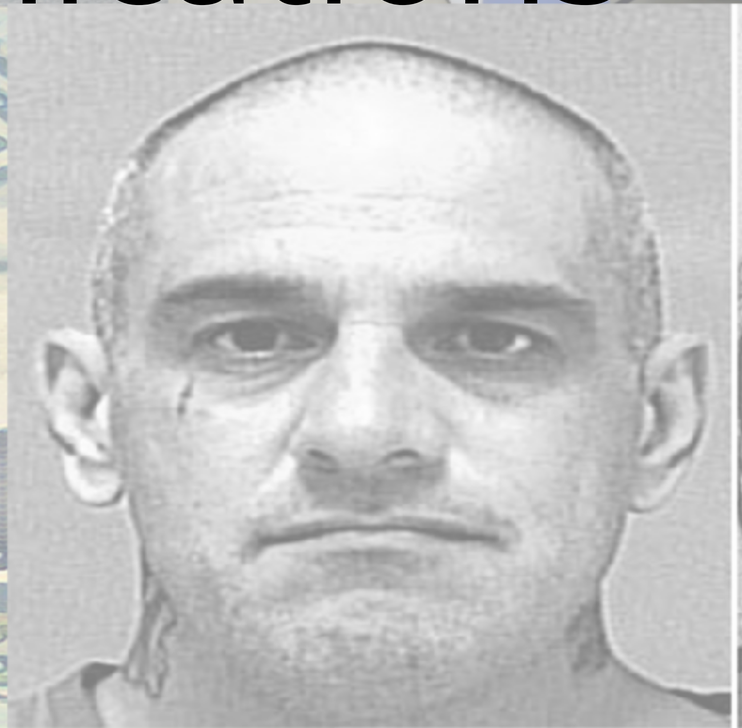
Another bubble?

- Google => DeepMind <London> ($600m)
- Twitter => Magic Pony <London> ($150m)
- Microsoft => Swiftkey <London> ($250m)
- Apple => Turi <Seattle> ($200m)
- Intel => Nervana <San Diego> ($400m)
- GM => Cruise <SF> (>$1bn)
- Uber => Otto <SF> ($700m)
- Salesforce => MetaMind <Palo Alto> ($35m)
- Amazon, Facebook, IBM, Nvidia, Ebay etc. etc.

So much money

- So far 50+ large acquisitions in 2016
- Most had little product
- No revenue, no customers - how do you quantify a deal: X $m per researcher
- Going rate >$10m per engineer/researcher
- 2015: 400 VC investments worth a cool $2.4b

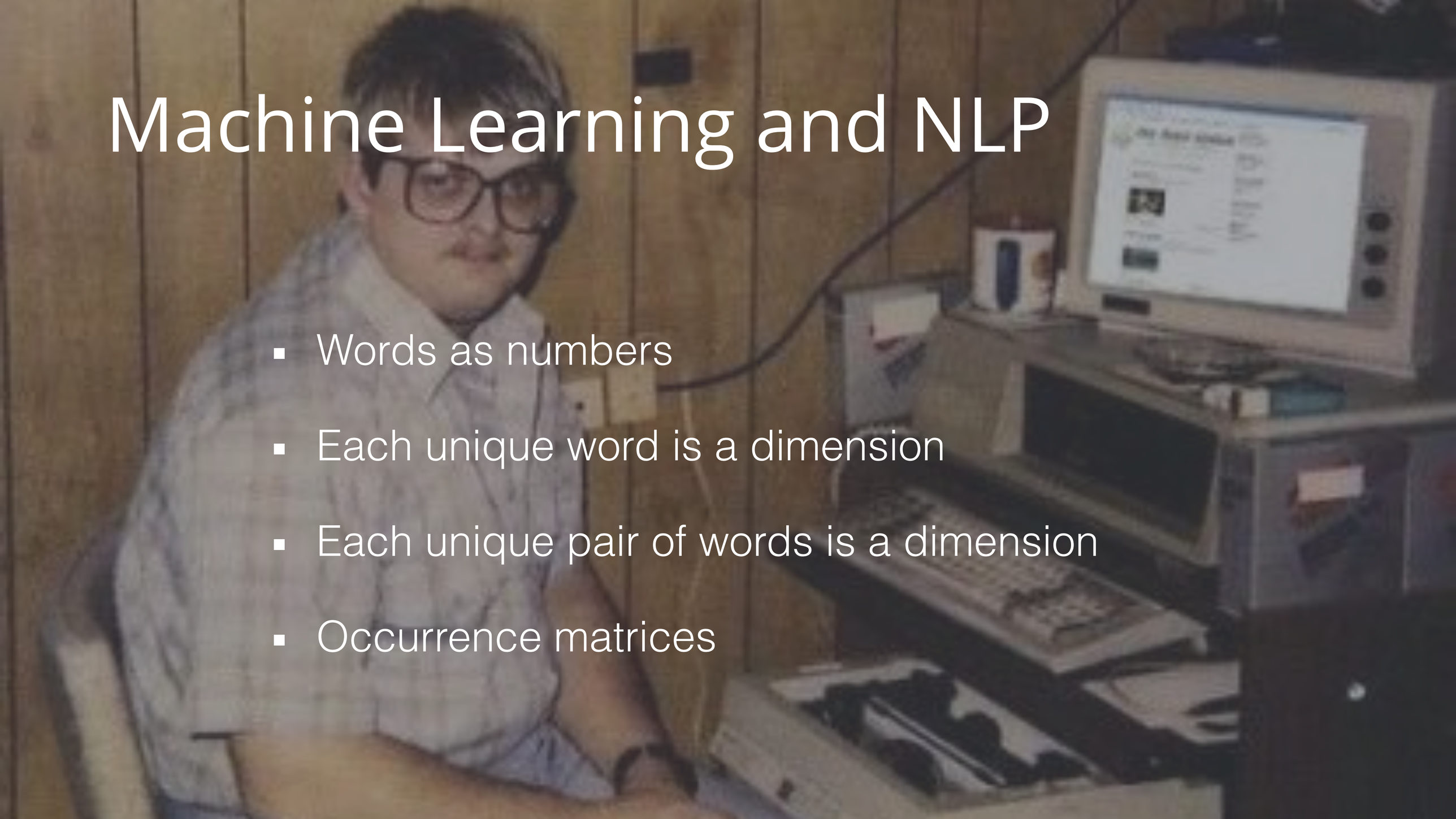
Applications



What are people doing?

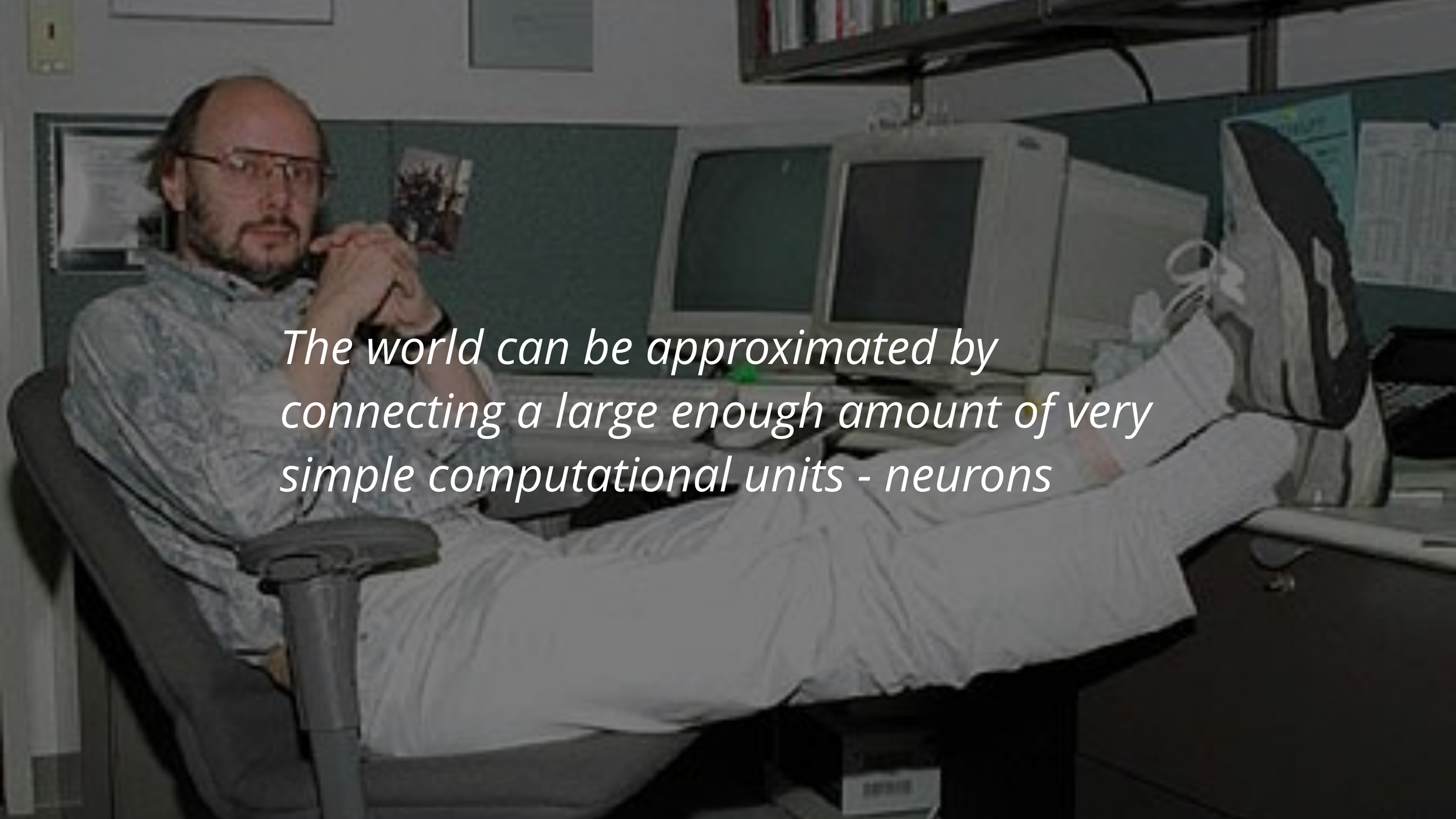
- Autonomous vehicles
  Google, Tesla, Apple, Uber, Airbus, MoD
- Personal assistants
  Alexa, Siri, Cortana, Assistant
- Consumer Finance
- Visual tasks
- Medical Diagnostics
- Recidivism

Deep Learning



Machine Learning and NLP

- Words as numbers
- Each unique word is a dimension
- Each unique pair of words is a dimension
- Occurrence matrices


The world can be approximated by connecting a large enough amount of very simple computational units - neurons

[1950s] Perceptron, *Rosenblatt*

[1980s] Backpropagation, *Hinton, LeCun*

[1980s] Handwriting recognition , CNN, *LeCun*

[1990s] RNN, LSTMs

[2010s] Deep learning

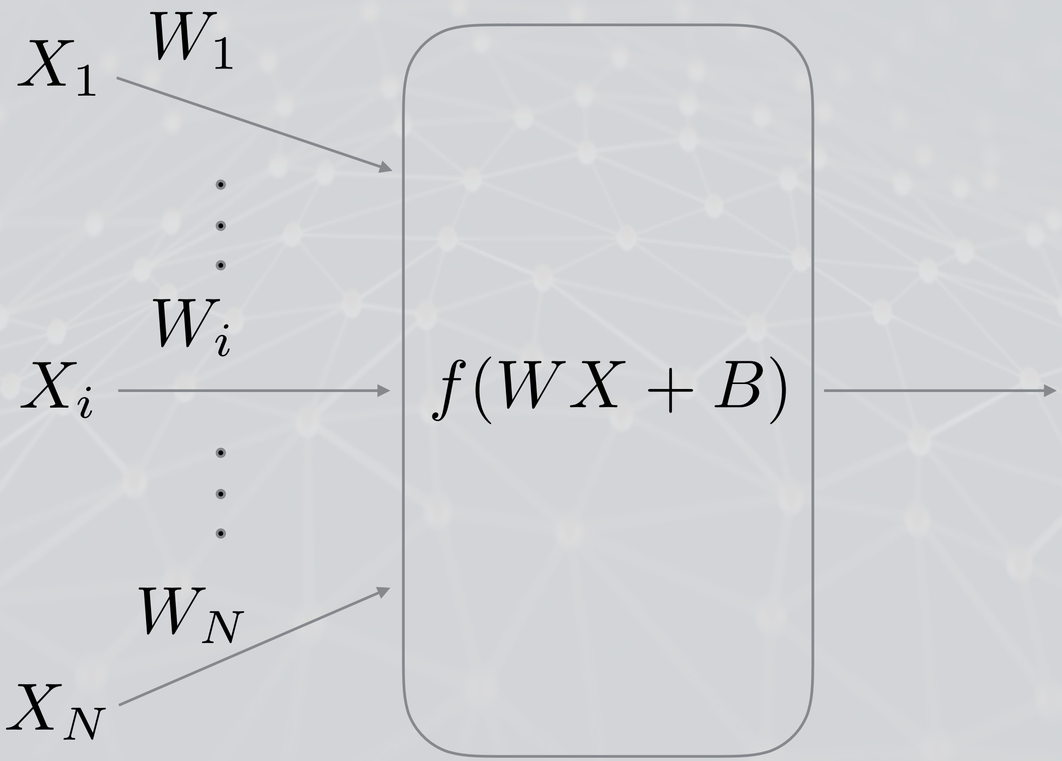
[2012] ImageNet, *Hinton*

[2014] Word2Vec, *Mikolov et al*

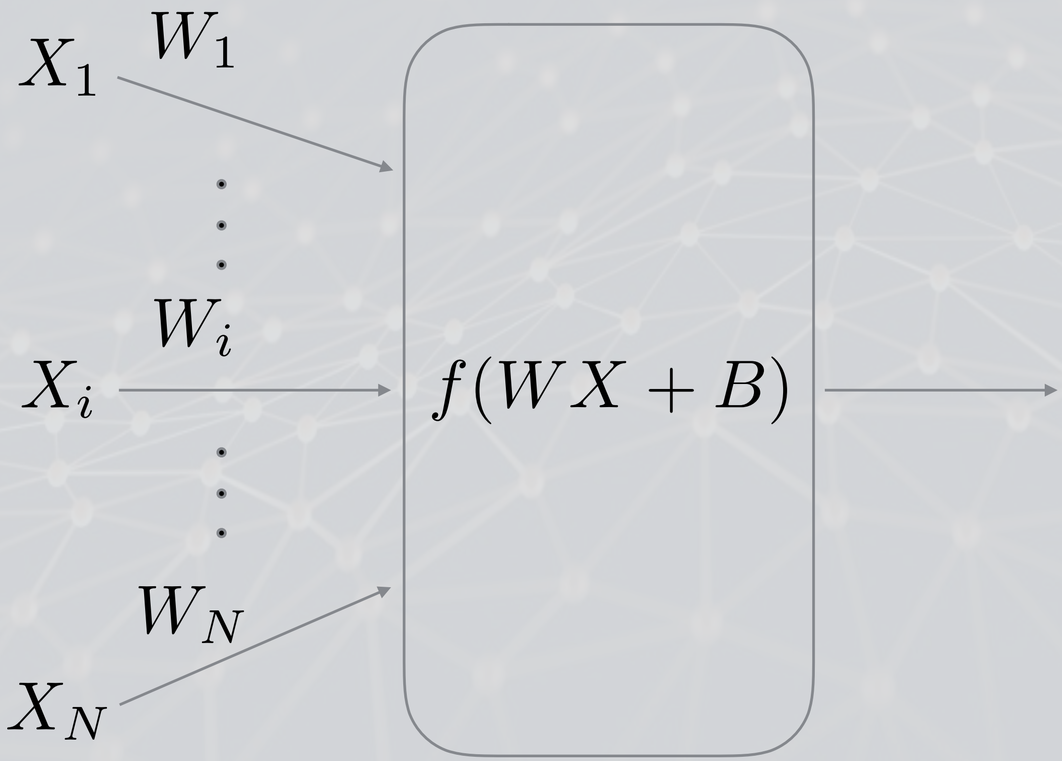
[2016] Wavenet, AlphaGo....

Neuron

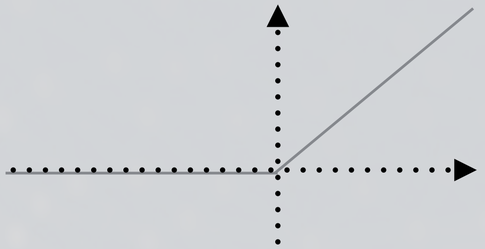
Neuron



Neuron

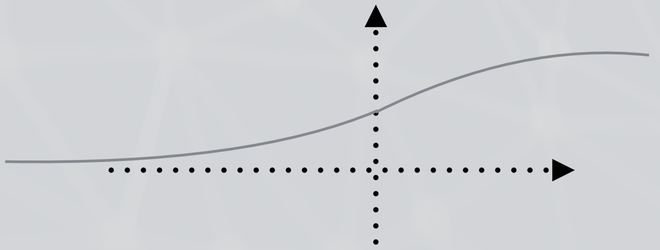


$$f(x) = \max(0, x)$$

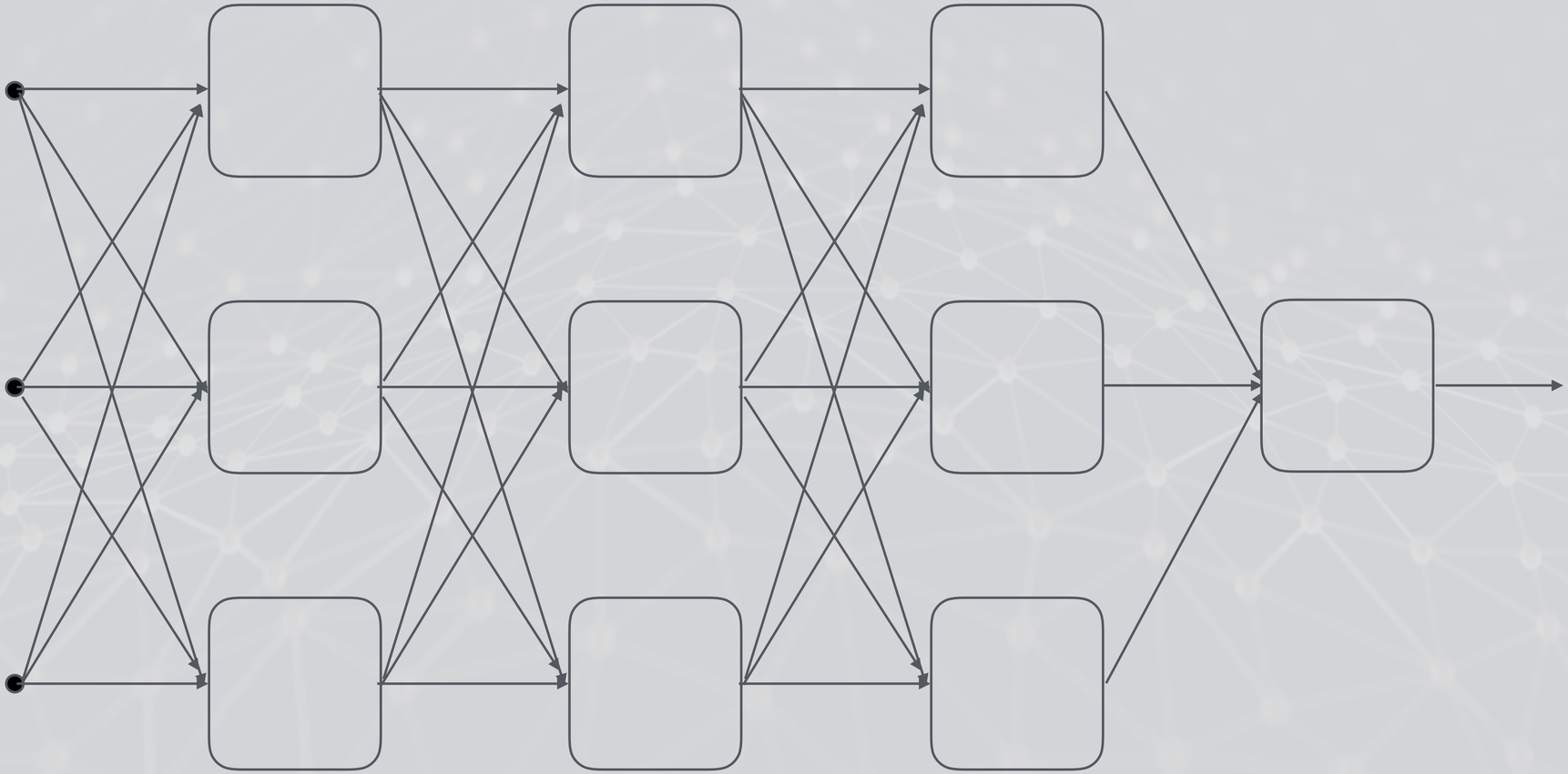


$$f(x) = \tanh(x)$$

$$f(x) = \frac{1}{1 + e^{-x}}$$



Neural Network



Training

Training

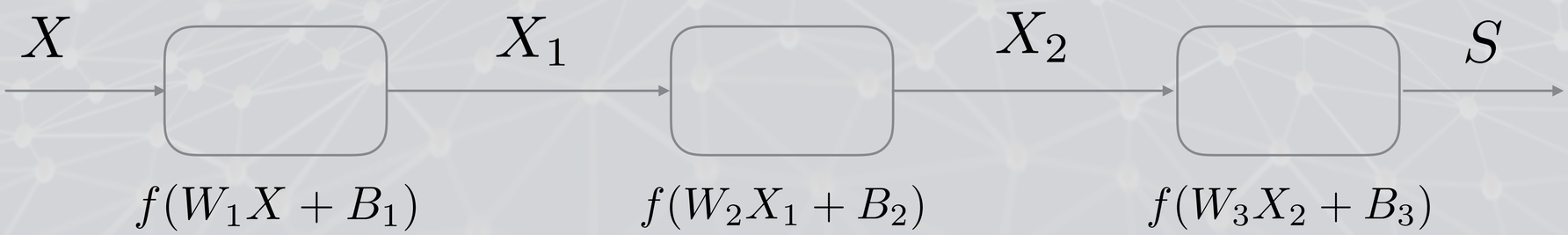
$$(f \circ g)' = (f' \circ g)g'$$

$$f(g(x))' = f'(g(x))g'(x)$$

$$\frac{\partial}{\partial x} f(g(x)) = \frac{\partial f}{\partial x}(g(x)) \frac{\partial g}{\partial x}(x)$$

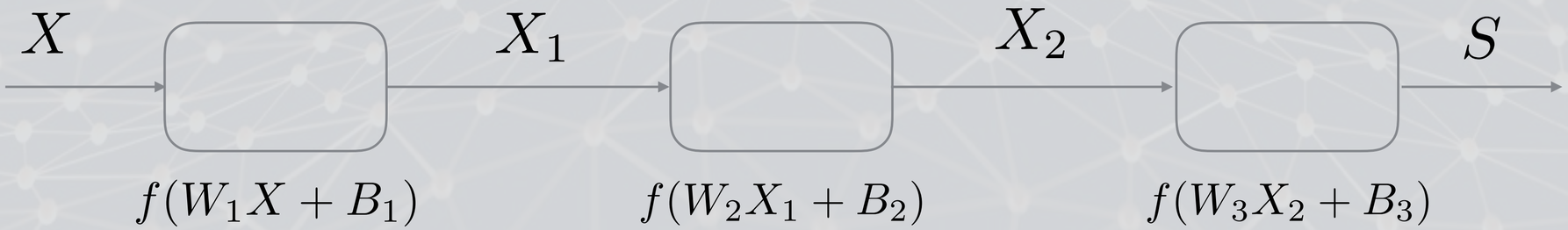
Training

$$(f \circ g)' = (f' \circ g)g'$$



Training

$$(f \circ g)' = (f' \circ g)g'$$

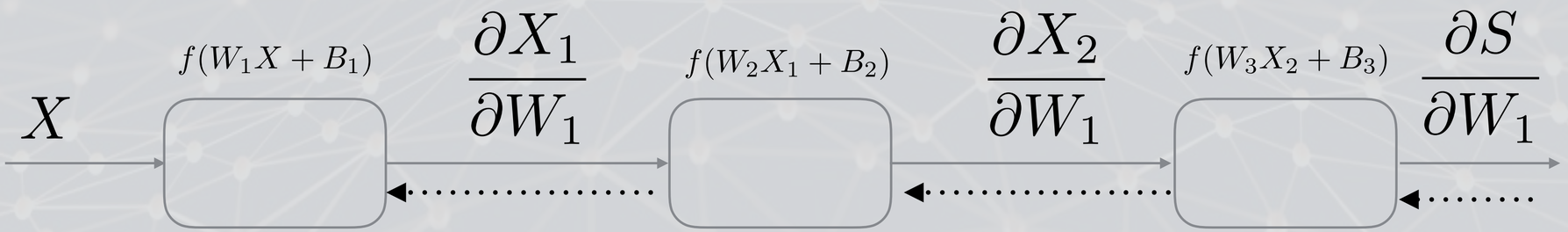


$$J = (S - \hat{S})^2$$

$$\frac{\partial J}{\partial W_1} = 2 * (S - \hat{S}) * \frac{\partial S}{\partial W_1}$$

Training

$$(f \circ g)' = (f' \circ g)g'$$



$f'(W_1X + B_1)X$ $\quad$ $f'(W_2X_1 + B_2)W_2\frac{\partial X_1}{\partial W_1}$ $\quad$ $f'(W_3X_2 + B_3)W_3\frac{\partial X_2}{\partial W_1}$

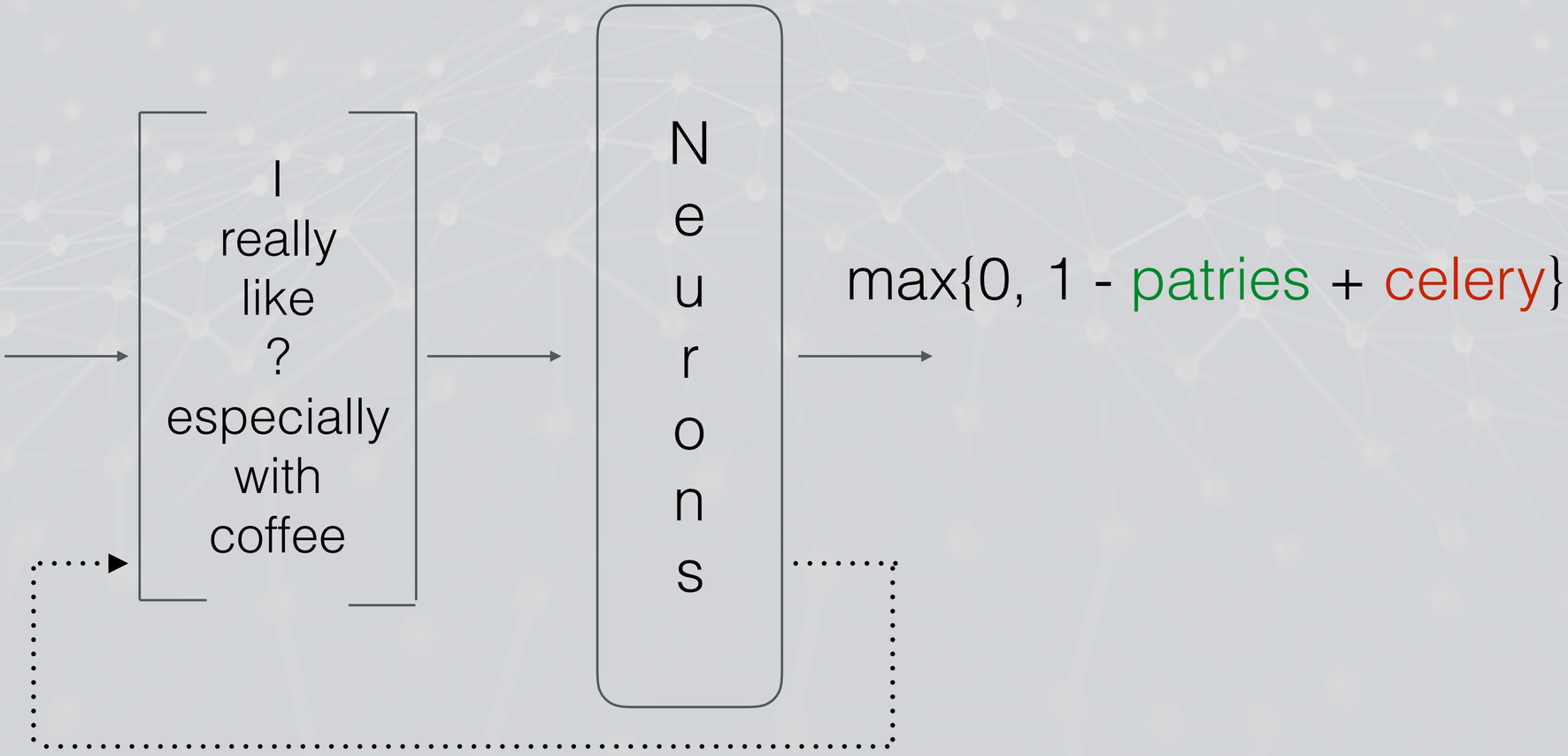
Embeddings

Word2Vec, Glove, … or your own Neural Net

Word2Vec, Glove, … or your own Neural Net

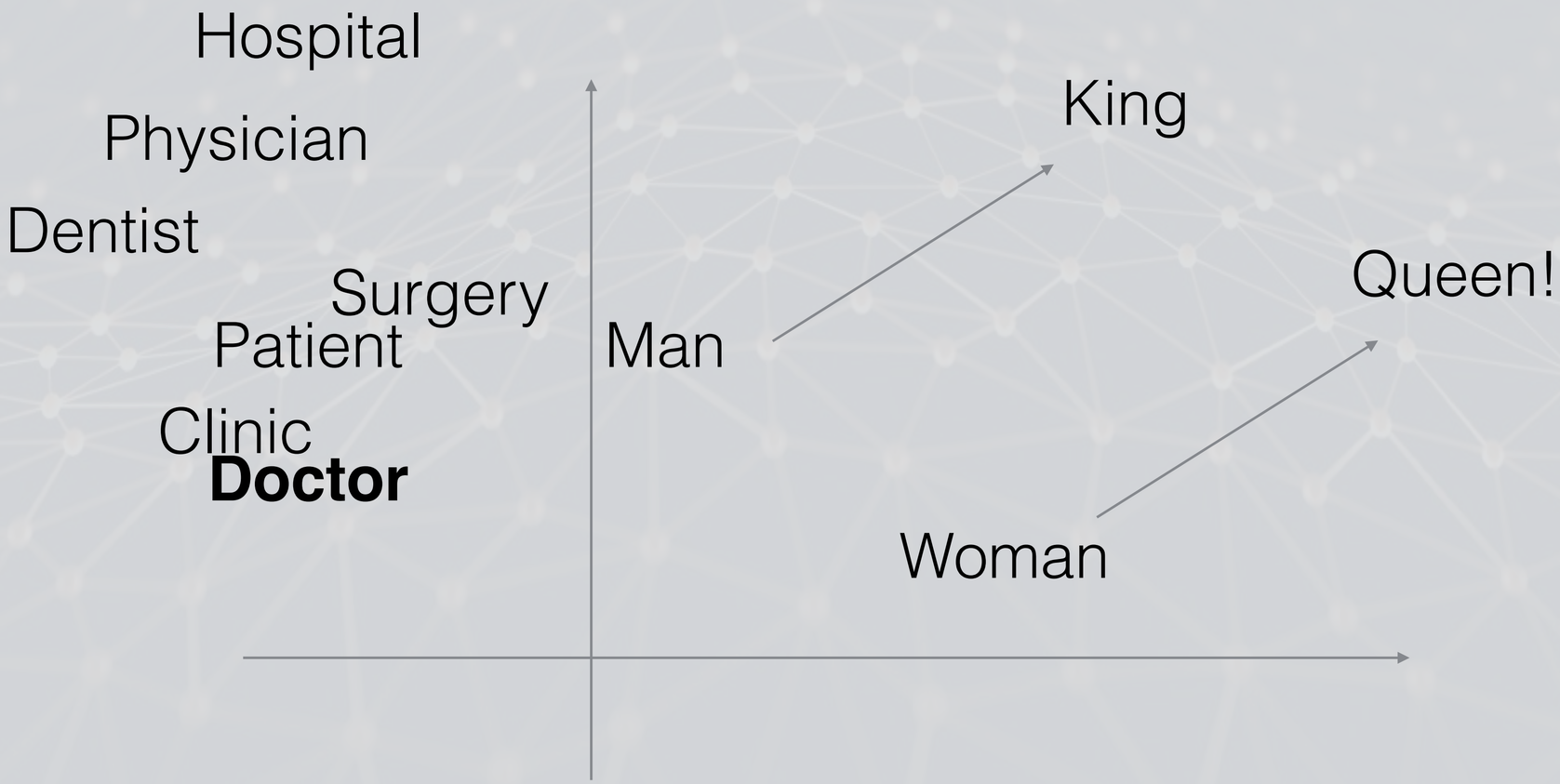
I really like pastries especially with coffee

I really like celery especially with coffee



Man is to *King* what *Woman* is to ?

What relates to *Doctor?*



- ‘*C++*’ cluster ‘*Java*’, ‘*VC++*’, ‘*Visual*’, ‘*MFC*’, ‘*Delphi*’, ‘*VB*’, ‘C’
- ‘Manchester’ cluster ‘Salford’, ‘Leeds’, ‘Liverpool’, ’Lancashire’, ‘Cheshire’
- *Teacher* is to *students* what *doctor* is to ‘*Patients’, ‘Admission’,…*
- *Paris* is to *France* what *London* is to *‘UK’, ‘England’,…*
- *London* is to *banking*, what *Manchester* is to ‘*Retail’, ‘Finance’,…*
- *London* is to *underground*, what *Manchester* is to ‘*Rail’, ‘Railway’, ‘Metrolink’, ….*

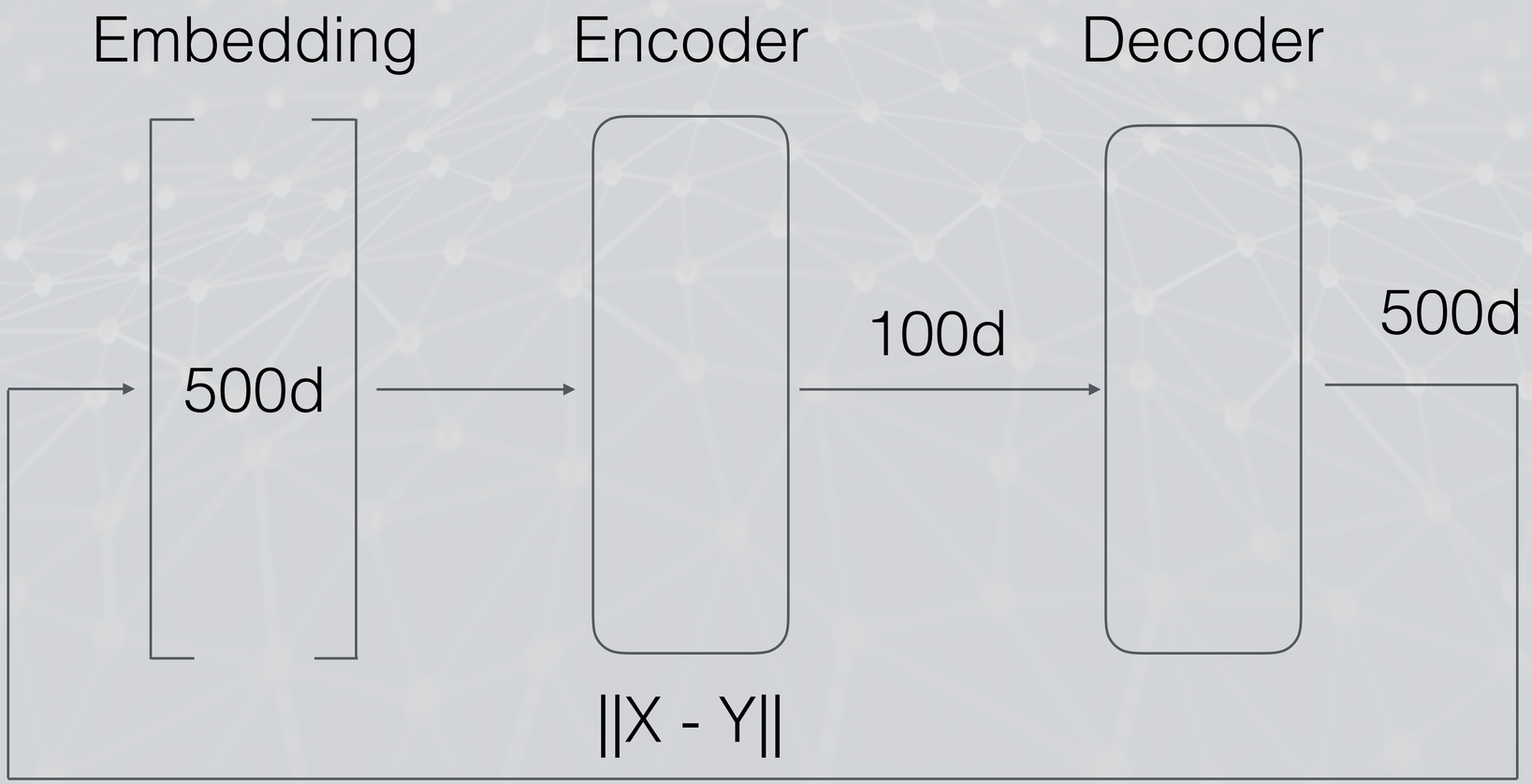
Deep Learning

AutoEncoder

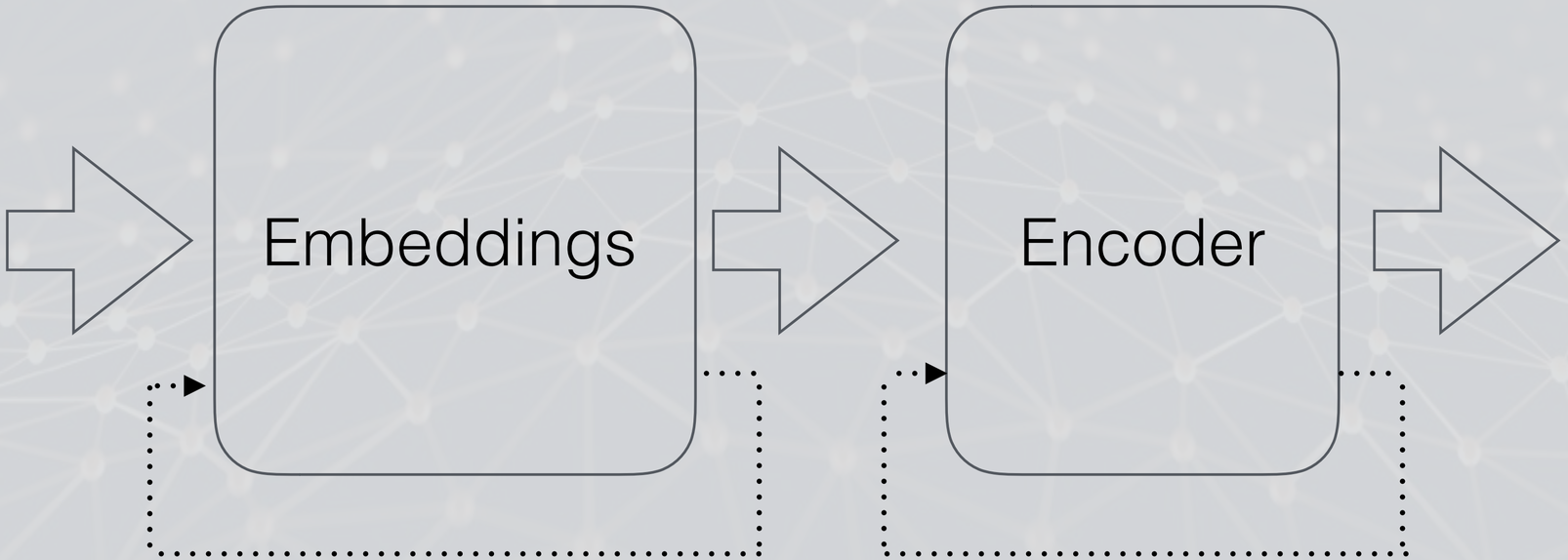
…reducing dimensions…

AutoEncoder

Train projection that preserves *enough* information



It Deep Learns!

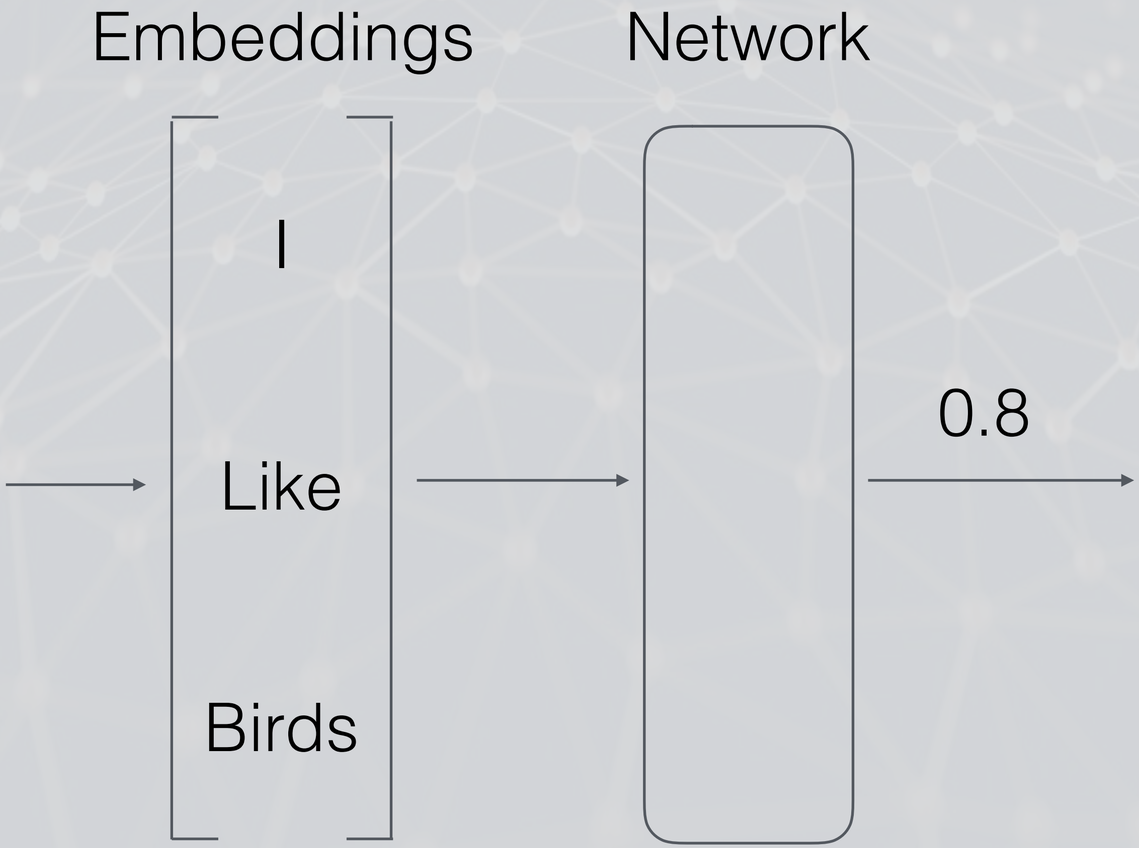


Classifying Sentences

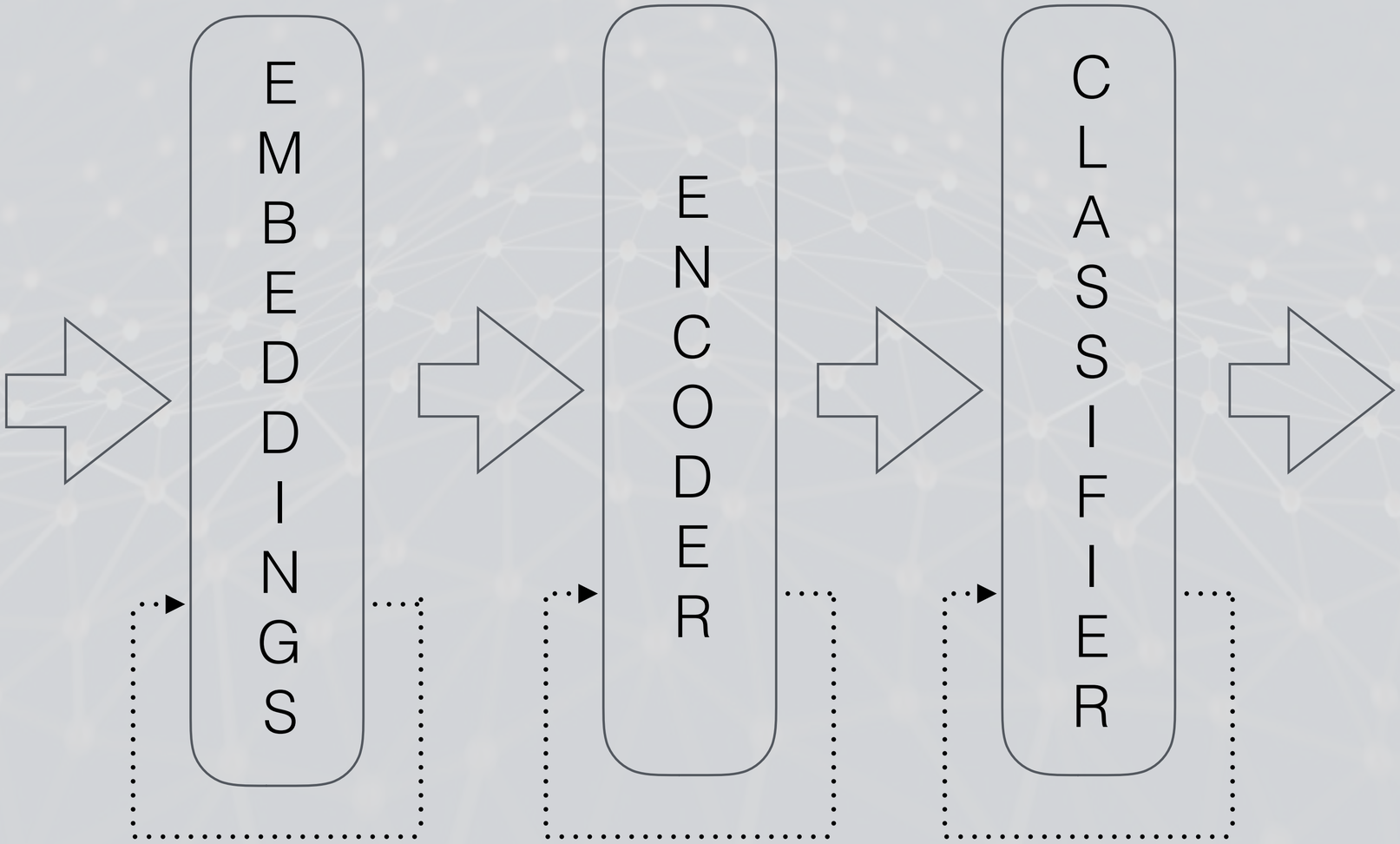
I like birds = positive = 1.0
I hate birds = negative = 0.0

Classifying Sentences

I like birds = positive = 1.0
I hate birds = negative = 0.0



Even Deeper!

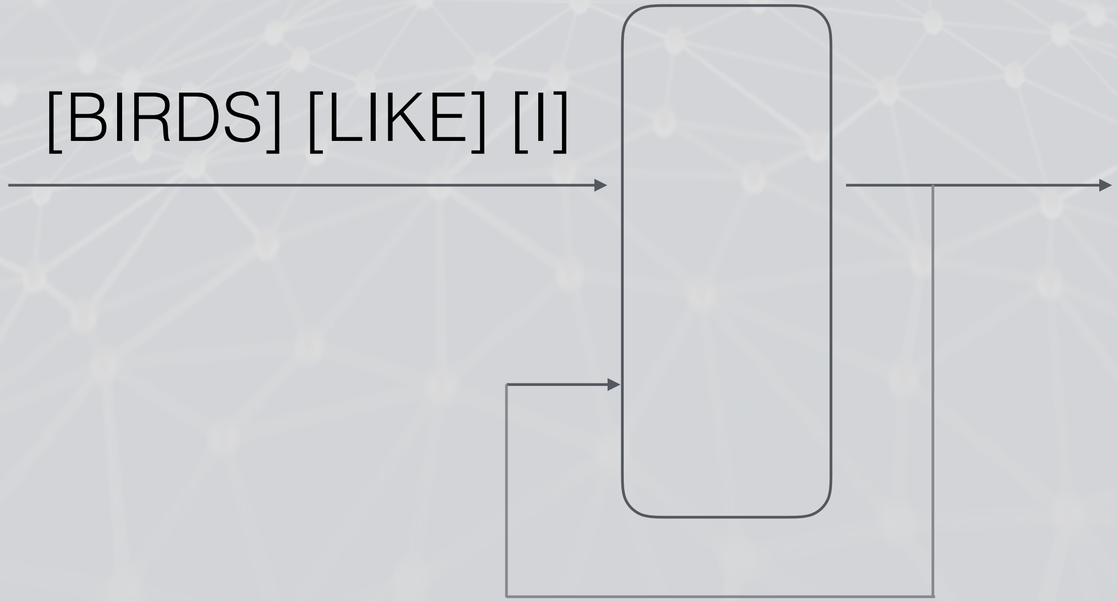


Recurrent Networks

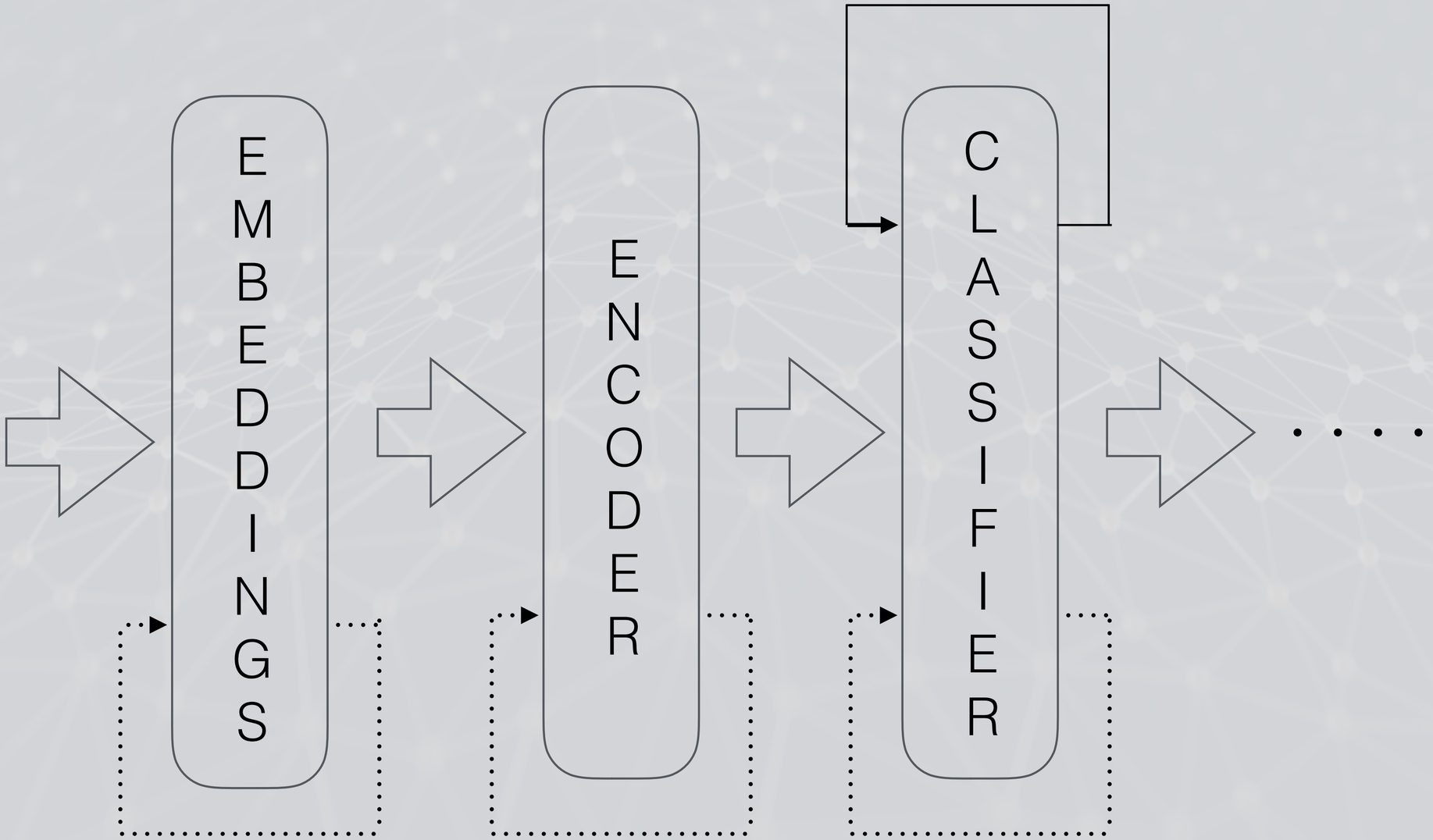
better temporal understanding

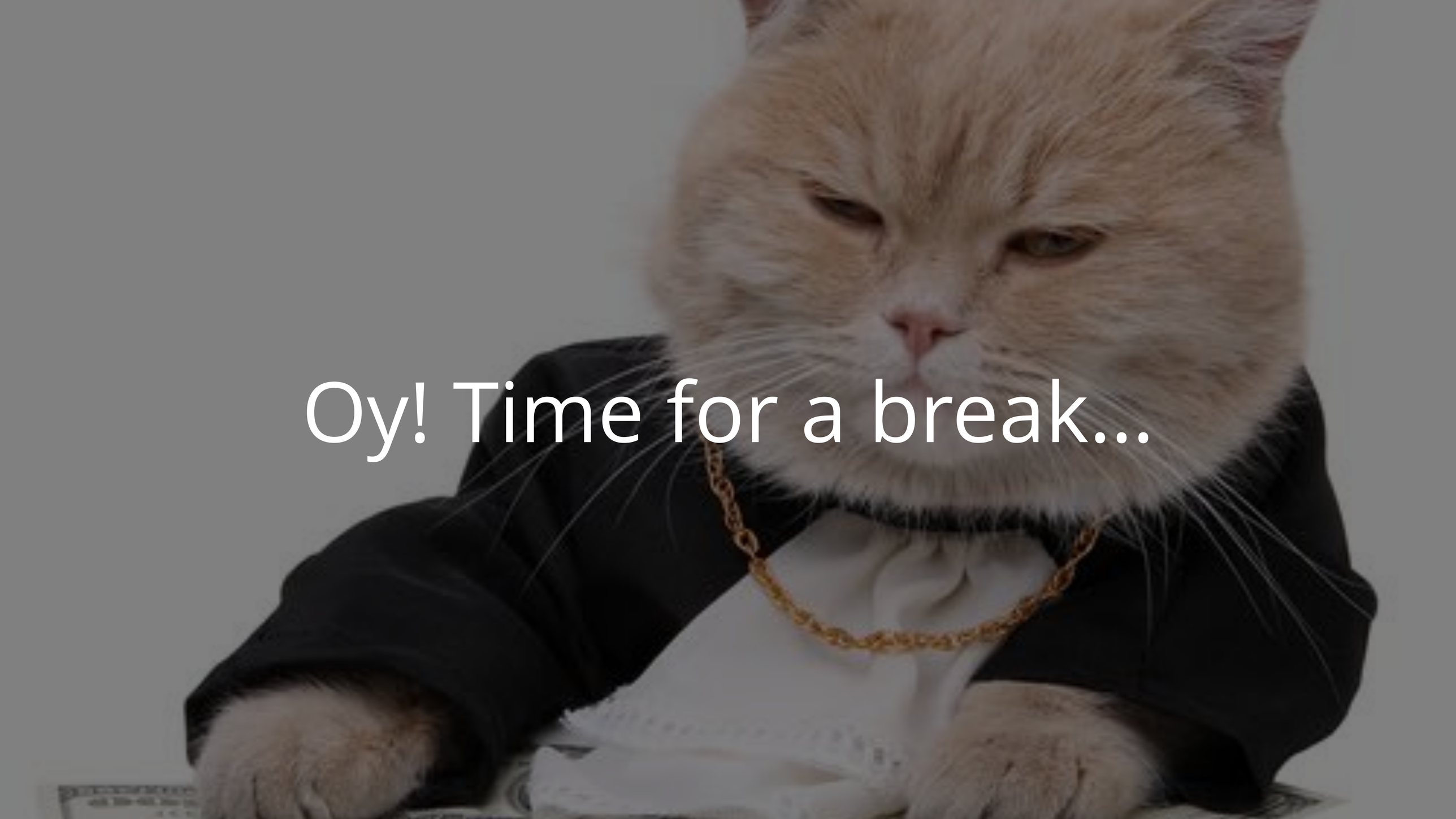
Recurrent Networks

One word at a time feeding the output back in



The Deepest!

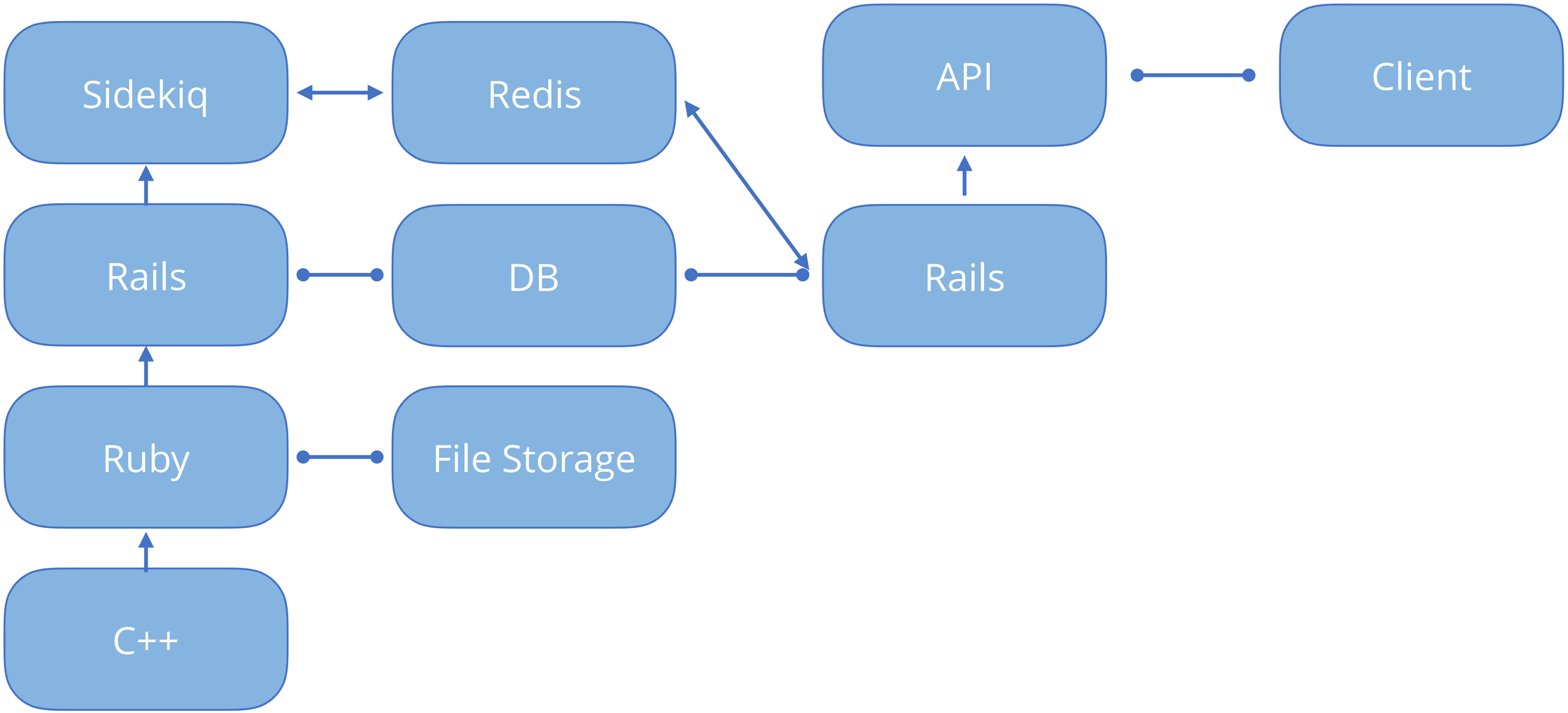



Oy! Time for a break...


PRINTING
ORTON LTD
GRAPHICS
E. mail: orton@netvision.net.il • TEL. 03-5286007 טל. • FAX

Part II: Ruby

Our Tech stack



Why Ruby?

- Plusses:
  - Awesome scripting language for data handling [each, collect, map,..]
  - Rails Frontend = Fast, easy prototyping and testing with real data
  - Great tools(gem, bundler, Jupyter, etc) and platform support
- Minuses:
  - Sketchy library support for numerics(no numpy, scripy, pandas...)
  - Almost non-existent ML support(no scikit-learn, tensorflow)

Ruby and ML

- There are a couple libraries:
  - SciRuby [http://sciruby.com/]
  - NMatrix [https://github.com/sciruby/nmatrix]
  - Wrappers for Octane, R, OpenCV, etc
- Best option: Roll your own gem to wrap existing, or new, C/C++ lib

Ruby and C/C++

- Binding generators
  - SWIG [http://www.swig.org/]
  - FFIG [https://github.com/FFIG/ffig] *please get involved :)*
- Dynamically load shared libs using ffi [https://github.com/ffi/ffi]
- Best option: Use the C-API to write your on gem!

C-API

- Extension build using the gem infrastructure
- The API purely works on pointers to Ruby objects
- All Ruby objects, except for a few special ones, are enriched C-structs
- Special objects are None, true, false,.. (Singleton objects)
- You can define new classes or extend existing

C-API gem layout

```
.\my_gem.gemspec

.\lib\my_gem.rb

.\lib\my_gem\calculator.rb

.\ext\my_gem\extconf.rb

.\ext\my_gem\src\main.cpp

.\ext\my_gem\include\calculator.h

.\ext\my_gem\src\calculator.cpp

.\ext\my_gem\src\ruby\calculator.cpp
```

CODE

Tips and tricks

The Garbage collector: can make it difficult to exchange objects between C and Ruby

```
a = A.new
b = B.new(a)
b.calculate(10)
```

Tip: Wrap all C++ objects in a shared_ptr!

Tip: Often easier to ‘extend’ C-functions with a Ruby layer for munching the data, handling defaults etc

Tip: Use ’_’ for extendable C-functions

Tip: Create macros/functions for getting the objects structs and handle invalid objects by throwing Ruby exception

Tip: Wrap C++-function bodies in try-catch that throws Ruby exception

Jupyter Notebook

The End:

Congratulations!

erik@octavia.ai