# The Science of Search Engines

9 Algorithms Workshop

Defence Institute of Advanced Technology

21st January 2017

John MacCormick

Dickinson College

# Overview of this lecture

- What is an algorithm?
  - What do most people think the word “algorithm” means?
  - Why should most people care about that?
- The science of search engines
  - PageRank
  - Location-based indexing

If time: a glimpse into pattern recognition and machine learning

**Definition 9.5**

An **algorithm** for a function $f : D \to R$ is a Turing machine $M$, which given as input any $d \in D$ on its tape, eventually halts with the correct answer $f(d) \in R$ on its tape. Specifically, we can require that

$$q_0 d \overset{*}{\vdash}_M q_f f(d), q_f \in F,$$

for all $d \in D$.

Linz, *An Introduction to Formal Languages and Automata*

we shall consider their important properties starting in Section 9.2.1. Turing machines that always halt, regardless of whether or not they accept, are a good model of an "algorithm." If an algorithm to solve a given problem exists, then

Hopcroft, Motwani & Ullman, *Introduction to Automata Theory, Languages, and Computation*

*We therefore propose to adopt the Turing machine that halts on all inputs as the precise formal notion corresponding to the intuitive notion of an "algorithm."* Nothing will be considered an algorithm if it cannot be rendered as a Turing machine that is guaranteed to halt on all inputs, and all such machines will be rightfully called algorithms. This principle is known as the **Church-Turing thesis**. It is a thesis, not a theorem, because it is not a mathematical result: It simply asserts that a certain informal concept (algorithm) corresponds to a certain mathematical object (Turing machine). Not being a mathematical

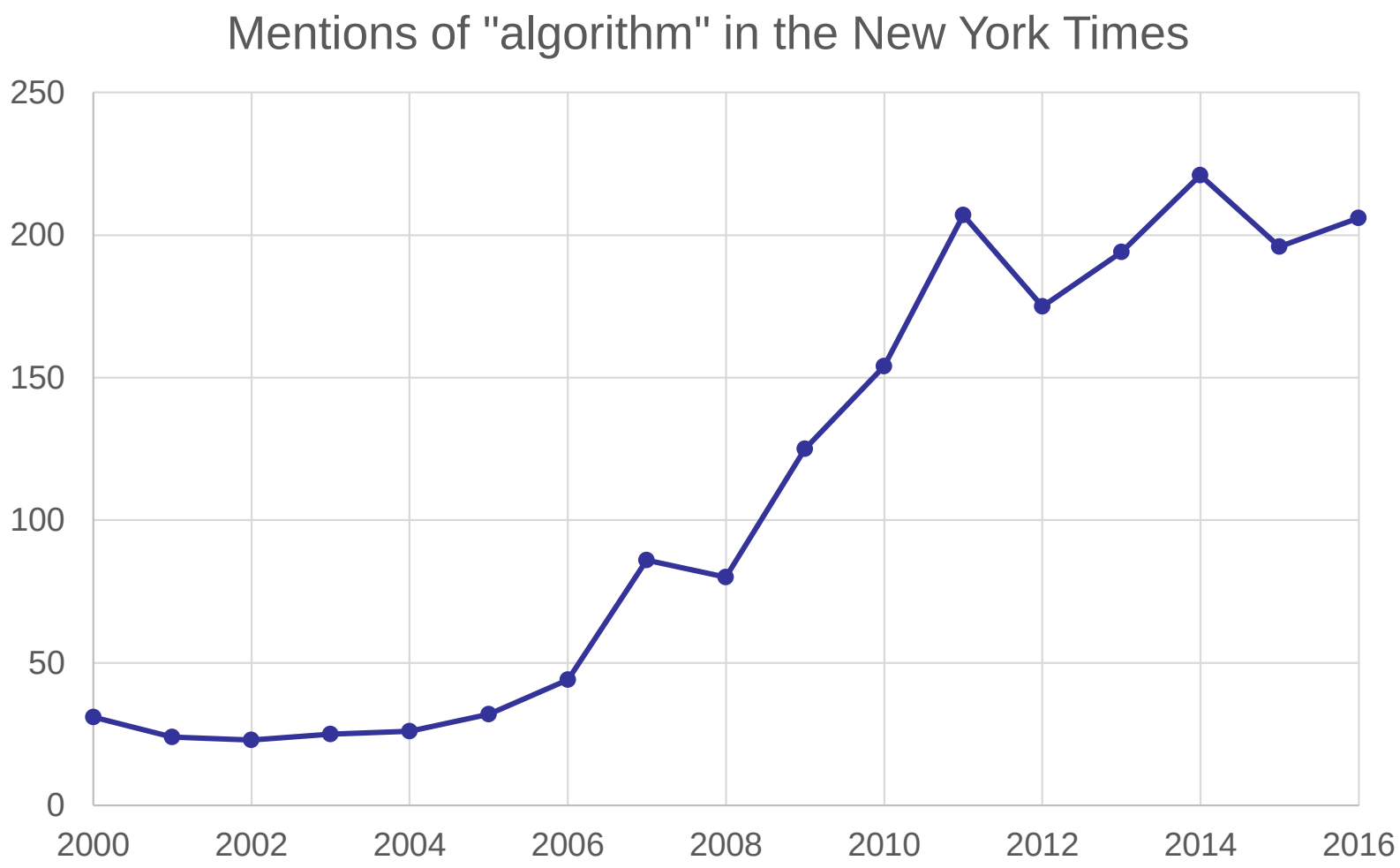
Lewis & Papadimitriou, *Elements of the Theory of Computation*

Informally, an ***algorithm*** is any well-defined computational procedure that takes some value, or set of values, as ***input*** and produces some value, or set of values, as ***output***. An algorithm is thus a sequence of computational steps that transform the input into the output.

We can also view an algorithm as a tool for solving a well-specified ***computational problem***. The statement of the problem specifies in general terms the desired input/output relationship. The algorithm describes a specific computational procedure for achieving that input/output relationship.

Cormen-Leiserson-Rivest-Stein, *Introduction to Algorithms*

# Do we live in an age of algorithms?



1. Introduction: What are the great ideas that power your PC?
2. Search engine indexing: finding needles in the world's biggest haystack
3. PageRank: the technology that launched Google
4. Public key cryptography: sending secrets on a postcard
5. Error correcting codes: mistakes that fix themselves
6. Pattern recognition: learning from experience
7. Data compression: something for nothing
8. Databases: the quest for consistency
9. Digital signatures: who *really* wrote this software?
10. What is computable?
11. Conclusion: more genius at your fingertips?

# Overview of this lecture

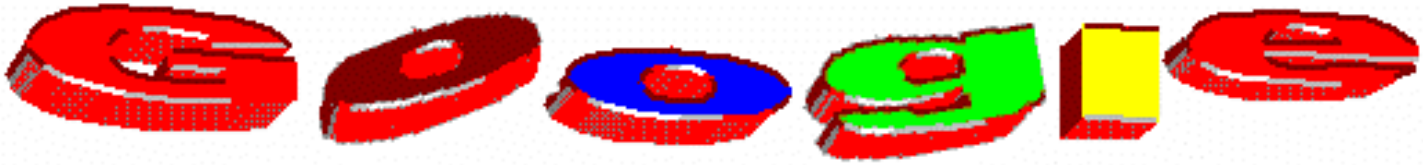
- What is an algorithm?
  - What do most people think the word “algorithm” means?
  - Why should most people care about that?

- The science of search engines
  - PageRank
  - Location-based indexing

# Search engines have profoundly changed the way ordinary people use computers

- huge amount of information available
  - “most” of the world’s “useful” information is out there on the Web??
- search engines are incredibly easy to use
  - no fancy query language needed

# Google in 1998

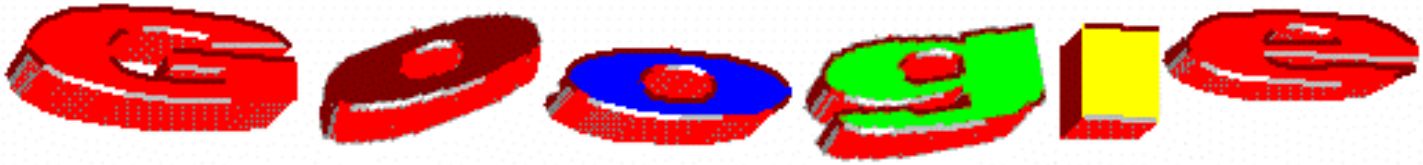


Search The Web (type only necessary words):

10 results | clustering on | Search

Current Repository Size: ~25 million pages (searchable index slightly smaller)

# Google in 1998



Search The Web (type only necessary words):

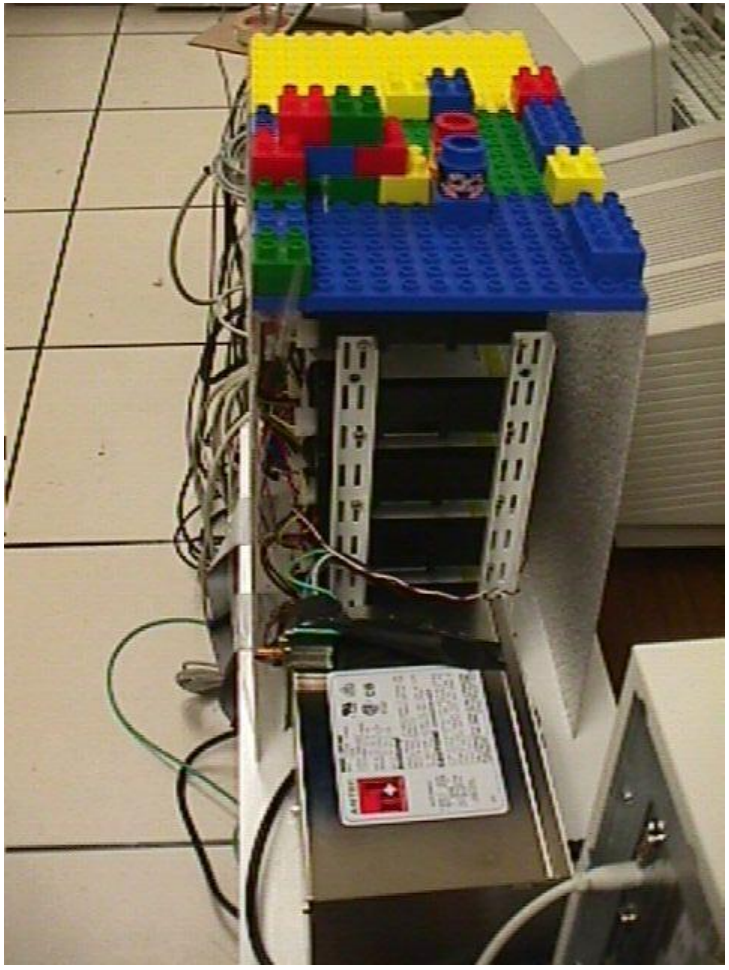
10 results | clustering on | Search

Current Repository Size: ~25 million pages (searchable index slightly smaller)

now billions!

# Google in 1998

# Google’s storage system in 1998



(ten 9-gigabyte hard drives)

# how do search engines do it?

1. web crawling
2. indexing
3. searching
    a) retrieval
    b) ranking

# how do search engines do it?

1. web crawling
2. indexing
3. searching
    a) retrieval
    b) ranking

# what is the ranking problem?

- “scrambled eggs” gets nearly one million hits
- user only has patience to look at about 10 results
- need to *rank* the one million hits, and present the top 10 on the first page of results

# which page is probably more “useful” or “authoritative”?

links **to** a page
confer authority
on that page

# which page is probably more “useful” or “authoritative”?

links from a more
authoritative
page confer
greater authority

# PageRank computes the authority of a page rigorously, using matrix algebra

- create “hyperlink matrix”

- transform slightly (details omitted)
- compute principal eigenvector
  - nth coordinate of the eigenvector is the PageRank of the nth page

# Overview of this lecture

- What is an algorithm?
  - What do most people think the word “algorithm” means?
  - Why should most people care about that?
- The science of search engines
  - PageRank
  - Location-based indexing

# standard indexing uses document IDs

1 the cat sat on the mat

2 the dog stood on the mat

3 the cat stood while a dog sat

| | | | |
|---|---|---|---|
| a | 3 | | |
| cat | 1 | 3 | |
| dog | 2 | 3 | |
| mat | 1 | 2 | |
| the | 1 | 2 | 3 |
| sat | 1 | 3 | |
| stood | 2 | 3 | |
| on | 1 | 2 | |
| while | 3 | | |

example queries:

while

dog

cat dog

“cat sat”

standard indexing is not powerful enough

# brilliant idea number 1: index word *locations* within documents

1 the cat sat on the mat

2 the dog stood on the mat

3 the cat stood while a dog sat

| | |
|---|---|
| a | 3.5 |
| cat | 1.2 3.2 |
| dog | 2.2 3.6 |
| mat | 1.6 2.6 |
| the | 1.1 1.5 2.1 2.5 3.1 |
| sat | 1.3 3.7 |
| stood | 2.3 3.3 |
| on | 1.4 2.4 |
| while | 3.4 |

# phrase queries are easy using location-based indexing

| | |
|---|---|
| ... | ... |
| cat | ... 5.9 6.1 8.3 ... |
| sat | ... 4.2 6.3 6.9 9.5 |
| ... | ... |

query: “cat sat”

result: no documents match

# phrase queries are easy using location-based indexing

| | |
|---|---|
| ... | ... |
| cat | ... 5.9 6.8 8.3 ... |
| sat | ... 4.2 6.3 6.9 9.5 |
| ... | ... |

query: “cat sat”

result: document 6 matches

# NEAR queries are also easy using location-based indexing

| | | | | | |
|---|---|---|---|---|---|
| ... | ... | | | | |
| cat | ... | 5.9 | 6.1 | 8.3 | ... |
| sat | ... | 4.2 | 6.5 | 6.9 | 9.5 |
| ... | ... | | | | |

query: cat NEAR sat

result: no matches

# NEAR queries are also easy using location-based indexing

| | | | | | |
|---|---|---|---|---|---|
| ... | ... | | | | |
| cat | ... | 5.9 | 6.9 | 8.3 | ... |
| sat | ... | 4.2 | 6.5 | 6.7 | 9.5 |
| ... | ... | | | | |

query: cat NEAR sat

result: document 6 matches

# knowing NEARness is also important for ranking

- example query: departed movie
- document 1:
  - "...The Departed is an great movie starring Jack Nicholson..."
- document 2:
  - "blog blog blog ... went to see a movie ... blog blog blog ... more blog ... had to fly to New York ... flight was late ... it finally departed at 10 PM"

document 1 should be ranked higher; location-based indexing lets you do that

# brilliant idea number 2: use *metawords* to permit queries that reflect the *structure* of documents

1 **My Cat**

the cat sat on the mat

2 **My Dog**

the dog stood on the mat

3 **My Pets**

the cat stood while a dog sat

# brilliant idea number 2: use *metawords* to permit queries that reflect the *structure* of documents

1 <title>My Cat</title> <body>the cat sat on the mat</body>

2 <title>My Dog</title> <body>the dog stood on the mat</body>

3 <title>My Pets</title> <body>the cat stood while a dog sat</body>

# brilliant idea number 2: use *metawords* to permit queries that reflect the *structure* of documents

1 <title>My Cat</title> <body>the cat sat on the mat</body>

2 <title>My Dog</title> <body>the dog stood on the mat</body>

3 <title>My Pets</title> <body>the cat stood while a dog sat</body>

| | | | |
|---|---|---|---|
| cat | 1.3 | 1.7 | 3.7 |
| sat | 1.8 | 3.12 | |
| ... | ... | | |
| <title> | 1.1 | 2.1 | 3.1 |
| </title> | 1.4 | 2.4 | 3.4 |
| <body> | 1.5 | 2.5 | 3.5 |
| </body> | 1.12 | 2.12 | 3.13 |

# queries on document structure are easy

| | |
|---|---|
| ... | ... |
| cat | ... 5.9 6.8 7.3 ... |
| ... | ... |
| <title> | ... 5.2 6.3 7.1 |
| </title> | ... 5.4 6.5 7.4 |

query: cat IN <title>

result: document 7 matches

# queries on document structure are easy

| | |
|---|---|
| ... | ... |
| cat | ... 5.7 6.4 8.3 ... |
| ... | ... |
| <title> | ... 5.2 6.3 7.1 |
| </title> | ... 5.8 6.5 7.4 |

query: cat IN <title>

result: documents 5 and 6

# IN queries also help with ranking

- example query: cat
- document 1: “<title>The Cat Page</title>...”
- document 2: “<title>John’s blog</title><body>blog blog blog...more blog blog...I dressed up as a black cat for Halloween...blog blog blog</body>

document 1 should be ranked higher; location-based indexing lets you do that

# location-based indexing can be implemented in an elegant object-oriented framework

- use *index stream reader* (ISR) objects
- ISR methods are:
  - get_loc()
  - get_next_loc()
  - get_loc_limit()
  - get_previous_loc()
- subclasses include:
  - ISR_and
  - ISR_or
  - ISR_not

# how do search engines do it?

1. web crawling
2. indexing
3. searching
    a) retrieval
    b) ranking

# some more science behind search engines:

- GFS (Google file system)
- MapReduce, Dryad (parallel computation)
- shingling (efficient similarity detection)
- ad pricing (real-time auctions)
- Mercator (web crawling)

# Overview of this lecture

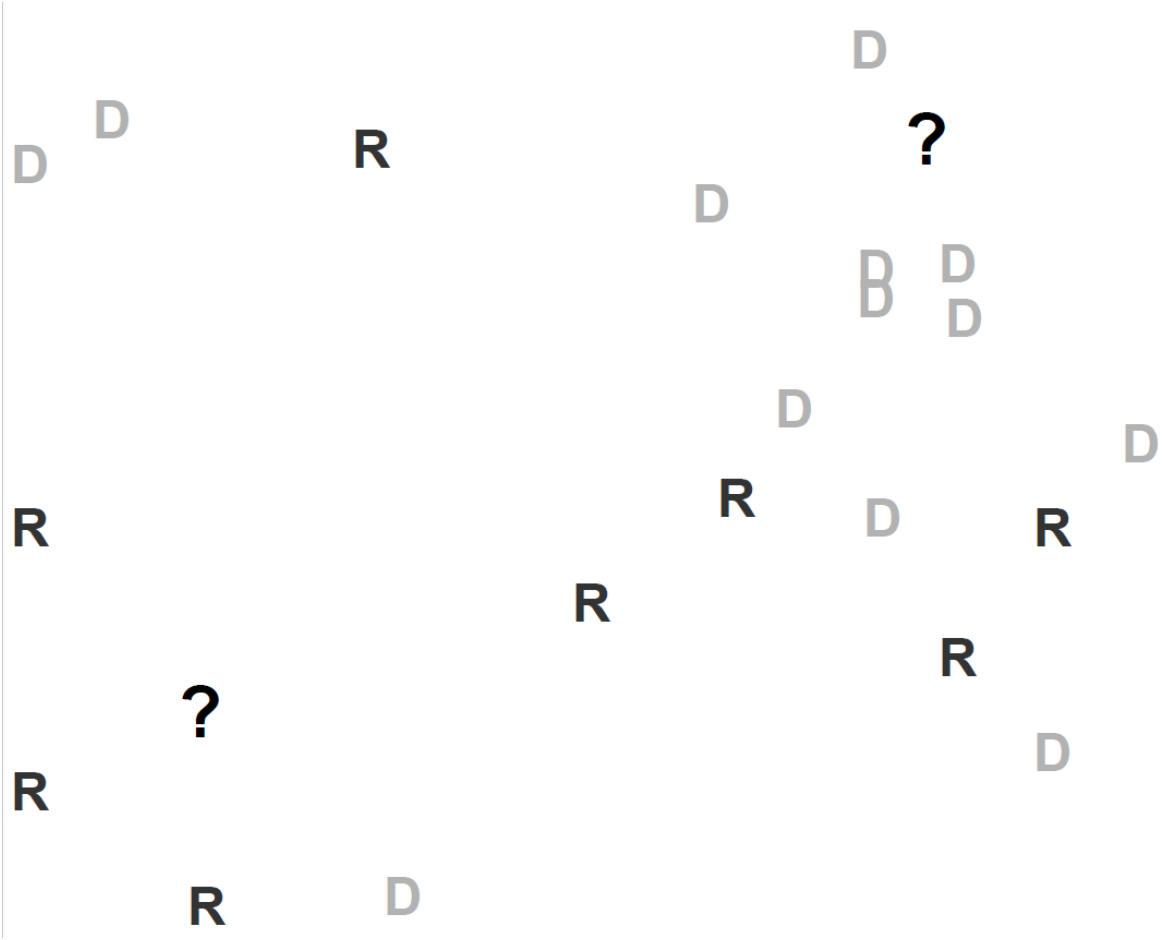
- What is an algorithm?
  - What do most people think the word "algorithm" means?
  - Why should most people care about that?
- The science of search engines
  - PageRank
  - Location-based indexing

If time: a glimpse into pattern recognition and machine learning

# A brief glimpse into pattern recognition and machine learning

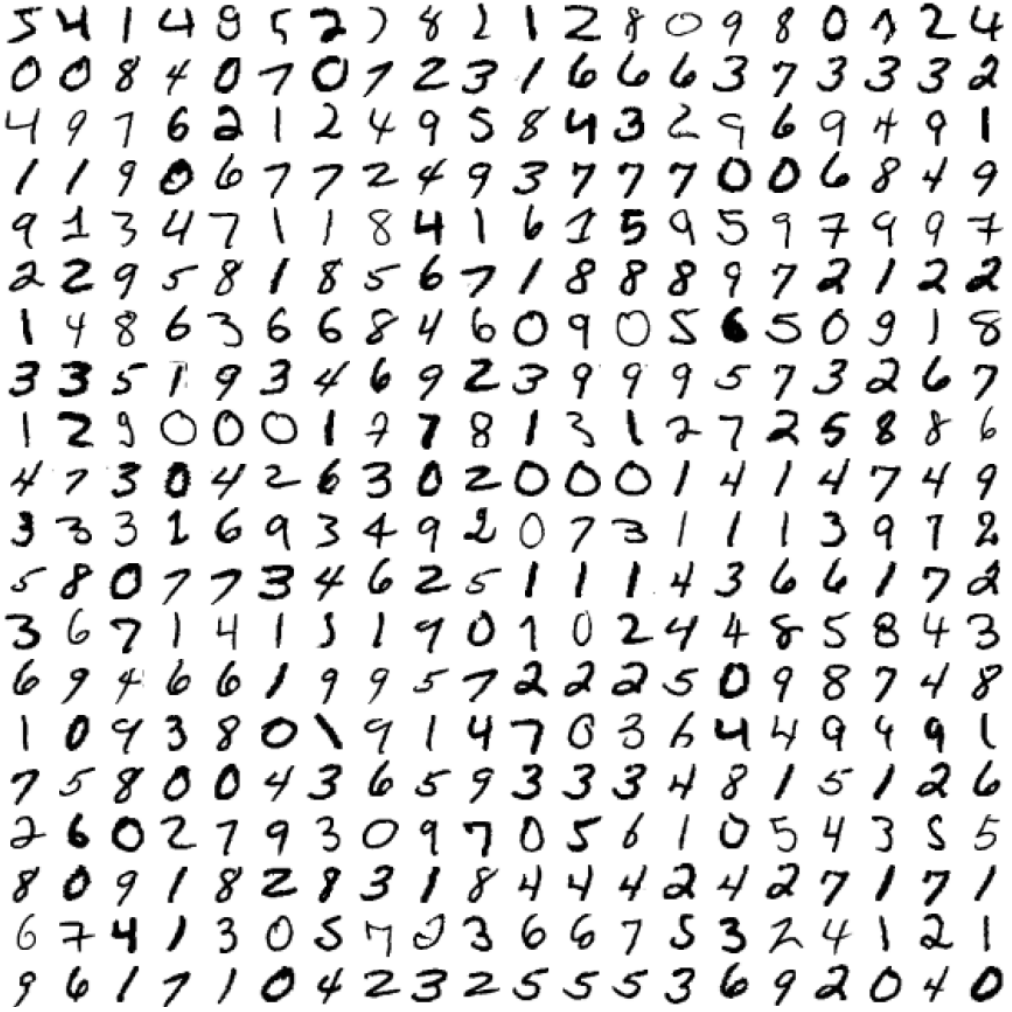
- Nearest neighbor classifiers
- Decision trees
- Neural networks

# Nearest neighbor classifier




Main idea: classify an unlabelled instance based on the nearest labeled instance

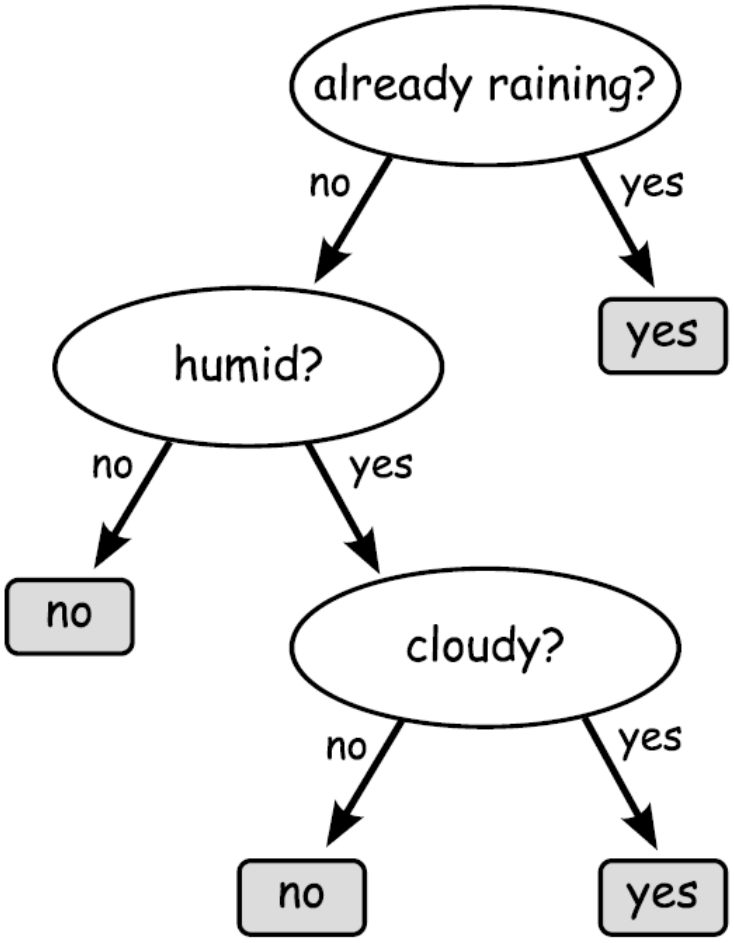
# Nearest neighbor gives surprisingly good results



Achieves 97% accuracy for recognizing handwritten digits

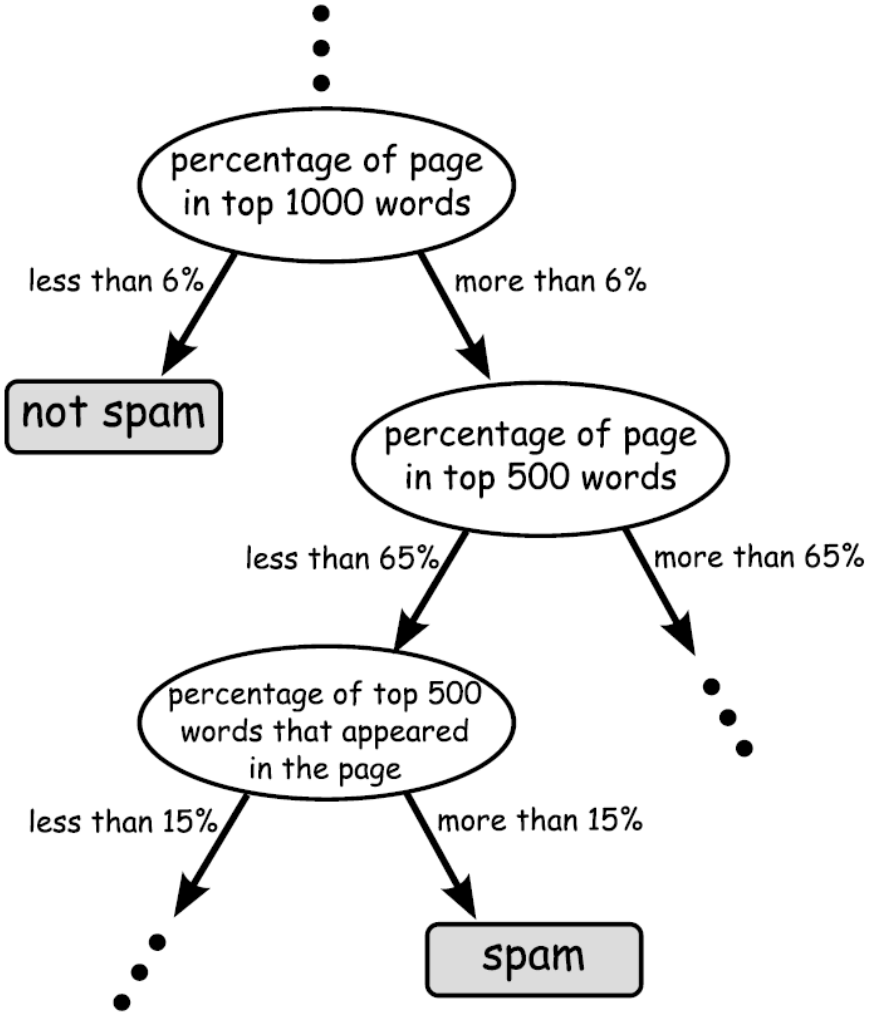
# Decision trees are another simple, surprisingly effective tool for classification

Decision tree for "should I take an umbrella?"



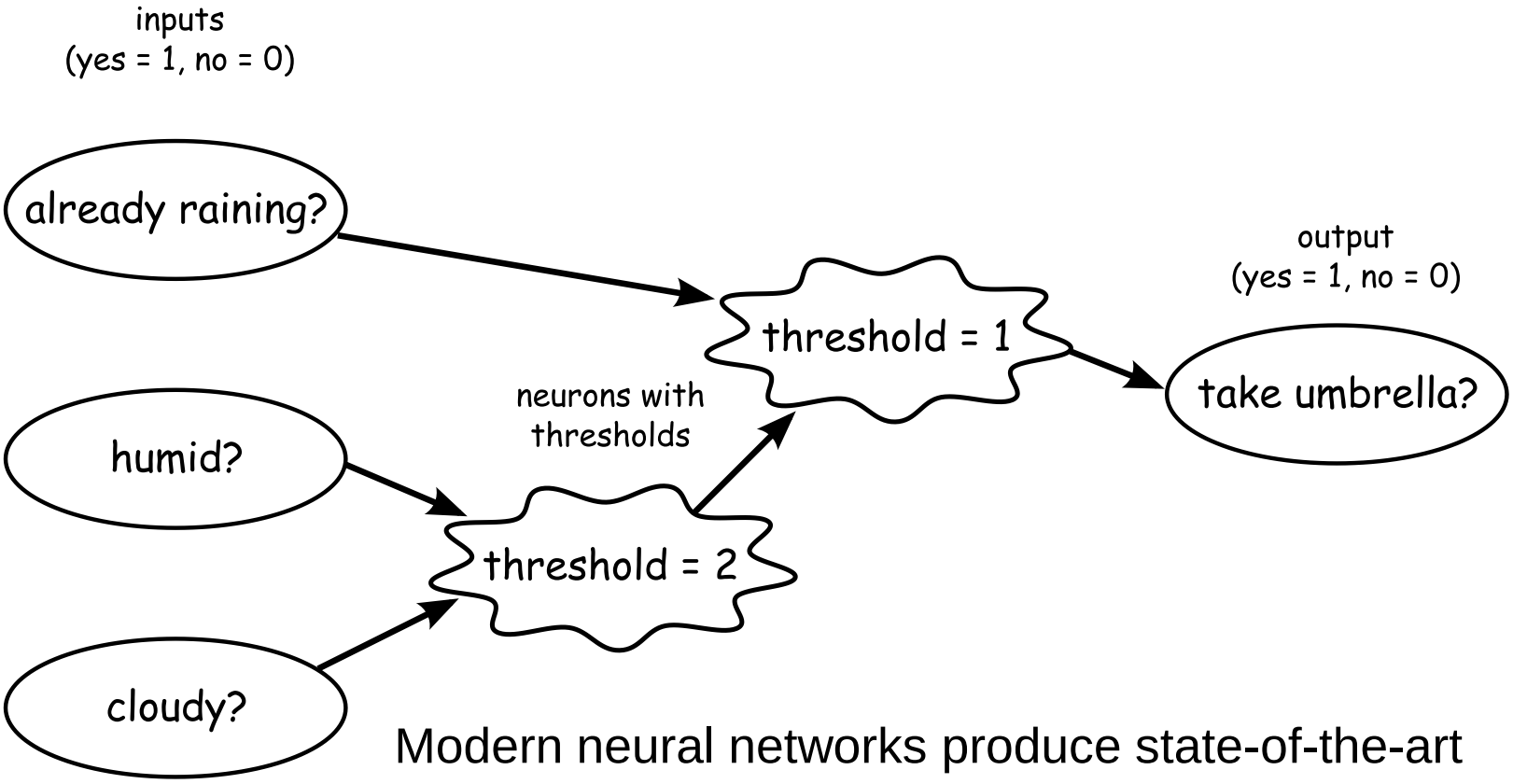
# Decision trees are another simple, surprisingly effective tool for classification

Part of a decision tree for “is this web page spam?”




Source: Ntoulas et al, 2006

# Neural network for “Should I take an umbrella?”




Modern neural networks produce state-of-the-art object recognition systems using hundreds of thousands of artificial neurons with millions of connections, connected in multiple layers (deep convolutional networks, e.g. Krizhevsky et al 2012).

# Overview of this lecture

- What is an algorithm?
  - What do most people think the word “algorithm” means?
  - Why should most people care about that?
- The science of search engines
  - PageRank
  - Location-based indexing

If time: a glimpse into pattern recognition and machine learning

thank you very much!

questions?