Data Mining: From thrillers to Tetris

Walter Kosters, Universiteit Leiden

Friday October 24, 2008
SIKS Computational Intelligence, Zeist

http://www.liacs.nl/home/kosters/zeist.pdf

Before the fact . . .

Crime novels

Crime novels deal with crime — in many ways. There are several sub-genres:

detective/ion Poe, Doyle, Christie (whodunnit)

psychological Rendell (whydunnit; inverted crime novels)

police procedural Sjöwall and Wahlöö, Simenon

private eye Hammett, Chandler

spy Ambler, Le Carré

thriller legal, historical, medical, literary, Da Vinci, . . .

. . .

Doyle (1893):

‘Is there any other point to which you would draw my attention?’

‘To the curious incident of the dog in the night-time.’

‘The dog did nothing in the night-time.’

‘That was the curious incident.’

Hammett (1927):

I first heard Personville called Poisonville by a red-haired mucker named Hickey Dewey in the Big Ship in Butte. He also called his shirt a shoit.

Rendell (1977):

Eunice Parchman killed the Coverdale family because she could not read or write.

From a scientific point of view you can approach crime novels from many angles:

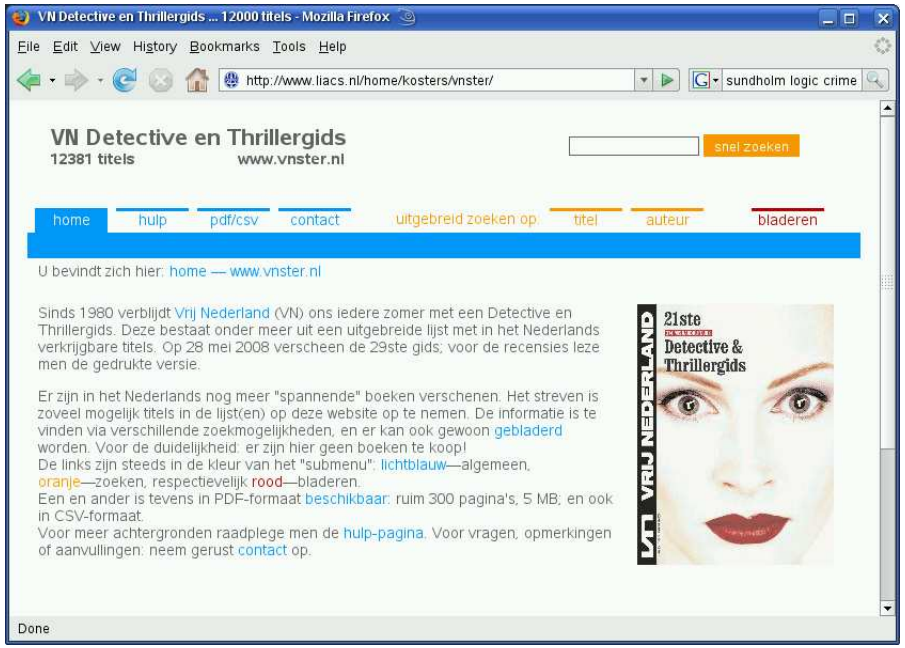
- Why do people like them?
- When is a novel good or bad?
- When is it literature?
- What are the boundaries between (sub-)genres?

But there are also computer science related issues like:

- What is the logic in a novel?
- How to build recommender systems?

Website

As a sort of escape from science, Hendrik Jan Hoogeboom and I built a website concerning crime novels, starting from the printed "Vrij Nederland Detective & Thrillergids": an extensive ranking system.



`www.vnster.nl`

How to recommend?

Data Mining

Data Mining tries to find interesting and (un)expected patterns in large amounts of (un)ordered data.

Cases (among others):

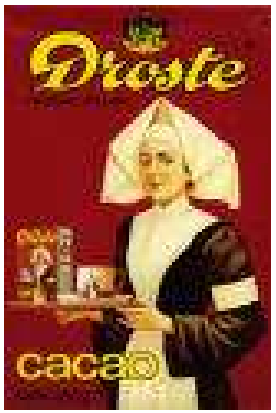
- Market basket analysis: What and how do we buy?
- Bio-informatics: What is common in DNA?

Problems (selection):

- need for expected results and for unexpected ones
- moving targets
- acquisition of data
- distance notion

The Data Mining process — or rather the KDD process, for Knowledge Discovery in Databases — is usually divided into the following steps:

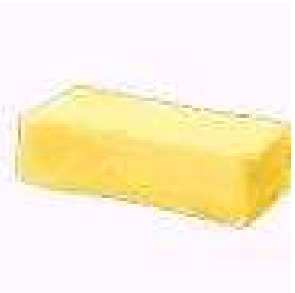
1. data selection
2. cleaning, e.g., de-duplication and domain consistency
3. enrichment, e.g., data fusion
4. coding
5. Data Mining — the real thing
   (no recursive pun intended)
6. reporting

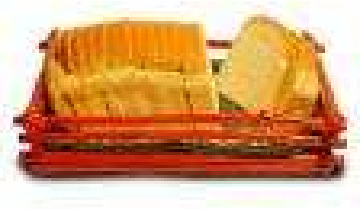


There are *many* Data Mining techniques, or rather algorithms that can be used to mine data:

- (don't forget:) statistical methods
- all sorts of machine learning techniques from AI: evolutionary computation, neural networks, Bayesian networks, . . .
- all sorts of clustering and classification techniques (decision trees, . . .)
- association rules
- ad-hoc methods

Association rules





We are interested in connections between (multi)sets of items: products, or molecules, or visits to websites. Rules should be like: "if you buy bread, you usually buy butter too". And: can you identify a customer by the products he/she buys? (Yes, to a large extent.)

Given a database with n records (customers, transactions), where every record consists of several items (products). The support of such a set of items is the number of records that contain that set, usually divided by n. A set with high support (above some threshold) is called frequent.

For example: 20% of the customers in a supermarket buy both bread, butter and cheese — and possibly more.

Now suppose that out of the α customers that buy a set A, β buy a set B too. Then we can say that the association rule $A \Rightarrow B$ has confidence β/α.

For example: it might be that 80% of the customers that buy butter and cheese, also buy bread.

Now we are interested in rules $A \Rightarrow B$ with both high confidence *and* high support, i.e., high support for the itemset $A \cup B$ (20% in the above case).

There is an extensive literature on association rules, in particular on the following aspects:

- efficient algorithms to find them . . .
- how to select the interesting ones . . .
- how to deal with non-Boolean attributes . . .

For this last issue one can use fuzzy logic, where instead of 0/1 (not-buy vs. buy) intermediate values can occur. For example: age can be "young" to an extent of 0.35.

Example

product = item customer = transaction	1	2	3	4	5	6	7	8	9
1	1	1	0	0	1	1	1	1	0
2	1	0	1	0	0	0	0	1	1
3	0	1	1	0	1	0	1	0	0
4	1	0	1	0	1	1	0	1	1
5	0	0	0	0	1	0	0	0	0
6	0	1	0	0	1	0	1	0	0

Even for this small database it is hard to see that the itemset $\{2, 5, 7\}$ is the only set with 3 items that is "bought" by at least 50% (some fixed bound, the so-called support threshold) of the customers.

Frequent itemsets naturally lead to association rules, like $\{2, 7\} \Rightarrow \{5\}$.

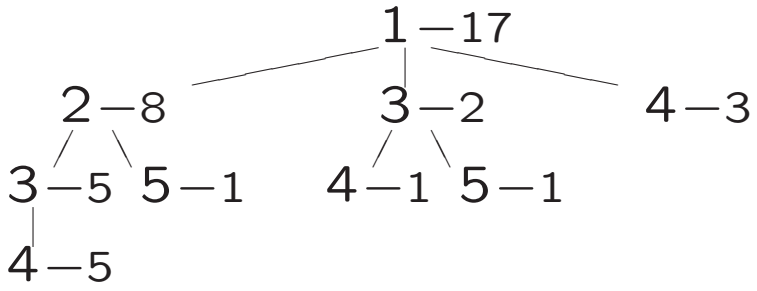
Most fast implementations rely on the following property:

A subset of a frequent set must be frequent too!

This is called the Apriori-property, because it is used in Apriori, one of the most famous Data Mining algorithms. The property is anti-monotone. It is applied like this: small frequent sets are the building blocks for larger ones: first you join them to make candidates, and for these candidates you compute the support.

The current fastest algorithms use FP-trees for condensed representations of the database.

An FP-tree, that contains a database, looks like:



Paths in the tree represent itemsets, including the number of customers that "buy" them. The example tree shows (among other things) that there are $5+1+3 = 9$ customers that "buy" the itemset $\{1, 4\}$ — so its (absolute) support equals 9.

Market basket analysis

Market basket analysis

Many stores try to analyze consumer behavior by doing so-called market basket analysis.
Some possible applications are:

- direct marketing
  "Which customer shall we present some special offer?"

- analysis of current catalog
  "Which products are of interest for a small shop, say in a railway station?"

- clustering and classification
  "Can we group (or recognize) customers by means of their behavior?"

A more general goal might just be a better understanding of the consumers.

Recognizing customers

It turned out that it was quite easy to recognize customers by what they buy. In order to do so, one has to define a symmetric distance measure between customers.

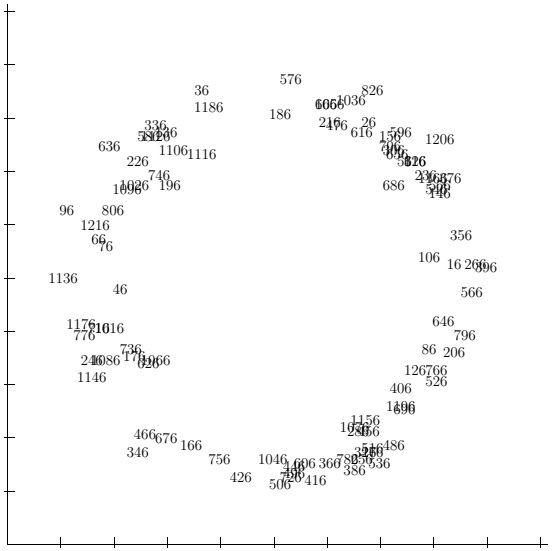
For instance, one can simply let $distance(Cust_1, Cust_2)$ be equal to the number of products that the two customers $Cust_1$ and $Cust_2$ have in common. This can be refined by weights, or by "punishing" differences. We could *for example* define (in set notation; $|Z|$ is number of elements in Z; $U \backslash V$ consists of elements in U that are not in V):

$$distance(X, Y) = (\, |Universe| - |X \cap Y| + |X \backslash Y| + |Y \backslash X| \,) \; / \; \ldots$$

You can "prove" this by the following experiment: if you visit a supermarket, observe the difference between your purchases and those from others, and remember the similarity between your current and previous baskets.

Clustering shops

A large chain of shops wanted to obtain insight in their "positioning", based on accumulated weekly sales. Pictures will look like:



Here we see 100 shops, clustered/visualized (using some Multi Dimensional Scaling technique as dimension reduction) in the unit square, with respect to their common weekly sales.

In order to cluster, one has to define a distance notion. Now suppose that two shops have the following sales lists for this week:

wine	bread	cheese	sausage	bananas
10	120	34	0	40
7	100	21	0	0

Their distance could be: $3 + 20 + 13 + 0 + 40 = 76$, or $3 + 20 + 13 = 36$, or $3/10 + 20/120 + 13/34$, or . . .
It seems reasonable to normalize for the total sales of a shop.

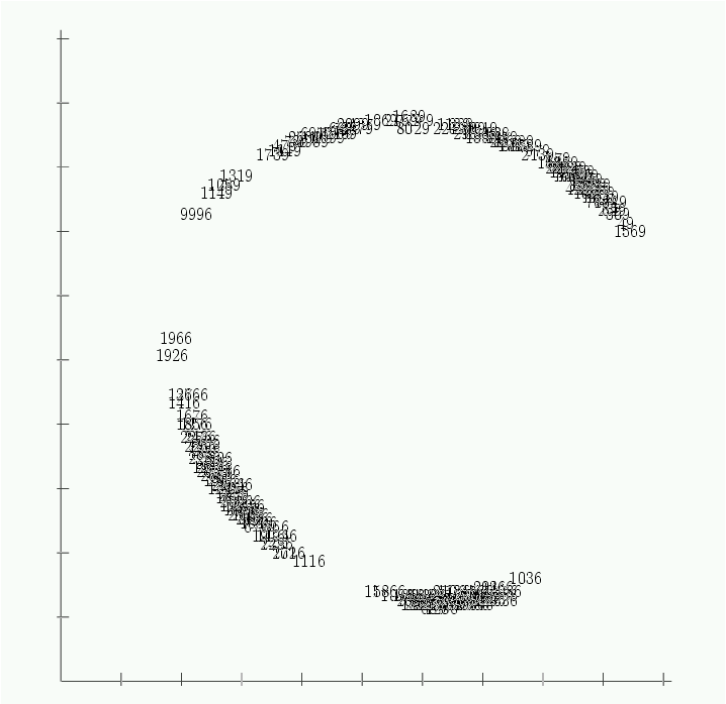
Anyway, there are many possibilities!

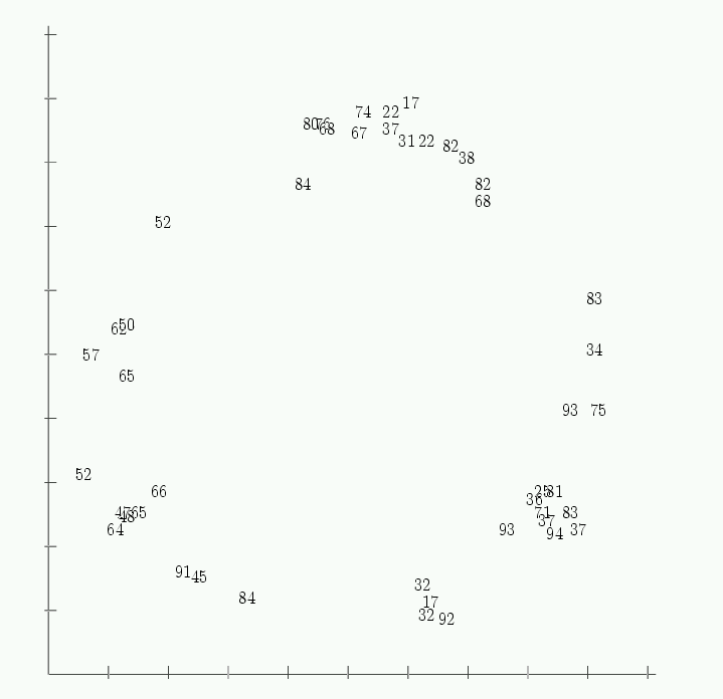
Clustering techniques can be

- hierarchical: repeatedly join smaller clusters to bigger ones
- non-hierarchical: improve existing clustering

We have chosen for a neural network like technique (cf. Self Organizing Maps (SOMs) and Multi Dimensional Scaling (MDS)).
Start from a random positioning of the data points. Then move (in random order) points toward one another if their relative distance is smaller than the Euclidean distance in their current position, and vice versa.

Left we see a clear distinction between shops with and without restaurant. Right we look at zip code; in that case one clearly distinguishes "eating habits".





In a supermarket there are *many* customers who can choose from a *large* stock. Association rules can be easily applied.

"Research" revealed that cigarette paper and tobacco are often bought together (what a surprise!), but also that special types of tobacco required special types of cigarette paper.

There is much interest in hierarchies. E.g., special brands in relation to more general categories.

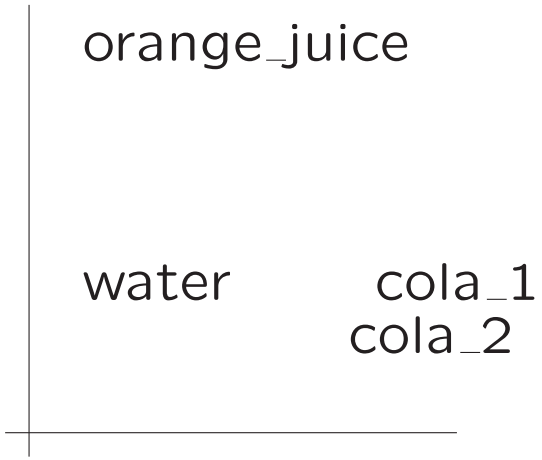
The diapers—beer connection is a fairy-tale.

Yet another interesting problem is the presentation of products in a shop. Is it possible to automate the displays?

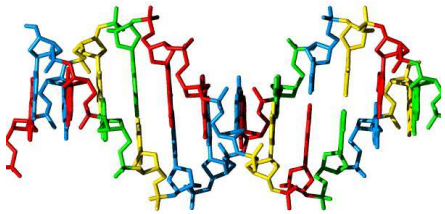
A possibility is to use Genetic Algorithms (Evolutionary Computation). We evolve candidate solutions by means of operators like selection, mutation and crossover. A solution (i.e., a possible display of all products) must be properly judged: the so-called fitness function. This is a sum of several properties (spare room, adjacent colors, . . .).

A final application in a supermarket environment is the following. We would like to group products, e.g., beverages, in such a way that products are near if they can "replace" one another.

This can be derived from the shopping carts: such products are hardly ever sold together. Similar techniques as above lead to pictures like:



Bio-informatics

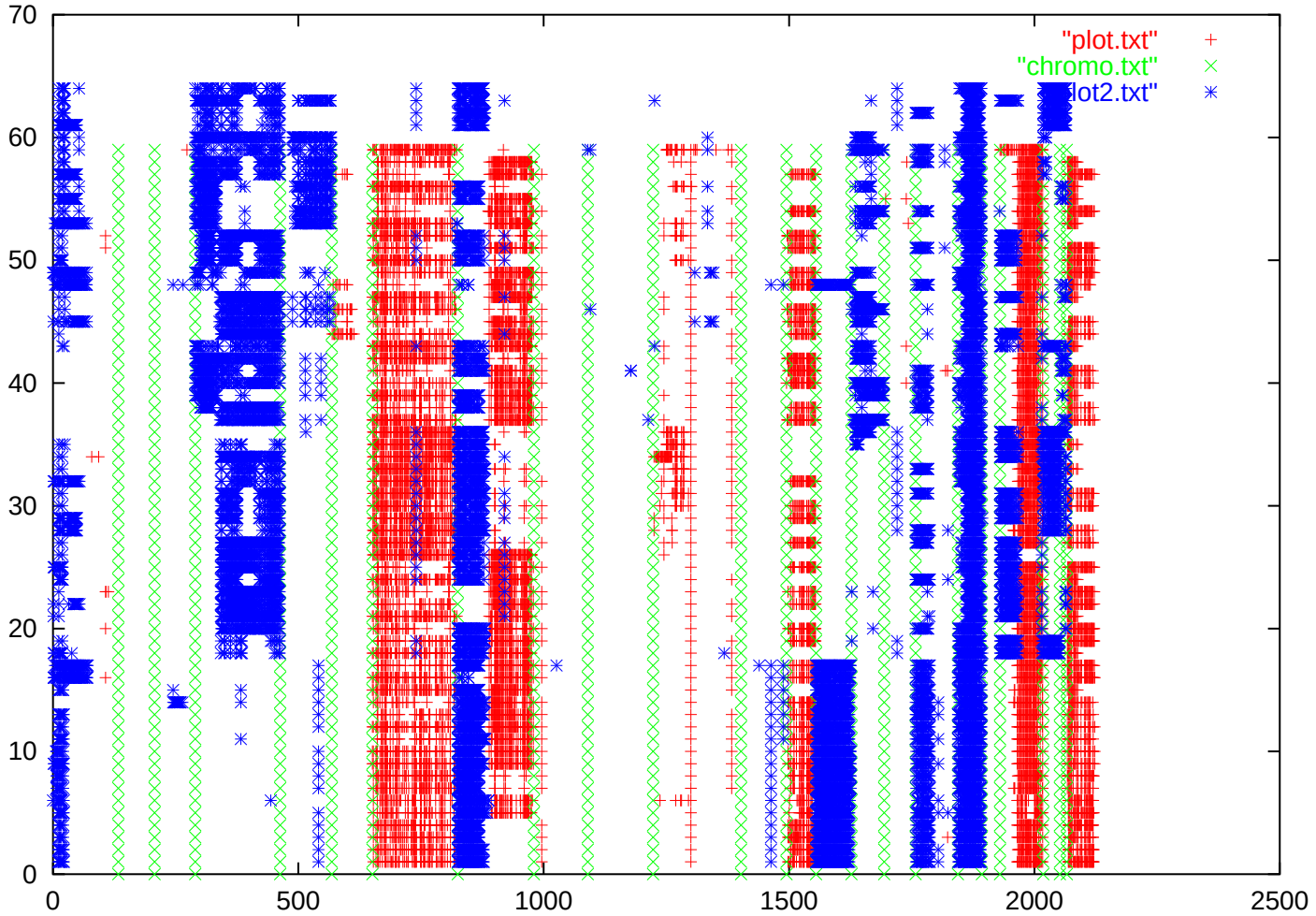


Many questions from Bio-informatics can be considered as Data Mining problems. Let us examine a few of them.

In a somewhat simplified way, DNA can be viewed as a string of characters from the alphabet $\{A, C, G, T\}$. Human DNA contains $\approx$ 3,000,000,000 characters — the human genome, divided among some 20+ chromosomes.

Now suppose that for some 2,000+ particular substrings, evenly spread over the human genome, we know how much they are present in 150 patients, say. Here, "how much" is a real number somewhere between -5 ("loss") and $+5$ ("amplification"). How to mine this heap of data?

Plot



chromosome boundaries, amplifications, losses

Explanation

In the plot we see many groups of patients, where each group behaves more or less the same on the 2,000+ given substrings ("clones").

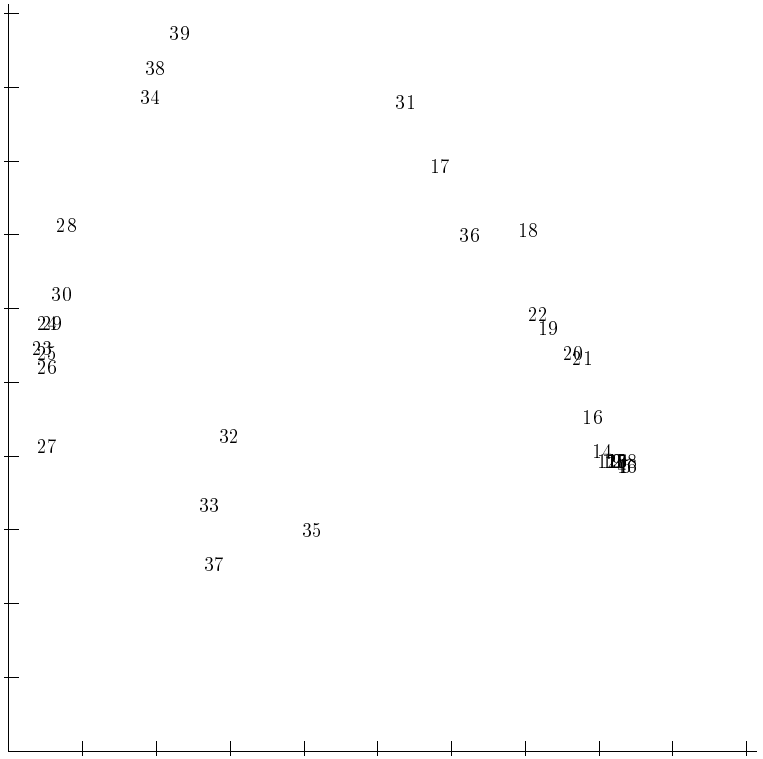
A group of patients is selected (is "frequent") if it satisfies the following property: all patients within the group have > 0.225 (some amplification) on the same c clones, where c is at least some threshold value. This yields the red data points. There are ≈ 60 groups shown.
Analogous for the losses, in blue.

So what we see are — loosely speaking — groups of patients having some (maximal) common behavior.

How to achieve these results — remembering the six KDD steps? And how to interpret them?

The association rule algorithm is easily applied to the data (after writing some Perl conversion script). It then takes some time to tune the parameters, such as the support threshold. After that the presentation deserves attention, see the previous plot.

It appeared that domain experts (biologists from the medical center) would like to see coherence — if any — in the frequent sets. This suggests a clustering. We can use the same technique as before, see the next plot (patients look like products ;-).



We see 39 "frequent sets of patients" being clustered in the unit square with respect to some distance notion: two such sets are near if they behave similar on chromosomes.

Another question from the biologists was: Generate X-thousand substrings of the human genome, such that:

- they are substantially far apart (but not too far)
- they are unique
- they contain "sufficiently many" C's and G's
- they react at the right temperature
- they do not "interfere"
- ...

This requires ad-hoc methods. And suffix trees/arrays ...

The suffix array of a string is the *lexicographically sorted array of all its suffixes*. Usually we give the indexes where the suffixes begin.

Example: the string `example` has 7 (non-empty) suffixes:

`ample`, `e`, `example`, `le`, `mple`, `ple`, `xample`.

So the array is $[2, 6, 0, 5, 3, 4, 1]$.

Suffix trees and suffix arrays are great when one wants to find, e.g., all overlaps in a large set of strings.

And I?

And why do *I* like crime novels?

Probably because of the suspense and the vivid dialogue, but also because of the puzzles in the "plot": what is the explanation of the events so far?

By the way, this often results in perhaps artificial and sometimes rather far-fetched scenarios, e.g., so-called locked-room mysteries.

Games

Babson task
Yarosh, Tim Krabbé

retrogade analyse
Smullyan

Games and puzzles give rise to complicated search algorithms, for example in chess, go, sudoku, . . . In particular, if chance and contingency play a role (poker), you have serious problems: Artificial Intelligence (AI)!

Often an evaluation function is used, to judge situations.

Many programs prune: they try to eliminate parts of the search space.

We just mention the minimax algorithm from Von Neumann (1928) and the α-β-algorithm from 1956/58. And there many others . . .

Types of games

Games can be distinguished as follows:

	deterministic	chance
perfect information	chess, connect-four, go, checkers, sudoku, othello	monopoly, backgammon
incomplete information	battleships, mastermind	bridge, poker, scrabble

Deterministic: move consequences are fixed, no chances.
Perfect information: players know everything of a state.

Chess has an extensive literature.
Currently, the ultimate challenge seems the game go.

There are many different strategies to determine (an approximation of) the "game-theoretic value" of a game. Shannon (1950; entropy, information theory) conceived three types:

type A compute everything up to a certain depth ("game tree"), and use an evaluation function there

type B sometimes go deeper (if it is turbulent: "quiescence"); use "heuristic" function gor guidance

type C goal oriented human search

A and B are brute-force, C is more "knowledge-based".

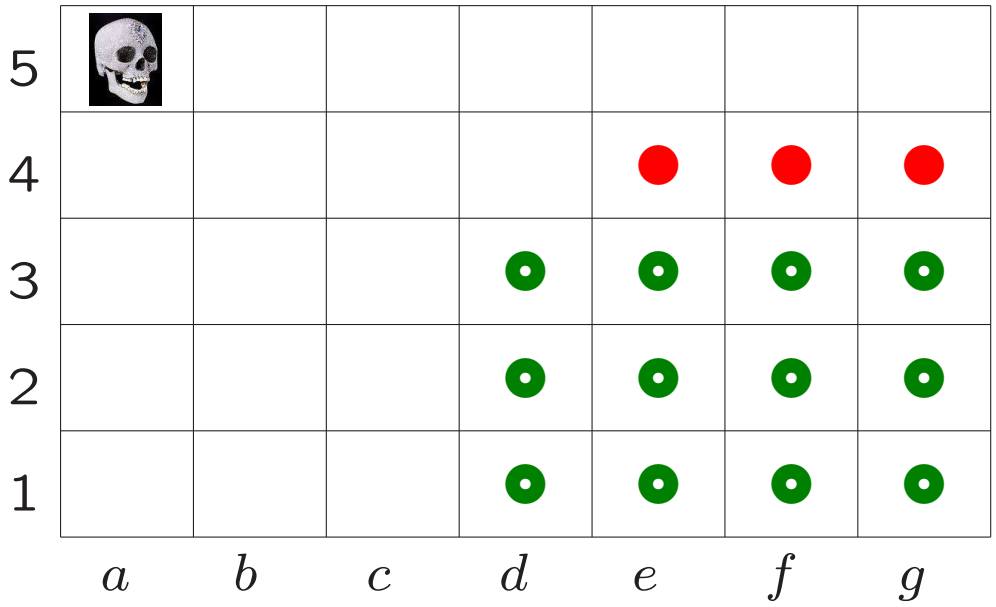
Some people distinguish three levels of game solving:

ultra-weak the game-theoretic value of the initial state is known: "one can win connect-four"

weak as above, and an optimal strategy is known: "start in the middle column, . . . ", see later

strong in every legal position an optimal strategy is known

The game of Chomp is played on a rectangular bar of chocolate, of which the players, taking turns, eat a rectangular part at the bottom right. He/she who eats the (poisonous) top left piece, has lost.



first bite: $d3$

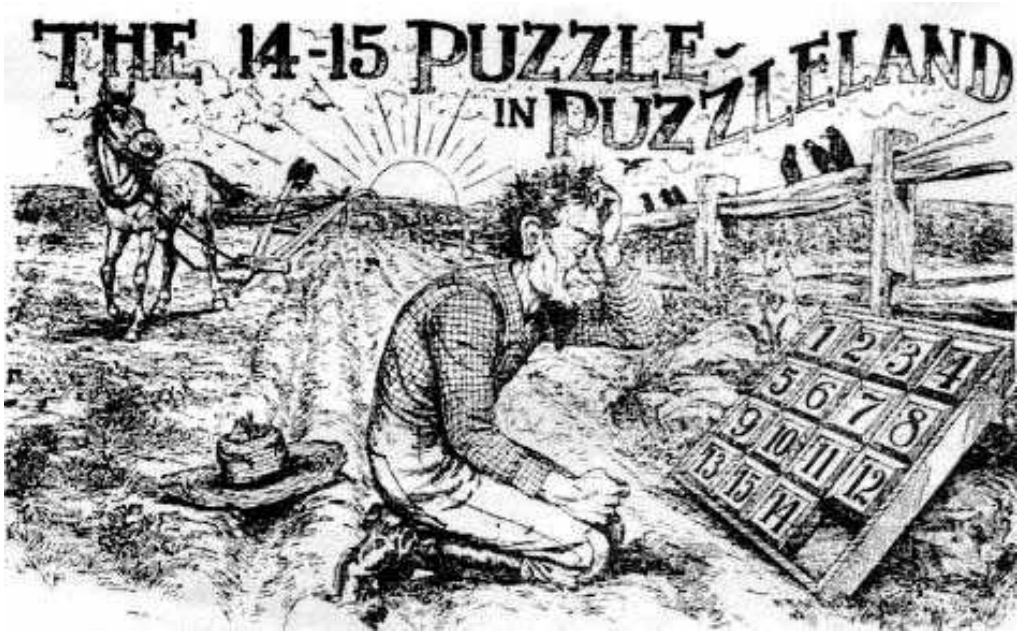
second bite: $e4$

Claim: if you start, you can always win at Chomp! Indeed, if you can win by taking the piece at the bottom right, then it is fine. If not, the adversary clearly has a (for him/her) winning bite. You could have taken that same bite in the first place, and win!
This argument is called strategy stealing.

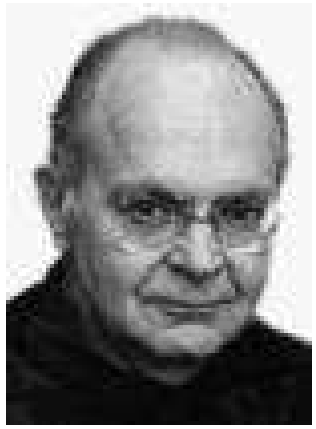
So Chomp is solved ultra-weakly, we do not know the real winning move . . .

For example, for 2×2 and 2×3 Chomp the bottom right piece "wins" (more general for squares: take the piece to the bottom right of the poisonous one); for 3×4 Chomp take the piece from the middle row, third column.

Mathematical plays



Knuth and Conway

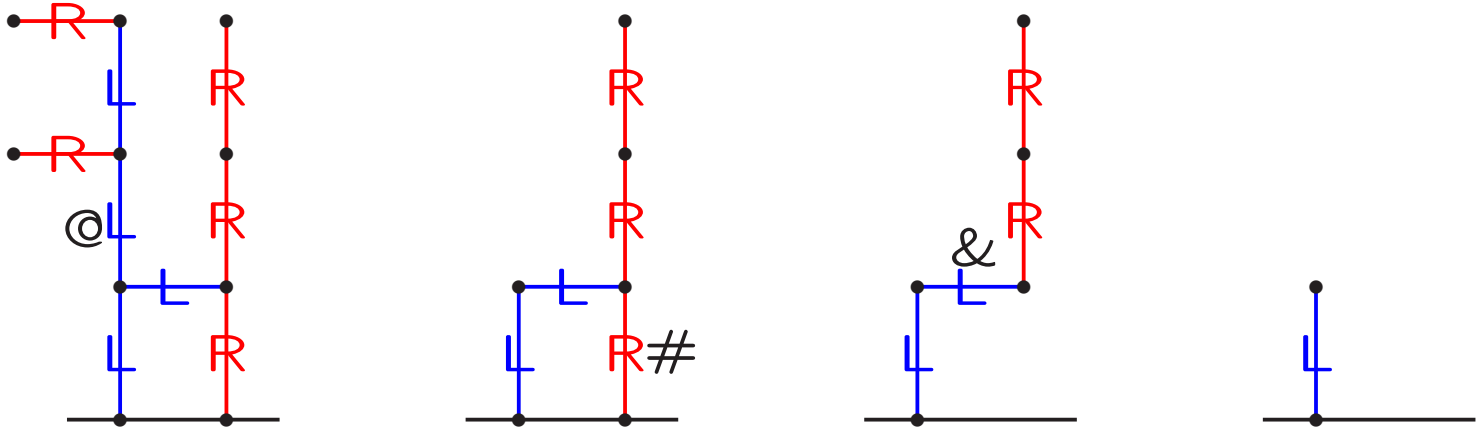


Donald E.(Ervin) Knuth
1938, US
NP; KMP
TEX
change-ringing; 3:16
The Art of Computer Programming

John H.(Horton) Conway
1937, UK → US
Co_1, Co_2, Co_3
Doomsday algorithm
game of Life; Angel problem
Winning Ways for your Mathematical Plays

Surreal numbers

When playing Hackenbush Left and Right take turns, and remove a bLue or a Red line, respectively, after which all lines that lose contact with the ground are removed. *If you cannot play, you have lost!*

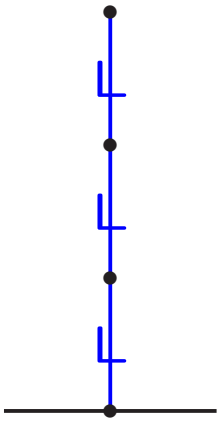


Left chooses @, Right chooses # (silly), Left chooses & and wins since Right cannot play.

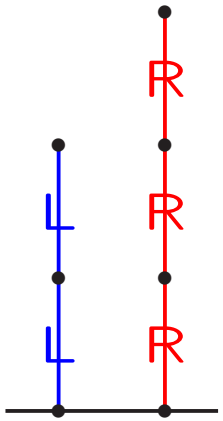
By the way, Right can win here, whoever starts!

Value of a position

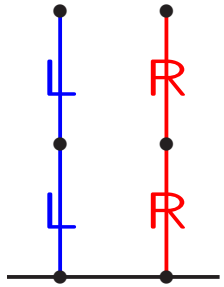
When playing Hackenbush, what is the value of a position?



value 3



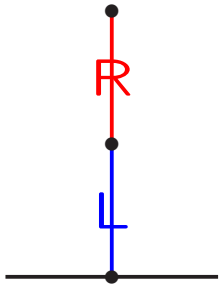
value $2-3=-1$



value $2-2=0$

If the value is positive (>0), Left *can* always win (whoever begins; in the example on the left with advantage 3), if the value is negative (<0) Right *can* always win, and if the value is 0 the player to begin loses.

But what is the value of this position?



If Left begins, he/she wins immediately; if Right begins, Left has another move, and also wins. So Left always wins. Therefore, the value is > 0.

Question: is the value equal to 1?

If the value in the left hand side position would be 1, de value of the right hand side position must be $1+(-1)=0$, and the player to begin should lose here. Is this true?

No: if Left begins, Left loses, and if Right begins Right can also win. So Right always wins (i.e., can always win), and therefore the right hand side position is < 0, and the left one is between 0 and 1.

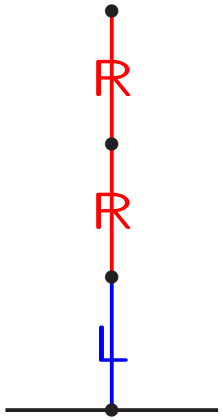
We denote the value of left hand side position by $\{\, 0 \mid 1 \,\}$.

Note that the right hand side position does have value 0: the player to begin loses. And so we have:

$$\{\, 0 \mid 1 \,\} + \{\, 0 \mid 1 \,\} + (-1) = 0,$$

and apparently $\{\, 0 \mid 1 \,\} = 1/2$.

We denote the value of a position where Left can play to (values of) positions from the set L and Right can play to (values of) positions from the set R by $\{\, L \mid R \,\}$.
An example:



The value is $\{\, 0 \mid \frac{1}{2}, 1 \,\} = \frac{1}{4}$.

The value appears to be the "simplest" number that is between left and right set.

In this way we define surreal numbers: "neat" tuples of sets of surreal numbers, defined earlier.

We start with $0 = \{\ \emptyset \mid \emptyset\ \} = \{\ \text{NOTHING} \mid \text{NOTHING}\ \} = \{\ \mid\ \}$: the game where the player to begin has no opportunities at all, and therefore loses.

And then $1 = \{\ 0 \mid\ \}$ and $-1 = \{\ \mid 0\ \}$.
And $42 = \{\ 41 \mid\ \}$.
And $\frac{3}{8} = \{\ \frac{1}{4} \mid \frac{1}{2}\ \}$.
And $\pi = \{\ 3, 3\frac{1}{8}, 3\frac{9}{64}, \ldots \mid 4, 3\frac{1}{2}, 3\frac{1}{4}, 3\frac{3}{16}, 3\frac{5}{32}, \ldots\ \}$.

The real numbers (the set $\mathbf{R}$) are special surreal numbers (the set $\mathbf{S}$).

The so-called Dali-function $\delta : \mathbf{R} \to \mathbf{S}$ takes care of the "embedding": $\delta(1) = \{\ 0 \mid \} = 1$. Bu there is more . . .

`www.tondering.dk/claus/surreal.html`

For example, we define:

$$\varepsilon = \{\, 0 \mid \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots \,\},$$

an "incredibly small positive number", and

$$\omega = \{\, 0, 1, 2, 3, \ldots \mid \,\} = \{\, \mathbf{Z} \mid \emptyset \,\},$$

a "terribly large number, some sort of ∞".

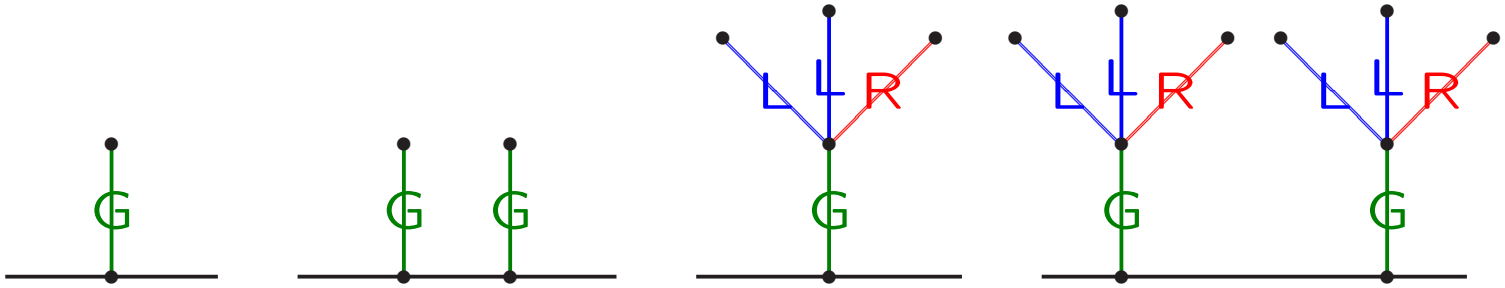
Then it turns out that:

$$\varepsilon \cdot \omega = 1$$

— if you have a well-defined multiplication . . .

And then you also have $\omega + 1$, $\sqrt{\omega}$, ω^ω, $\varepsilon/2$, and so on!

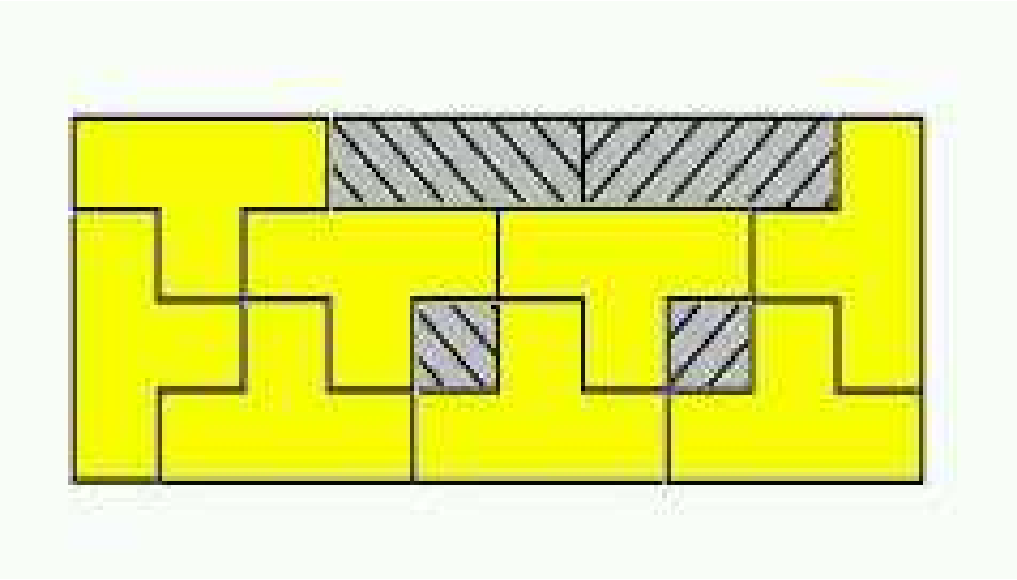
For Hotchpotch-Hackenbush we also have Green lines, that can be taken by *both* players.



The lefmost position has value $* = \{ 0 \mid 0 \}$ (that appears to be not a surreal number!), because the player to begin can win by taking the Green line. The second from the left is $* + * = 0$ (player to begin loses).

The second from the right is again a win for the player to begin. The rightmost position is a win for Left (whoever begins), and so is > 0.

After the fact . . .



Tetris

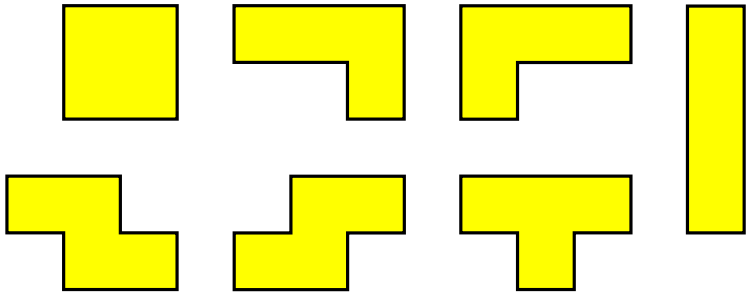
There are several interesting questions attached to a game like Tetris (no connection with Data Mining ;-):

- How to play well? (AI — Artificial Intelligence)
- How hard is it? (complexity)
- What might happen?

It has been shown that certain Tetris-problems are NP-complete (joint work with researchers from MIT), that you can reach almost all configurations, but that not all problems are "decidable".

Tetris: NP-complete?

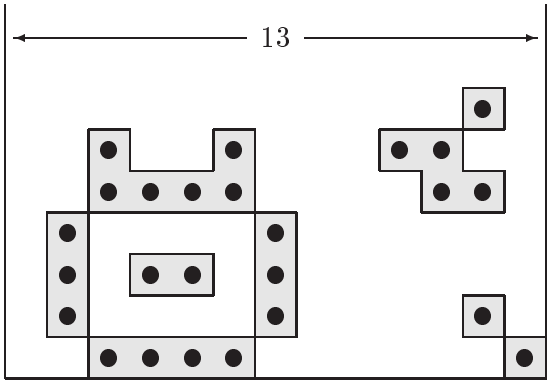
The 7 Tetris-pieces:



Random pieces fall down, and filled lines are cleared.
The question "Is it possible, given a finite ordered series of these pieces, to clear a partially filled game board?" is NP-complete.
If someone clears the board, this is easy to verify. If clearing is *not* possible however, up till now the only thing one can do to prove this is to check all possibilities, one by one!

Tetris: reachable?

An "arbitrary" configuration:



This figure can be made by dropping 276 suitable Tetris-pieces in the appropriate way, see

`http://www.liacs.nl/home/kosters/tetris/`

Claim: on a game board of odd width every configuration is reachable.